Scalable Computing: Practice and Experience

Scientific International Journal for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest din Timişoara

Volume 20(2) June 2019

SUBSCRIPTION INFORMATION: please visit http://www.scpe.org

Scalable Computing: Practice and Experience

Volume 20, Number 2, June 2019

TABLE OF CONTENTS

DOI 10.12694/scpe.v20i2.1558
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. iii–iv. http://www.scpe.org © 2019 SCPE

INTRODUCTION TO THE SPECIAL ISSUE ON RECENT TRENDS AND FUTURE OF FOG AND EDGE COMPUTING, SERVICES, AND ENABLING TECHNOLOGIES

Cloud computing has been established as the most popular as well as suitable computing infrastructure providing on-demand, scalable and pay-as-you-go computing resources and services for the state-of-the-art ICT applications which generate a massive amount of data. Though Cloud is certainly the most fitting solution for most of the applications with respect to processing capability and storage, it may not be so for the real-time applications. The main problem with Cloud is the latency as the Cloud data centres typically are very far from the data sources as well as the data consumers. This latency is ok with the application domains such as enterprise or web applications, but not for the modern Internet of Things (IoT)-based pervasive and ubiquitous application domains such as autonomous vehicle, smart and pervasive healthcare, real-time traffic monitoring, unmanned aerial vehicles, smart building, smart city, smart manufacturing, cognitive IoT, and so on. The prerequisite for these types of application is that the latency between the data generation and consumption should be minimal. For that, the generated data need to be processed locally, instead of sending to the Cloud. This approach is known as Edge computing where the data processing is done at the network edge in the edge devices such as set-top boxes, access points, routers, switches, base stations etc. which are typically located at the edge of the network. These devices are increasingly being incorporated with significant computing and storage capacity to cater to the need for local Big Data processing. The enabling of Edge computing can be attributed to the Emerging network technologies, such as 4G and cognitive radios, high-speed wireless networks, and energy-efficient sophisticated sensors.

Different Edge computing architectures are proposed (e.g., Fog computing, mobile edge computing (MEC), cloudlets, etc.). All of these enable the IoT and sensor data to be processed closer to the data sources. But, among them, Fog computing, a Cisco initiative, has attracted the most attention of people from both academia and corporate and has been emerged as a new computing-infrastructural paradigm in recent years. Though Fog computing has been proposed as a different computing architecture than Cloud, it is not meant to replace the Cloud. Rather, Fog computing extends the Cloud services to network edges for providing computation, networking, and storage services between end devices and data centres. Ideally, Fog nodes (edge devices) are supposed to pre-process the data, serve the need of the associated applications preliminarily, and forward the data to the Cloud if the data are needed to be stored and analysed further.

Fog computing enhances the benefits from smart devices operational not only in network perimeter but also under cloud servers. Fog-enabled services can be deployed anywhere in the network, and with these services provisioning and management, huge potential can be visualized to enhance intelligence within computing networks to realize context-awareness, high response time, and network traffic offloading. Several possibilities of Fog computing are already established. For example, sustainable smart cities, smart grid, smart logistics, environment monitoring, video surveillance, etc.

To design and implementation of Fog computing systems, various challenges concerning system design and implementation, computing and communication, system architecture and integration, application-based implementations, fault tolerance, designing efficient algorithms and protocols, availability and reliability, security and privacy, energy-efficiency and sustainability, etc. are needed to be addressed. Also, to make Fog compatible with Cloud several factors such as Fog and Cloud system integration, service collaboration between Fog and Cloud, workload balance between Fog and Cloud, and so on need to be taken care of.

It is our great privilege to present before you Volume 20, Issue 2 of the Scalable Computing: Practice and Experience. We had received 20 Research Papers and out of which 14 Papers are selected for Publication. The aim of this special issue is to highlight Recent Trends and Future of Fog and Edge Computing, Services and Enabling technologies. The special issue will present new dimensions of research to researchers and industry professionals with regard to Fog Computing, Cloud Computing and Edge Computing.

Sujata Dash et al. contributed a paper titled Edge and Fog Computing in Healthcare- A Review in which an in-depth review of fog and mist computing in the area of health care informatics is analysed, classified and discussed. The review presented in this paper is primarily focussed on three main aspects: The requirements of IoT based healthcare model and the description of services provided by fog computing to address then. The architecture of an IoT based health care system embedding fog computing layer and implementation of fog computing layer services along with performance and advantages. In addition to this, the researchers have highlighted the trade-off when allocating computational task to the level of network and also elaborated various challenges and security issues of fog and edge computing related to healthcare applications.

Parminder Singh et al. in the paper titled Triangulation Resource Provisioning for Web Applications in Cloud Computing: A Profit-Aware proposed a novel triangulation resource provisioning (TRP) technique with a profit-aware surplus VM selection policy to ensure fair resource utilization in hourly billing cycle while giving the quality of service to end-users. The proposed technique use time series workload forecasting, CPU utilization and response time in the analysis phase. The proposed technique is tested using CloudSim simulator and R language is used to implement prediction model on ClarkNet weblog. The proposed approach is compared with two baseline approaches i.e. Cost-aware (LRM) and (ARMA). The response time, CPU utilization and predicted request are applied in the analysis and planning phase for scaling decisions. The profit-aware surplus VM selection policy used in the execution phase for select the appropriate VM for scale-down. The result shows that the proposed model for web applications provides fair utilization of resources with minimum cost, thus provides maximum profit to application provider and QoE to the end users.

Akshi Kumar and Abhilasha Sharma in the paper titled Ontology driven Social Big Data Analytics for Fog enabled Sentic-Social Governance utilized a semantic knowledge model for investigating public opinion towards adaption of fog enabled services for governance and comprehending the significance of two s-components (sentic and social) in aforesaid structure that specifically visualize fog enabled Sentic-Social Governance. The results using conventional TF-IDF (Term Frequency-Inverse Document Frequency) feature extraction are empirically compared with ontology driven TF-IDF feature extraction to find the best opinion mining model with optimal accuracy. The results concluded that implementation of ontology driven opinion mining for feature extraction in polarity classification outperforms the traditional TF-IDF method validated over baseline supervised learning algorithms with an average of 7.3

Avinash Kaur, Pooja Gupta and Manpreet Singh in the paper titled Hybrid Balanced Task Clustering Algorithm for Scientific workflows in Cloud Computing proposed novel hybrid balanced task clustering algorithm using the parameter of impact factor of workflows along with the structure of workflow and using this technique, tasks can be considered for clustering either vertically or horizontally based on value of impact factor. The testing of the algorithm proposed is done on Workflowsim- an extension of CloudSim and DAG model of workflow was executed. The Algorithm was tested on variables- Execution time of workflow and Performance Gain and compared with four clustering methods: Horizontal Runtime Balancing (HRB), Horizontal Clustering (HC), Horizontal Distance Balancing (HDB) and Horizontal Impact Factor Balancing (HIFB) and results stated that proposed algorithm is almost 5-10

Pijush Kanti Dutta Pramanik et al. in the paper titled Green and Sustainable High-Performance Computing with Smartphone Crowd Computing: Benefits, Enablers and Challenges presented a comprehensive statistical survey of the various commercial CPUs, GPUs, SoCs for smartphones confirming the capability of the SCC as an alternative to HPC. An exhaustive survey is presented on the present and optimistic future of the continuous improvement and research on different aspects of smartphone battery and other alternative power sources which will allow users to use their smartphones for SCC without worrying about the battery running out.

A. Dhanapal and P. Nithyanandam in the paper titled The Slow HTTP Distributed Denial of Service (DDOS) Attack Detection in Cloud proposed a novel method to detect slow HTTP DDoS attacks in cloud to overcome the issue of consuming all available server resources and making it unavailable to the real users. The proposed method is implemented using OpenStack cloud platform with slowHTTPTest tool. The results stated that proposed technique detects the attack in efficient manner.

Mandeep Kaur and Rajni Mohana in the paper titled Static Load Balancing Technique for Geographically partitioned Public Cloud proposed a novel approach focused upon load balancing in the partitioned public cloud by combining centralized and decentralized approaches, assuming the presence of fog layer. A load balancer entity is used for decentralized load balancing at partitions and a controller entity is used for centralized level to balance the overall load at various partitions. Results are compared with First Come First Serve (FCFS) and Shortest Job First (SJF) algorithms. In this work, the researchers compared the Waiting Time, Finish Time and Actual Run Time of tasks using these algorithms. To reduce the number of unhandled jobs, a new load state is introduced which checks load beyond conventional load states. Major objective of this approach is to reduce the need of runtime virtual machine migration and to reduce the wastage of resources, which may be occurring due to predefined values of threshold.

Mukta and Neeraj Gupta in the paper titled Analytical Available Bandwidth Estimation in Wireless Ad-Hoc Networks considering Mobility in 3-Dimensional Space proposes an analytical approach named Analytical Available Bandwidth Estimation Including Mobility (AABWM) to estimate ABW on a link. The major contributions of the proposed work are: i) it uses mathematical models based on renewal theory to calculate the collision probability of data packets which makes the process simple and accurate, ii) consideration of mobility under 3-D space to predict the link failure and provides an accurate admission control. To test the proposed technique, the researcher used NS-2 simulator to compare the proposed technique i.e. AABWM with AODV, ABE, IAB and IBEM on throughput, Packet loss ratio and Data delivery. Results stated that AABWM performs better as compared to other approaches.

R. Sridharan and S. Domnic in the paper titled Placement Strategy for Intercommunicating Tasks of an Elastic Request in Fog-Cloud Environment proposed a novel heuristic IcAPER,(Inter-communication Aware Placement for Elastic Requests) algorithm. The proposed algorithm uses the network neighborhood machine for placement, once current resource is fully utilized by the application. The performance IcAPER algorithm is compared with First Come First Serve (FCFS), Random and First Fit Decreasing (FFD) algorithms for the parameters (a) resource utilization (b) resource fragmentation and (c) Number of requests having intercommunicating tasks placed on to same PM using CloudSim simulator. Simulation results shows IcAPER maps 34% more tasks on to the same PM and also increase the resource utilization by 13% while decreasing the resource fragmentation by 37.8% when compared to other algorithms.

Velliangiri S. et al. in the paper titled Trust factor based key distribution protocol in Hybrid Cloud Environment proposed a novel security protocol comprising of two stages: first stage, Group Creation using the trust factor and develop key distribution security protocol. It performs the communication process among the virtual machine communication nodes. Creating several groups based on the cluster and trust factors methods. The second stage, the ECC (Elliptic Curve Cryptography) based distribution security protocol is developed. The performance of the Trust Factor Based Key Distribution protocol is compared with the existing ECC and Diffie Hellman key exchange technique. The results state that the proposed security protocol has more secure communication and better resource utilization than the ECC and Diffie Hellman key exchange technique in the Hybrid cloud.

Vivek Kumar Prasad et al. in the paper titled Influence of Monitoring: Fog and Edge Computing discussed various techniques involved for monitoring for edge and fog computing and its advantages in addition to a case study based on Healthcare monitoring system.

Avinash Kaur et al. elaborated a comprehensive view of existing data placement schemes proposed in literature for cloud computing. Further, it classified data placement schemes based on their assess capabilities and objectives and in addition to this comparison of data placement schemes.

Parminder Singh et al. presented a comprehensive review of Auto-Scaling techniques of web applications in cloud computing. The complete taxonomy of the reviewed articles is done on varied parameters like auto-scaling, approach, resources, monitoring tool, experiment, workload and metric, etc.

Simar Preet Singh et al. in the paper titled Dynamic Task Scheduling using Balanced VM Allocation Policy for Fog Computing Platform proposed a novel scheme to improve the user contentment by improving the cost to operation length ratio, reducing the customer churn, and boosting the operational revenue. The proposed scheme is learnt to reduce the queue size by effectively allocating the resources, which resulted in the form of quicker completion of user workflows. The proposed method results are evaluated against the state-of-the-art scene with non-power aware based task scheduling mechanism. The results were analyzed using parameters– energy, SLA infringement and workflow execution delay. The performance of the proposed schema was analyzed in various experiments particularly designed to analyze various aspects for workflow processing on given fog resources. The LRR (35.85 kWh) model has been found most efficient on the basis of average energy consumption in comparison to the LR (34.86 kWh), THR (41.97 kWh), MAD (45.73 kWh) and IQR (47.87 kWh). The LRR model has been also observed as the leader when compared on the basis of number of VM migrations. The LRR (2520 VMs) has been observed as best contender on the basis of mean of number of VM

migrations in comparison with LR (2555 VMs), THR (4769 VMs), MAD (5138 VMs) and IQR (5352 VMs).

Anand Nayyar, Duy Tan University, Danang, Vietnam
Rudra Rameshwar, Thapar Institute of Engineering and Technology, Patiala, Punjab, India
Pijush Kanti Dutta Pramanik, National Institute of Technology, Durgapur, India

DOI 10.12694/scpe.v20i2.1504
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 191–205. http://www.scpe.org © 2019 SCPE

EDGE AND FOG COMPUTING IN HEALTHCARE – A REVIEW

SUJATA DASH[*], SITANATH BISWAS[†], DEBAJIT BANERJEE[‡], AND ATTA-UR-RAHMAN[§]

Abstract. The architectural framework of Fog and edge computing reveals that the network components which lie between the cloud and devices computes application oriented operations. In this paper, an in-depth review of fog and mist computing in the area of health care informatics is analyzed, classified, and discussed various applications cited in the literature. For that purpose, applications are classified into different categories and a list of application-oriented tasks that can be handled by fog and edge computing are enlisted. It is further added that on which layer of the network system such fog and edge computing tasks can be computed and trade-offs with respect to requirements relevant to healthcare are provided. The review undertaken in this paper focuses on three important areas: firstly, the enormous amount of computing tasks of healthcare system can take mileage of these two computing principles; secondly, the limitation of wireless devices can be overcome by having higher network tiers which can execute tasks and aggregate the data; and thirdly, privacy concerns and dependability prevent computation tasks to completely move to the cloud. Another area which has been considered in the study is how Edge and Fog computing can make the security algorithms more efficient. The findings of the study provide evidence of the need for a logical and consistent approach towards fog and mist computing in healthcare system.

Key words: Fog computing, edge computing, healthcare system

AMS subject classifications. 68M14, 92C42

1. introduction.

The up-coming developments and research in the technology has adjusted the epicenter of the medical sciences not only for analysis but rather how to prevent those diseases with incomplete recognition and appropriate along with precise health information through a leading in demand technology called the "Internet of Things".

In order to record physiological symptoms and signals, various efficient devices have been discovered and are feasible in the market which can be easily connected to the internet over smart phones, computer or any nodal devices. There is a high potential appeal of the fog computing inside the IoT-based monitoring systems as per the contemporary surveys. The Internet of Things is pointed out towards the physical objects like appliances, electronics, devices which are always developing throughout the world that can feature internet connectivity within an IP address using unique identities are interconnected for enabling the various objects to interact and act accordingly. For obtaining various essential signs within real time, efficient and working in active way occurring in medical devices which includes ECG or for maybe for measuring pressure and temperature, WBAN is used as among various basic prototypes in IoT methodology of healthcare. WBAN establishes a multitude imperceptible or attire sensor node to percept and convey the data over a wireless network. The transmission protocols used for this are Wi-Fi or IEEE.802.15.4. Basically WBAN-based structure is Low-cost and power competent and plays a leading role in several region of healthcare environments such as tracking, and care clinic towards various disease management including chronic and taken precaution towards it. Distant cloud servers are used in many health monitoring systems for keeping and organizing huge data possessed as of sensor nodes in huge quantity. Cloud computing has a number of benefits like services which are made cheap, capacity for the huge facts storing capability and more surplus preservation rate but few challenges also exists like large data transmission, location perception and latency-sensitive issues. For the slowdown in data transmission and data error packet dropping possibilities have increased. More the data is sent over a grid system, the greater is the chance of mistakes that may result in wrong or less precise treatment decisions for which there will be adverse effect on emergency situation and critical illness of mankind. Thus, there will be a great requirement of minimizing the flow of data absorbed system in excess with the value of provision. One solution for accomplishing the problem of a predictable gateway and a remote cloud server is by adding a coat in between them. This added coating is termed as fog sheet and it aids in lowering the large chunks data but also yet guarantying service quality and so protecting the bandwidth of the network by pre-analysing the

[*]North Orissa University, Baripada, India (sujata238dash@gmail.com)
[†]North Orissa University, Baripada, India (sitanathbiswas2006@gmail.com)
[‡]KIIT University, Orissa, India (debajit2409@gmail.com)
[§]Imam Abdulrahman Bin Faisal University, Saudi Arabia (aaurrahman@iau.edu.sa)

information. Also, fog computing provides services of higher level at the edge of the wireless systems well as reduces the burden of the cloud [1]. Although the cloud computing hypothesis is brought by cloud computing at the edge of the network, it also reports which aren't proved before unfit in the cloud computing paradigm. Some of the fundamental characteristics of fog computing are low latency, geographical distribution, edge localization, location awareness, inter operability, as well assist aimed at on-line analytic. That is why Fog computing can be taken into consideration for advancing the human health observing WBAN-based systems for the feature like low energy, little bandwidth, truncated processing power and involving the hardware with constrained nodes. To say in literal words, the arrangement of WBAN-based system, with cloud computing and also fog computing can be a defensible clarification aimed at summoning in the recent IoT healthcare systems. As a concept,effective IoT-enabled healthcare system constructions presented and can be profited starting the idea of fog and edge computing. Utilizing the arrangement, we all came-up with an effective fog computing and edge computing healthcare systems which will be IoT enabled and can be utilized in bandwidth also with QoS guarantee, and backup reminder. This paper primarily focuses on the following three important aspects:

- The requirements of IoT based Healthcare model and the description of services provided by Fog computing layer to address them.
- The architecture of an IoT based healthcare system embedding Fog computing layer.
- Implementation of the fog computing layer services along with performance and advantages.

2. Related work. There have already been many attempts in creating brilliant access towards healthcare function.Like for example, Chen et al. [18] introduced a technology of wireless sensor network flow which is used for a smart gateway aimed at the health care system. The endorsed gateway operating for instance can over pass amongst public communication system and a wireless sensor network. It basically has some exceptional features like having a data with agreement system, very flimsy database plus the usefulness just before declaring the needy persons in case of any tough times. In addition, the gateway generates the request- response message function which creates a process of shrinking a distant server's fatigue. Mohapatra et al. proposed semi-hybrid architecture by using a sensor cloud which is needed in distant patient monitoring with working network flow [19]. Advantages were taken by utilizing the sensor cloud for patient's health condition which can be monitored and can be shown in their proposed system. The writers did present a cloud computing with a solution intended for patient's data collection in healthcare organizations. Sensors are used in various systems which are adhered to medical equipment's for assembling patient data then sent the data to cloud to provide restricted permit. Yang et al. presented health monitoring gateway system based on Smart phones [21]. The recommended requirements for a gateway in a Bluetooth console to upload gathered data and send them to remote servers. In [22], the sensing servers are used as gateways in the system which is handled by the sensor network system. But, the proposal is exaggerated the insufficient and extensible as well as inefficient for many applications. In many organizations [23], authors have researched and proposed an idea with an example of a lively IPv6. Less powered wireless with area network (6LoWPAN) is a signal router which was established on Hidden Markov Model. This model was used for developing settlement in the health status. In [24], the researchers have already designed a mobile gateway for pervasive healthcare system used for ZigBee and Bluetooth. The gateway which favours the feebleness with many services such as investigation of the medical data etc. are efficient, though the gateway can be ineffective within terms of actual power utilization which cannot be recommended for practical use. Zhong et al. presented an alternative way centered on a mobile phone which is used to connect sensor network system nodes along with the devices promoting CDMA or Bluetooth [25]. In a relative work [26], the recommended design can achieve data through many individual health gadgets. USB and ZigBee along with Bluetooth can be considered as a communication process. Subject including body area sensor network systems can be used to attract the attention among a lot of researchers during the past few years. Specifically, in the healthcare domain numerous amount of work has exponentially expanded various undiscovered stepping stones with only one intention for betterment of healthcare monitoring system.

As few works seek the newer systems along with further methods besides number of services although some other researchers approached to suggest new way or else new approaches. Still, some narration in the above-explained examples, a great quantity of systems are specifying along with ZigBee though it can be tough for guarantying the characteristics of service done in ZigBee while observing streamed bio-signals like ECG, EMG etc, with the extreme data rate nearly at 250 kbps of ZigBee. Oppositely, Bluetooth Technology can be used to solve problems of ZigBee which is lesser data speed and other small-range communication protocol. In spite of this, maybe it is tough for depicting a gateway for supporting mobility as well as employing data through multiple targets with utilizing Bluetooth Technology. The above systems fundamentally use simple gateways from node so collect data and transmit these data for remote servers. Further, sometimes any of these workings had been accounted completely taking advantages of the computing paradigm about fog and contributing insightful to the gateways. The target of the section is used to uplift the IoT-based health monitoring system utilized among different surroundings like among family also in hospital just by imparting the smart gateway [27]along with fog layer consists an extra advanced services. Fog computing conception is actually the elaboration of pointing applications from the cloud computing paradigm along with a vision and province where the full support of the prototype from the cloud is not achieved. Few among them includes fast response employed containing video conference applications, besides Voice over Internet Protocol (VoIP) which requires very less latency for the reason that QoS might get reduced by unusual delays.

- An Enormous data solicitation which is assembled in a huge load with data taken from countless sensors after which it transmits the data via the networks essential for the availability to achieve high bandwidth [28].
- Observing the applications, which can be controlled constantly, needs no interruption during data connection because of connectivity loss takes place among the cloud then monitoring systems [29]. Features use din the presentations are mentioned above which were indistinguishable along with the features of health monitoring systems in real-time. These network that have massive quantity of collected data commencing a variety of environmental along with bio sensors. Later this massive volume of data transmits above the network which is ultimately monitored remotely through end users like care-taker or doctors. Hence, fog computing stands on these systems effectively and is appraisable that instead of removing before lessening the importance of the cloud in IoT applications, from the facet of location awareness,less latency, scalability, real-time interactions, heterogeneity and inter operability that fog computing technique is completely cooperated as well as compatible by the cloud to enhance the existing IoT applications, rather than replacing or lessening the importance of the cloud in IoT applications.[30].

3. Fog Computing.

3. Fog Computing. Fog computing, networking, storage and other domain with particular services are handed over to the IOT system by Fog computing layer. The Healthcare domain differentiates it from the other Iot applications by having the most used feature of remote monitoring which require high degree of reliability. The protection and personal aspects towards the health care were highly essential and should be implemented along with the Fog layer focussing on the health care. Moreover,with that it permits the Healthcare IOT to address the necessities of the proximity of Body Area Network (BAN) to Fog layer. Some necessities along with the Healthcare IOT services were inclusive and useful for any application domain. The concluding view of the Fog layer and services are discussed separately and shown pictorially in Figure 3.1.

3.1. Managing Data. Data management is important in Fog computing because the sensory data is locally used in order for withdrawing essential information of anyone to provide a response to user and notify the system plan adaptations. Agreeing to the network architectonics, Fog layer dynamically accepts some huge quantity of data which is auricular from the sensor network system in a very small interval in time, so its main focus is to control the incoming data in order to give a fast reaction with respect to number of users as well as conditions required in the system. This task acquires more importance in healthcare scenarios since latency and uneasiness occurs in decision making might cause incurable damages to the patients. According to the numerous fulfilment of the data management work, we establish it with five different units and all are required for an intelligent e-health gateway. All these parts are named as local storage, data analysis, data filtering, data confining and data fusion.

3.1.1. Local Repository. The gateway requires storing the data which are received from many different sources in a local storage unit so that it can nonchalantly use for analysis in future. The operating system in the gateway contains a file server and a database which stores and also recovers the data. The gateway of the local storage stores the files which are encrypted or maybe contained in a compressed format depending on the type, size, along with the priority of the data. The gateway cause the local storage for exporting data to medical level information like Health Level Seven (HL7)[2][35] when needed. Other features of the gateway

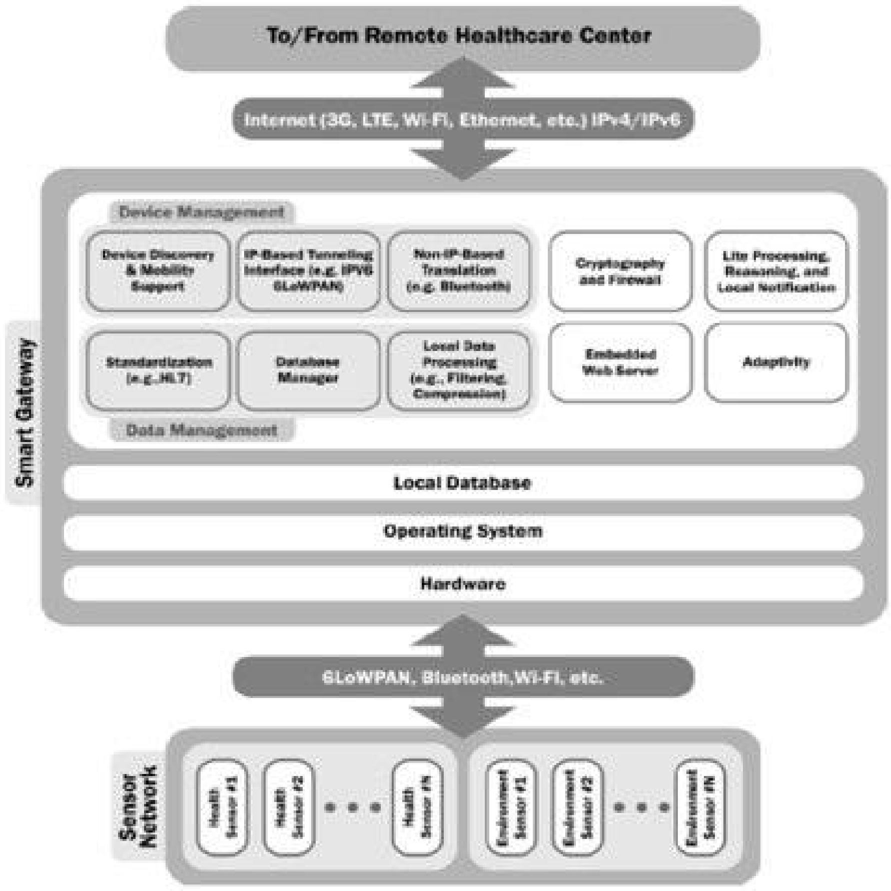

To/From Remote Healthcare Center
Internet (3G, LTE, Wi-Fi, Ethernet, etc.) IPv4/IPv6
Smart Gateway
Device Management
Device Discovery & Mobility Support
IP-Based Tunneling Interface (e.g. IPV6 6LoWPAN)
Non-IP-Based Translation (e.g. Bluetooth)
Cryptography and Firewall
Lite Processing, Reasoning, and Local Notification
Standardization (e.g.,HL7)
Database Manager
Local Data Processing (e.g., Filtering, Compression)
Embedded Web Server
Adaptivity
Data Management
Local Database
Operating System
Hardware
6LoWPAN, Bluetooth,Wi-Fi, etc.
Sensor Network
Health Sensor #1
Health Sensor #2
Health Sensor #N
Environment Sensor #1
Environment Sensor #2
Environment Sensor #N

FIG. 3.1. ***Different Fog layer and services.***

like analysing the data, compressing it,filtering, and inscription are basically inter-dependent on a local storage. The overall data rate can be moved among the cloud layer and also the Fog layer depending on the network pace then more towards local estimation along the gateway is visited throughout its power of processing. So it can occur in any displacements during transfer time along with computation time, the local storage acts as a local cache memory to make the data flow reliable. Local storage can also support and save the data when the internet connection isn't accessible among gateway and cloud server. When the internet is restored, the gateways can send saved data into the cloud.

3.1.2. Data Filtration. This is the first step of data processing unit that implements data filtering methods at the edge after receiving it from the sensor grid system. To get a patient's medical condition, numerous bio-signals are collected using pertinent fact-finding from devices like electrocardiogram (ECG), electromyography (EMG) and photo plethysmogram (PPG). These bio-signals relatively include difficult shapes along with tiny amplitude (i.e., millivolts) and accordingly they were dominated by specific unwanted an unreliable sounds which is misled throughout the health monitoring. The noises that are perceived can be thermal noise, electromagnetic interference noise etc. Some available light-weight filters at the nodes of the sensors reduce the assembled noises yet; this may effect in some practical aspects. Thus, the Fog layer containing the data filtering unit which removes the noise and can boost some features of the signals (e.g., signal-to-noise ratio) by the use of different filters (e.g., finite impulse response (FIR) filter) earlier than any other local data analysis.

3.1.3. Compressing Data. For decreasing a massive quantity of transferred data throughout the communication network, data could be flattened by lossless or lossy compression methods used. The lost data can generate inapt or incomplete disease diagnosis in healthcare IoT applications because lossless compression is preferred. Even if compression algorithms which are lossless can retrieve indigenous data with precision, are considered to be complex. Also, the algorithms are not most suited for sensor nodes which are actually resource-constrained concerning computation, memory capacity also battery capacity. Like for an instance, in all sorts of sensors lossless ECG compression methods [3, 4, and 5] cannot be used. In particular cases, at sensor nodes these lossless algorithms can be fortunately employed within it but these can also originate huge power utilization and latency. Fog computing aids to improve these defaults of the sensors and can be kept away by diverting all toughes of the sensor nodes towards the gateway and can manipulate these complications effectively during considerations of the real-time circumstances.

3.1.4. Data Blending. It is a data processing unit which can place sensory data commencing the numerous sources for getting more meaningful and strong information. By avoiding repetitive data and restoring the latest data this processing unit can completely diminish the volume of sensory facts.The local data analysis and data transmission gets boosted by the data reduction towards the distant servers.Three separate classes are categorised in Data fusions interdependent concerted also competitive with others [6]. Firstly, At least two different details acquired for more extensive knowledge of the Fog layer achieved from dissimilar places when it mixes with data fusion. For instance, when an ecosystem context data mixes with patient's health specification it shows a more complete data about the patient's state. Secondly,by integrating data which is formulated from one source with (at least) two other sensors the fierce data binding can regulate the quality of the data along with system's decision making capability. For example, utilizing the values taken from an ECG signal also from a respiration signal further powerful heart rate value is extracted. Lastly, using multiple sensors latest information can be yielded with relevant data fusion from a particular place. An example of a cooperative data fusion is the patient's medical state which uses vital signs (e.g., heart rate and respiration rate).

3.1.5. Analysing Data. In data analysing unit the sensory local data is carried out by the healthcare system. The quality of the system in this unit and then sent to the cloud servers after diminishing the response latency and data transmission. For example, when an emergency occurs like patient's health deprivation the emergency matter gets triggered and response comes to play quickly and appropriately because in the cloud instead of getting imparted the local processed data waits for a suitable response. Furthermore, data reliability as well as the consistency of the system is enhanced by the data analysis unit.Finite bandwidth along with Connectivity losses during data transportation in the long-term remote health monitoring were like an inevitable event because the patient may engage in different activities in assorted habitat. Hence,the system's local functionality,sensory data as well as the arithmetic can be directed and taken care by the confined edge of the storage data analysis and also balance the Fog layer after recounting it to the network with a remote cloud.

3.2. Managing Event. A number of major events like change in vital signs, activities or surroundings of the patient can take place in patient's audit. In actualize systems with a specified gateway or maybe a shift in a learned behaviour can be activated by these events. In case of critical condition low latency clarification which might help in notifying health experts, caretaker and even patients are supplied by Fog computing technique immediately.The event administration service fortifies on time and with proper signal delivery in case of medical actions or automatic system actuation when there is a need in quick response. In some medical events like changing the frequency of nerves stimulation related to heart rate or automatic insulin pump with blood glucose level which is regulated with the quick response of activators when it is essential. Some requirements are alerted to the rapid responding team, caretakers or family members of the patient when there is an incident occurred.

3.3. Resource Effectiveness. Resource proficiency is among the most important factors in healthcare IoT function, since catastrophe can lead to severe corollary for sensor nodes' failure to incorrect diagnosis among diseases in resource management. Expectations of collected data presented at end-user's terminals essential to be constantly kept a note of with the energy exhaustion of sensor nodes. They are dealt with in details as following:

3.3.1. Efficient use of energy of the nodes. Sensor nodes in health observing systems should be capable to work for a long interval like a whole day or even a few days which are generally small and resourceinhibited, such as miniature battery capacity. Sensor nodes should be functioning efficiently in relation to energy utilization for fulfilling these necessities. For the fulfilment of these tasks many methods contain software and hardware-based procedures which are useful. For example,the design of sensor node hardware is used for specific purposes instead of generic tasks. The unnecessary mechanisms or high-power overwhelming components can be avoided by controlling the energy depletion by this technique. However, customizing the software successively at the nodes is more stimulating for scheming energy efficient nodes. Specifically, the software is simple in order to reduce reckoning time and must be able to execute primary errands of the sensor nodes. For example, sensor nodes in IoT-based plunge recognition systems could intricate the fall detection algorithms by only collecting digital data and address the data to Fog layer [7]. Dramatically condensation can occur in the sensor nodes' power depletion.

3.3.2. Latency. Latency produces unsuitable disease enquiry and delays the supervisory procedure in health monitoring solicitations. Handling and conveying of data both presented latency in IoT for transmitting latency to end-users from sensor nodes via gateways, like Cloud and processing access. In multiple instances,trade-off relationship is often dealing out and showing latency expectancy. However,reduction of the total latency is not always occurred buttes ting the handling data latency. Data processing can even surge the total latency in some circumstances. For some certain sensor nodes and applications a pragmatic differentiable procedure must be done to minimize the total latency and fulfil the time necessities of real-time health monitoring. In other cases, to remove noise and void data some cleaning methods are used in reducing the latency in total as well as huge amount of data transmission. For example,Fog can act as an alternative for sensor nodes to reduce the total latency in an ECG feature withdrawal solicitation; the feature extraction algorithm must be routed towards it [9].

3.4. Management of System. Device supervising consist many parts of Internet of Things organization. In this part, the main area of the debate is on device supervising from the fact of observation of locating device and keeping the connection through in mobility.

3.4.1. Mobility and locating. In the sensor and actuator systems the reserve restriction of devices had been stated previously. The life time of the battery is very important specifically fora battery motorized devices which also require actual supervision. Devices must be in sleep mode in a supervised manner whenever they are about to gain idle condition. Any transmission that takes place when the device is in the sleep condition requires care for the cover of Fog.The patient,who is trying medical sensors for moving from one location to other in a healthcare scenario, can adapt the liable gateway that handles transmission. This shows that the sleeping sensors need to wake up at the sector of various gateways. To locate and effectively handle the sleeping cycle device discovery service must support other devices that can link to a sleeping sensor node; the full utilization can be found for this service [10]. The Fog layer service is used to communicate the nearby kept data from one gateway to other and let it occur without riveting much of the finite resources but also should afford the liberty of flexibility for the patient. For smoothly pursuing the observation activity at the new place in other gateway the device must be located in a new situation, the bestowal takes place. A shortened execution of flexibility and locating device functioning simultaneously is represented in Fog computing [11], These services can be used for more provinces as well as these facilities too.

3.4.2. Inter-operability. The IoT is generally a bunch of diverse set of transmission protocols, platforms and data setups. There are multiple regularity try to initiate constantly amongst the various modules. The present upright subdivisions of presentations require to be joined to simplify the modeling of not just a part of but the entire of the healthcare applications. Traditionally affinity is facing the dare related to resource restriction of the most of the end gadgets. The presence of fog figuring layer is to cover end devices and thus prosper inter operability which caters a vital part in giving amenities gripping. These utilities actually work as a connector among various transmission protocols, data setups plus platforms. At the Fog layer some work can be experienced forgiving semantic inter operability to the data obtained via the sensor son the cloud.

3.5. Personification. Various presentations of Fog computing can be setup in trial or through in run time by behaviour of a machine. This might not bead equate for healthcare structure since users may have hold in various triggers through assorted surroundings and innumerable medical state. So, a powerful plan for the machine is necessary, not only to customize the machine performance in granting to specific user requirements but also assertive habituation to the machine over time, exceptionally in case of a crisis. In this regard, it optimizes the energy proficiency and also enhances the local decision making. Various health applications using rule-based skills and machine learning processes are illustrated by Customized scheme behaviour. According to the patient's condition, the proper principles are chosen from the sensor sampling frequency and data transmission rate which are well described. Further, precedence is customized with respect to the medical history of the patient. Simply putting together, the machine would acquire to give more proper heart associated frameworks only if a heart failure is diagnosed for a patient throughout the observation.

3.6. Privacy and Security. Overall, safekeeping is very vital for every solicitation in healthcare is important because a single apprehensive part of a machine might endanger a human life. Fore.g, an insulin pump in an IoT glucose controlling machine can be hacked within 100 feet which is reason for harm [11]. The full machine which consists of sensor nodes, gateways, Fog and Cloud must be protected adequately for delivering a secure IoT healthcare machine. The complete system can easily be manipulated by hackers if somehow any components are hacked. For facts encryption and decryption various like CMAC-AES-128 or AES-128 are used at sensor nodes and gateways, respectively. For constructing IP tables giving consent to proper transmission ports IP table provided by Linux can be utilized in the gateways [12]. For securing the whole machine these techniques can't be trusted as powerful one because these can modify safety at different levels, whereas, in many occasions, it provides a high level of security in the literature [13, 14, and 15]. Still, their complicated cryptographic procedures are not concern to resource-constrained sensor nodes as well as IoT machines. Rahimi et al. [16, 17] currently launched end-to-end secure parameters for directing the safekeeping troubles in IoT healthcare machines. The chief parts, comprising of various complicated security algorithms, are executed at the Fog layer as well as the parameters that can give better verification and approval for Health-Internet of Things machine.

4. Edge Computing. Edge computing says about an astonishing eyes opening technology that allow calculating at the extremity of the linkage, on data of downstream or upstream kind. Downstream will come from cloud services where-as upstream data is organized on the part of IoT services. And now we declare "edge" as figuring and integrating grid system locality along the way among data obtain ability and cloud data servers. Like, a smart phone is the border or edge connecting living beings and cloud data, a gateway can be referred to a smart home. The hypothesis of edge computing is that computation should always take place near a data source. According to our idea of research, fog computing can be replaced by edge computing [32] but fog computing marks extra on the substructure side, whereas edge computing stresses mainly on the effective side. We envisage that edge computing when compared with the cloud computing can possess a large and better effect on our community. Fig. 4.1 displays the two-way computing rush for edge computing. Edge computing models, things that acts as data consumers, but can also act as data producers. On the edge, representing the computing works which is commenced on the cloud but also the possessions,can't appeal facility and content from the cloud. Edge computing can conduct data transfer, storing the data, accumulating and handling, as well as spread appeal and delivering amenity from cloud to user. The edge itself requires to be well planned to converge the need that can be effective with praise to reliability, security, and isolation protection with all the kinds of tasks in the network.

In edge computing, the center of attention is on setting the computing attached to the closeness of data causes. This has various advantages contrast to conventional cloud-form computing paradigm. To illustrate the probable advantage we use several premature outcomes from the circle. The response time has moved the computation from cloud to the edge by reducing it from 900 to 169 Ms and the Research workers raised a proof-of-concept platform to execute face acknowledgment presentation [64]. Ha et al. [33] used the cloudlets for offloading the computing tasks for wearable cognitive support, and the advancement of reaction time is between 80 and 200ms when the results get displayed. Furthermore, by use of cloudlet offloading the energy utilization could also be condensed by 30%-40%. Between the mobile and the cloud with their prototype which could shrink 20 times the executing time and energy for tested uses can perform merge dividing, relocation with merging, and the on-demand instantiation in the clone cloud.

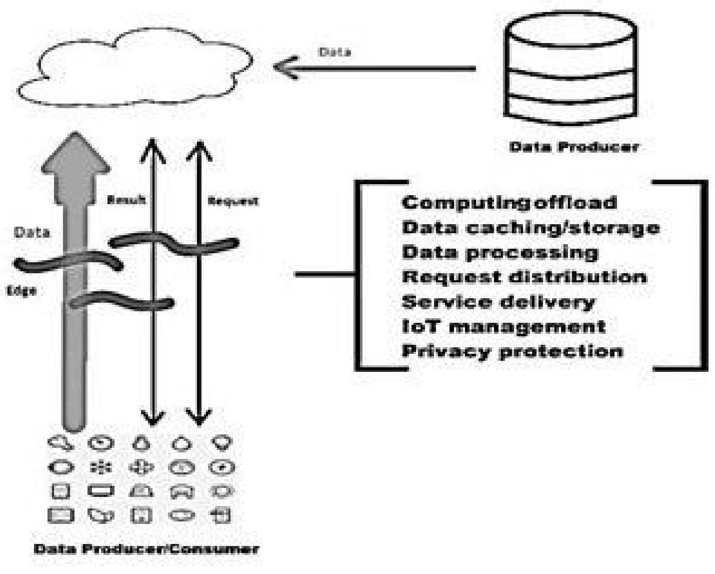


FIG. 4.1. ***Edge computing architecture.***

5. System Architecture, Fog and Edge computing. Figure 5.1 exhibits a complete diagram of a Health-IoT system that shows how the elements can be systemized in such a scattered way over the three layers which can be used in smart hospitals. Among these classifications, the implanted sensors note the associated material of the patient's health, thus the private monitoring of various variables can be furnished by helping the patient. This health data can be also boosted by means of date, time, location, temperature, etc. Circumstance-awareness allows identifying a typical design and creates more exact inferences regarding the situations. The staff like CAT scan, magnetic resonance imaging can be attached with other medical equipment for transferring information from machine. The machine construction contains following important elements:

- Medical Sensors and Actuators networks: Some omnipresent identification, sensing, and transmission capability, biomedical and perspective signals are allowed to betaken from the body and surrounding. Some wireless or wired communication protocols such as Bluetooth, Wi-Fi, networks of intelligent e-Health helps in transferring the data to the access.
- Gateways: After several geographically scattered intelligent e-Health gateways this layer is constructed, such as developing a virtual fog. Every gateway, acts as a connection point among a sensor network along with the local switch/ Internet which helps in different transmission protocols. These higher-level facilities like data aggregation, filtering and dimensionality depletion can only be accepted if data from various subnetworks, bearings protocol conversion provided to them.
- Back-End System: A cloud computing programme that executes streaming, data warehouse as well as data analytic is achieved in this Back-end scheme. At last, it can adjunct the web client exhibition such as GUI regarding final picture and feedback. A source of large data [36, 37, 63] for statistical and epidemiological detecting forthcoming epidemic diseases can be constituted by the possessed health and setting data.

5.1. Properties in addition to features of the smart e-Health gateways. To help several wireless proprieties and take charge of inter-device transmission are the most significant part of this gateway. According to this part, we enlarge its particular part which turns out to be fog enabler by (i) establishing the orchestra by the linkage of gateways and (ii) applying various characteristics like acting as repository to volatile store users' data with sensors' and combining them with aggregation,interpretation techniques, and data fusion. For the local pre-processing of sensors' data these techniques are essential and can be called as an intelligent e-Health Gateway.

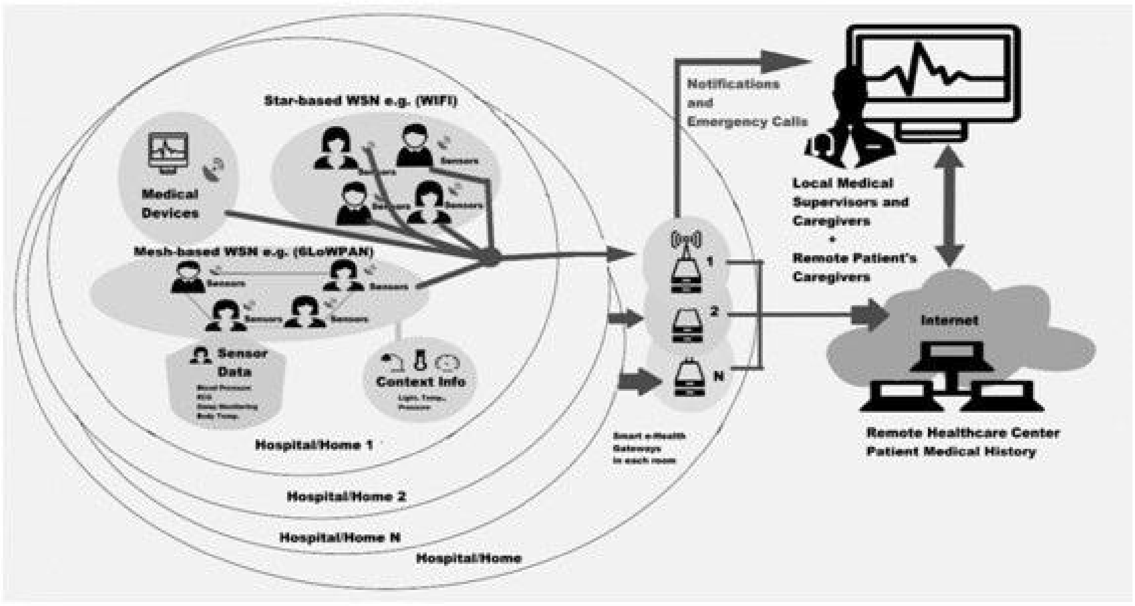

Star-based WSN e.g. (WIFI)
Medical Devices
Sensors
Mesh-based WSN e.g. (6LoWPAN)
Sensor Data
Context Info
Light, Temp., Pressure
Hospital/Home 1
Hospital/Home 2
Hospital/Home N
Hospital/Home
Notifications and Emergency Calls
Local Medical Supervisors and Caregivers + Remote Patient's Caregivers
1
2
N
Smart e-Health Gateways in each room
Internet
Remote Healthcare Center Patient Medical History

FIG. 5.1. ***Smart e-health monitoring system***

5.1.1. Local data processing. Between the several characterised fogs computing has further added edge of local data handling which is executed to give smartness by which the streaming data is surveyed locally at the gateway. Agreeing towards the architecture of the machine, fog layer requirements constant approach of a huge quantity of sensory data in less time and react properly in surrounding to several situations. This task develops to be more vital in medical cases by allowing the machine to response as quickly as it can in case of medical crisis [38, 61]. Figure 5.2 explains the usage of a local processing unit for data filtering, data compression, data fusion, and data analysis with a conceptual architecture of an intelligent gateway.

5.1.2. Data filtering. Accepting data,after number of sensing devices and making it important to use it properly, early bird processes at the edge previous doing some kind of high-tech processing example data survey is really important. Bio-signals sucked up from any patient are always major sources of data for evaluating a patient condition. These generally have complex figures with little voltages and lots-off frequencies. Throughout the time of gathering information from a patient body, sound is also collected to the bio-signals and twists the indication standard. Such disturbances in the electromagnetic interference from other electrical machines, electric power grid, and unsuitable extension of sensors to patient's body can be built by several resources like oscillations of ac. These problems can be directed by the smart e-Health gateway at the fog layer in which it interfaces sensors at straight. By the use of several transmission protocols the fog layer accepts digitalised signals from sensors. Even though sensors might be executing light-pressures training to eradicate disturbances to a specific standard at the data receiving unit, additional tough and complicated information filtering at the fog layer is of primary importance.

5.1.3. Data compression. To minimize transmission waiting and power disbursed throughout transaction; the factors of data transmission, data compression is very essential. Both lossy and lossless data subjected on the solicitation are squeezed broadly utilizing it in the IoT domain. Computation for executing composite algorithms more demand than the Lossless data compressed for its short coming. So, for lossless data compression methods are advanced which have the ability and necessities for a processor speed and memory size. Both lossy and lossless compression technique are appropriate in the system layer of Health-IoT machine. Although, some restrictions occurs just as battery life time and availability of treating energy in some cases where lossy data compressing technique is more suitable for resource-constrained sensors. For example, numerous standard lossless ECG looseness techniques [39-40] don't suitable to many different kind of sensors like in lossy com-

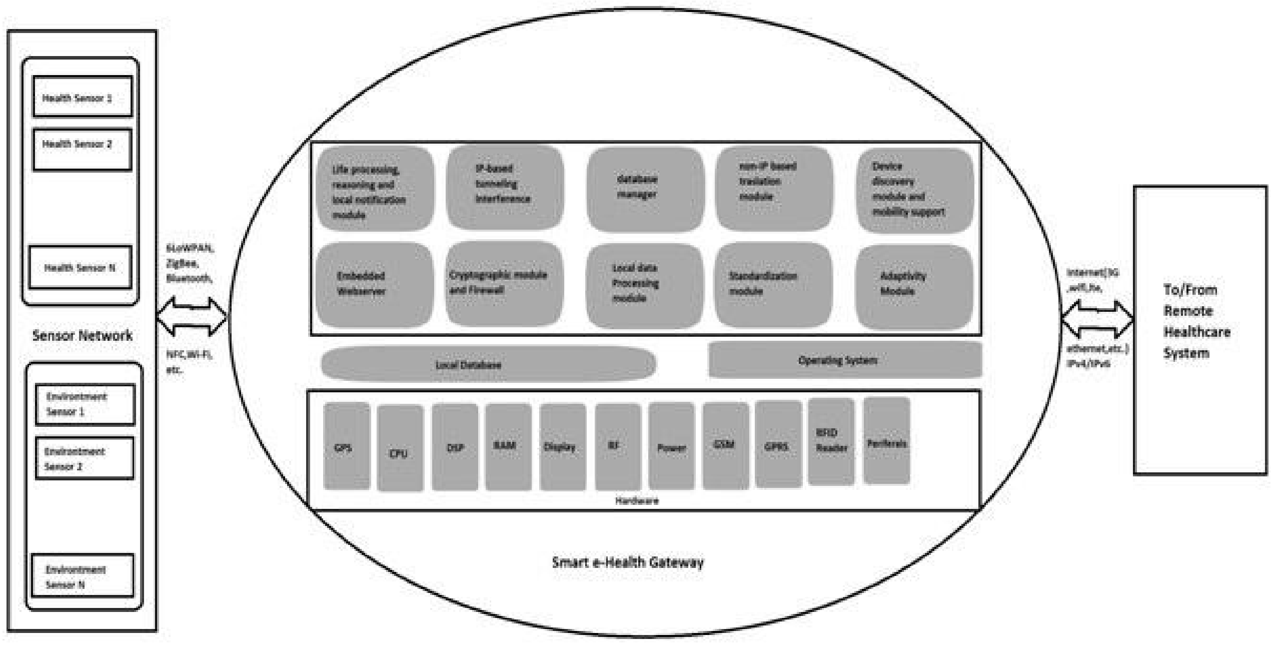


FIG. 5.2. ***Gateway architecture of smart e-health system.***

pression techniques, for instance the technique proposed by Yu et al. [41], were practical in hardware demand with its features. For checking all structures of the signals are remained ensure of with big accuracy which is presented as in real-time ECG monitoring, has an advantage to have lossless compression. By discharging the liability from system layer Fog computing supply the required computational energy for systematic directed towards lossless data compression algorithms with complexities. While consuming lossless data compressing techniques it can authorize real-time actions.

5.1.4. Data blending. Data blending authorizes to productively lower the capacity of data, and there by lower the energy necessary for transmitting data. Whereas, Data fusion is actually divided in three different parts: cooperative,complementary as well as competitive [42]. Corresponding data fusion is applied on the fog layer to obtain stronger worldwide knowledge. Gathering the temperature dissimilarity between body and the neighbouring at any occasion is given from two different sensors. Competitive data fusion is also employed on the edge that data since a distinguished factor is gathered from various resources to refine the precision and the stability of outcomes in instances of sensors' collapses. Ultimately, collaborative data fusion too gives advantages in the edge such that fresh data is gathered in intelligent gateways from the diverse information which is gathered from different origins. For example, cooperative data fusion provides all-inclusive information regarding the medical condition of a patient from their important symptoms.

5.1.5. Data analysis. The reactivity of the machine can be enhanced by implementing localized data examination in the edge. It may help the machine to erase and forecast alarming conditions. For example, detecting the fall of an aged person, the fog layer could provincially supply detection of falling of a person regards giving out not only despatching variables to a cloud and resting for the replies. Subsequently, the machine responds to the alarming condition as fast as possible and is more trust worthy and executes real-time reactions. In the fog layer,Apart from that the sensitivity of the machine, exploiting facts examination to allow the machine on the way to lower the latencies along with crucial variables to a maximum limit. Moreover, localized data examination and local response from the sensory information upgrade the machine dependability and stability in case of non-avail ability of Internet connectivity. For continuing distant observing of discrete suffering from persist entitles, Internet disconnection can be recurrent happenings. Fog computing authorizes to remain all the performance of the machine function close by. Additionally,giving out findings in a localized repository in the fog layer as well as feasible to rescue the sensory information after harmonizing them along with the Cloud and connecting it later when it is back in position and in operation.

5.1.6. Adaptability. Taking into account, the implementation of fog layer in different situation, some early defined variables (example:sensors to cloud transmission rate) are fixed considering the use case. Nevertheless, this is very important for the fog layer to be customizable as well as adaptable over time, exceptionally when crucial incident happens. The customization may be vigorously implemented for different assistances employing progressive ML algorithms like incremental SVM and incremental NN. Data transmission of the fog layer requires being regularly incremented, which combines data appeal from the sensors and also transmitting the data taking it as a ratio of the cloud to the fog layer. For example, on continuing observation of a sick person sustaining cardiovascular disease, the machine must acquire to grow the appeal rates (priority) for heart-related variables on identifying unusual heart-related issues. Nevertheless, rate of transmission to the cloud is executed by allocating prime concerns to various services besides parameters. The prime concerned of rate of the cloud for patients to the transmission of data with critical illness involves is being elevated, while low transmission rate is required for the patients suffering from chronic diseases. As a conclusion,the performance of machine is not good as the adaptive fog computing thereby growing sensitiveness with explicitness of crucial limitations and on dependability of the prime concern the system adapts.

5.1.7. Local storage. To safeguard that the machine is able to recuperate evenly, the data, gateways must be efficient of keeping the arriving data with the local drive. The OS on the gateway manages the local repository and also keeping the data inside of a non-volatile storage. Data can be kept in local drive in a squeezed or protected way depending on the category as well as consequences. Data can be transported into a medical quality formats like Health Level-7 (HL7) [43] if needed in the repository. For additional purposes of the gateway storing of data is correspondingly important. As considered before, access will be in charge of data survey, compressing, filtration as well as encoding; all these tasks may need a local volatile drive. The pace for transmitting data from the gateway to the cloud is restricted by network transmission capacity, besides computations were restricted by giving out capacity of the gateway, for the situation of inequality of data processing and data transmission, constant data drift along with the local drive will act as a cache to appliance uniformity. When the network is not available but it makes the Data drive on gateways machine dependable and powerful. The local repository is managed by the database administrative unit shown in Figure 5.2.

5.1.8. Local impulsion. The actuation can be categorized into different configurations in the IoT-based healthcare system. The actuation can be used in the configuration of data streaming, administrating medical actuators as well as sensor network reconfiguration. , A foreseeable and fast reaction time is proven in many situations. Illustrations for fast reaction actuators modify the frequency of electrical nerve which is restored depending on the heart rate, otherwise altering insulin liberate rate in unmanned pumps based on the additional key indications and patient blood Glucose. In real-time to regulate panel for medical exception lists is important for transmission to slow down where at least samples/second rate is required to attain when there is a streaming in a patient's medical signals. Utilizing the networking potential along with local processing force the gateways are capable of streaming real-time signals like ECG and PPG (photo plethysmogram) towards a user machine (i.e., tablet), on the Internet connectivity separately. For intelligent e-Health gateways at the edge of the network some statements are required between the notable functions. Health care observing systems frequently requires notifying and informing care givers, medical teams and the patient about censorious circumstances that need immediate response. Any miscarriage in the notification utility may cause critical issues for medical therapy as well as the patient. Differentiate of sending notifications to a cloud server which is capable after numerous practices, a gateway which has restricted measures only can inform through few fix media. However, there maybe a lack of cloud server but the gateway-based statements proceed autonomously (e.g., via the local grid system or GSM) throughout the server, after amplifying the constancy of the machine we must be sure that users can obtain important reports in time.

5.1.9. Security. In the Health-IoT systems security is regarded as of great importance on the morale that in secured machine may have dangerous problems. In Linux the gateways like the IP table is exploited when it is approached by a big level of security as well as operating system level. When Linux kernel firewall supplies IP tables and IP FW it is also applied for the set of regulations in the network packets configuring the IP packet table. At some ports IP tables are constructed for giving privileges to communicate while other ports will be obstructed meant for obstructs dispensable traffic [44, 60]. Although the gateway behaves as an embedded web server when the set-up is not available or when it is required, it may convey through secured validate sensor junctions plus HTTP sustaining the integrity,privacy, and originality of the machine. Even though IP tables give a few lead, they can' be reviewed as a powerful appliance for safety. In structured to have a towering level of safety, IP tables need to be in alliance with extra powerful safety methods. To label these problems, distinct advance towards have been currently suggested in the compositions [45-47, 59]. Although, cryptanalysis performance in these advancements are bulky because of the capability of force and processing capability limits building these illogical for resource-limited system. In a current report cantered on safety, Rahimi et al. [48, 49] proposed a safe and robust authenticated and authorized proposal for the Health-IoT machines which is required for few processing power in the edge. More accurately, on the conflict to run safe techniques at resource-limited sensor nodes, and the proposal utilizes features of an intelligent gateway in the fog computing for high end security which is related to the workload.

5.1.10. Inter operability and re-configurability. Apart from the uniformities attempts, it was apparent that inter operability has a crucial part for the accomplishment of Health-IoT tools. By alike heterogeneous mixture of grid system technologies, platform and protocol selection for executing IoT-based framework, implementing these approach is an apparent confrontation. This intelligent e-Health gateway has a crucial part forgiving inter operability for different sensors attached through distinguished grid system edges. As depicted in Figure 5.2, the intelligent e-Health Gateway utilising either wireless network or wired connections are attached to the context sensors plus the health sensors where it implements various grades (e.g., Wi-Fi, ZigBee, Bluetooth, 6LoWPAN,) to interact to the gateway. The cleverness of the gateway arises within the shape of uncomplicated combination of technologies and standards with these heterogeneous network protocols, soit permits them to interchange data along with effort easily.

Technical aspect of Inter operability. The different machine parts in the IoT-based machine are developed by number of sellers, and thereby they implement various grid system standards and interfaces. In this intelligent e-Health gateway development, technical inter operability can be gained by instructing switched data to the intelligent e-Health gateway that has various interfaces.

Syntactic Inter operability. The layer for Syntactic inter operability actually depends on the foregoing, technical inter operability layer. This handles the structure of messages interchanged across the systems.

Semantic Inter operability. To give semantic inter operability in two different types a proposed intelligent e-Health gateway had been developed. The first inter operability represents the information gathered from sensor nodes when it is concerned with additional systems. On the other hand,Internet plus the human readable data is attached with the gateway.

5.1.11. Discovering devices and mobility assistance. Usually mobility has two major techniques, handover and roaming, those are required for loss of data and interruption in services, and to manage the Quality of Service (QoS). In mobile host, bestowal happens when the connection channel is switched to other channel during roam occurs while going from one network to other. Macro and micro class are two divisions of Mobility and they were expressed as mobility inside the various network regions also inside the network region, commonly [48, 57]. There remain a number off ewer techniques which hold up mobility in edge routers [50, 56, 58]. Nevertheless, substitute technique is not present in this period to wholly represent the toughest the IoT scenario. For instance, inconsistency with routing which is multi hop, besides the need of NS (Neighbour Solicitation)/NA (Neighbour Advertisement) interchanges occur among the dualistic real problems inside the proxy IPv6 mobile. NEMO changes to further complex when various kinds of mobility (micro and macro) happen at a time.

5.1.12. Energy efficient criteria for sensor nodes. Recently, several approaches [51, 54, 55, 62] are being focused on giving energy efficient for IoT based Health products. When Gia et al. [53] was utilizing the short power transmission protocol, Otto et al. [52] executed the authentic signal handling on sensor nodes towards the sensor nodes conserving the energy of transmission. The appreciable quantity of energy can be easily stored or kept unused through fog computing by redistributing a few capacities from sensor module to intelligent gateways.However,those approaches may increase the energy efficiency of the sensor nodes.

6. Conclusion. The research work presented in this article focuses on retrospection of Edge and Fog computing applications in healthcare. These jargons are associated with Cloud computing and named according to their architectural relationship. Various applications can function more effectively using Edge and Fog level rather than migrating to the Cloud. This is for various factors like nature of client, data locality, grid system overhead, device and cloud resources and their availability etc. The main intention remains to make the service available in all types of circumstances. It has been found that various healthcare related applications are more applicable for execution in Edge and Fog rather than Cloud for its solution to the various constraints of the various sensors. However, the review depicts that a massive computation which happens in healthcare applications need to be processed in a layer that exist between the sensor network and the cloud. The articles reviewed here also show that all the network layers have potential to get involved in computation work. Apart from this, we also have highlighted the trade-off when allocating computational task to the levels of the network and discussed the challenges and security issues of fog and edge computing related to healthcare applications. Though the sensor devices are not powerful enough for performing the computation task, they overcome this limitation by offloading the computation task. In addition, due to some restrictions, cloud computing is not an efficient solution for such offloading. But, the flexibility of fog computing allows to induct computation as a part of the network infrastructure, therefore regarded as a most appropriate solution for healthcare.

REFERENCES

[1] J. Burt, "Fog computing aims to reduce processingburden of cloud systems," http://www.eweek.com/grid systeming/fog-computing-aims-to-reduce-processing-burden-of-cloud-systems, html, 2010.

[2] Health Level Seven Int'l. Introduction to HL7 Standards, 2012, www.hl7.org/implement/standards [accessed 2015-07-30].

[3] M.L. Hilton. Wavelet and wavelet packet compression of electrocardiograms. IEEE Trans.Biomed., 44(5):394 - 402, 1997.

[4] Z. Lu, D. Youn Kim, and W.A. Pearlman. Wavelet compression of ECG signals by the setpartitioning in hierarchical trees algorithm. IEEE Trans. Biomed., 47(7):849 - 856, 2000.

[5] R. Benzid, A. Messaoudi, and A. Boussaad, "Constrained ECG compression algorithmusingthe block-based discrete cosine transform. Digital Signal Processing", 18(1):56 - 64, 2008.

[6] H. F. Durrant-Whyte. Sensor models and multisensor integration. Int. J. Rob. Res., 7(6):97-113, 1988.

[7] T. Nguyen Gia, T. Igor, V.K. Sarker, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H.Tenhunen. Iot-based fall detection system with energyefficient sensor nodes. In IEEE Nordic Circuits and Systems Conference (NORCAS'16), 2016.

[8] T. Nguyen Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, K. Mankodiya, P. Liljeberg, and H.Tenhunen. Fog computing in body sensor grid systems: An energy efficient approach. In IEEE International Body Sensor Grid systems Conference (BSN'15),2015.

[9] T. Nguyen Gia, M. Jiang, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. Fog computing in healthcare internet of things: A case study on ecg feature extraction. In Proceeding of International Conference on Computer and Information Technology, pages 356-363,2015.

[10] B. Negash, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. Lisa:Lightweight internet of things service bus architecture. Procedia Computer Science, 52:436 - 443,2015. The 6th International Conference on Ambient Systems, Grid systems and Technologies (ANT-2015), the 5th International Conference on Sustainable EnergyInformation Technology (SEIT-2015).

[11] N. Paul, T. Kohno, and D. C. Klonoff. A review of the security of insulin pumpinfusionsystems. Journal of Diabetes Science and Technology, 5(6):1557-1562, 2011.

[12] netfilter/iptables - nftables project. http://netfilter.org/projects/nftables/ [accessed 2015-07-24].

[13] G. Kambourakiset, E. Klaoudatou, and S. Gritzalis. Securing Medical SensorEnvironments:TheCodeBlue Framework Case. In Proceeding of The Second International Conference on availability, Reliability and Security, pages 637-643, 2007.

[14] R. Chakravorty, A programmable Service Architecture for Mobile Medical Care, In Proceeding of Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, pages 5 – 536, 2006.

[15] J. Ko, J. H. Lim, Y. Chen, R. Musvaloiu-E, A. Terzis, G. M. Masson, T. Gao, W. Destler, L. Selavo, and R. P. Dutton. Medisn: Medical emergency detection in sensorgrid systems. ACMTrans. Embed. Comput. Syst., 10(1):11:1-11:29, 2010.

[16] S.R. Moosavi, T.N. Gia, A. Rahmani, E. Nigussie, S. Virtanen, H. Tenhunen, and J. Isoaho. SEA: A Secure and Efficient Authentication and Authorization Architecture for IoT-Based Healthcare Using Smart Gateways. In Proceeding of 6th International Conference on AmbientSystems, Grid systems and Technologies, page 452-459, 2015.

[17] S.R. Moosavi, T.N. Gia, E. Nigussie, A. Rahmani, S. Virtanen, H. Tenhunen, and J. Isoaho, Session Resumption-Based End-to-End Security for Healthcare Internet-of-Things, In Proceedingof IEEE International Conference on Computer and Information Technology, pages581-588,2015.

[18] Y. Chen, W. Shen, H. Huo, and Y. Xu, A smart gateway for health caresystem using wireless sensor grid system, in 2010 Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM), July 2010, pp. 545-550.

[19] S. Mohapatra and K. S. Rekha, Sensor-cloud: a hybrid framework forremote patient monitoring, Int. J. Comput. Appl,

vol. 55, no. 2, pp.1-11, 2012.
[20] C. O. Rolim, F. L. Koch, C. B. Westphall, J. Werner, A. Fracalossi, and G. S.Salvador, A cloud computing solution for patient's datacollection in health careinstitutions, in ETELEMED'10, Second International Conference on e-Health, Telemedicine, and Social Medicine,2010. IEEE, 2010, pp. 95-99.
[21] S. Yang and M. Gerla, Personal gateway in mobile health monitoring, in 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), March 2011, pp. 636-641.
[22] J. W. Yoon, Y. K. Ku, C. S. Nam, and D. R. Shin, Sensor grid systemmiddlewarefor distributed and heterogeneous environments, in NISS'09.International Conference on New Trends in Information and ServiceScience, 2009. IEEE, 2009, pp. 979-982.
[23] W. Shen, Y. Xu, D. Xie, T. Zhang, and A. Johansson, Smart borderroutersfore-healthcare wireless sensor grid systems, in 2011 7th International Conference on Wireless Communications, Grid system and MobileComputing (WiCOM), Sept 2011, pp. 1-4.
[24] T. H. Laine, C. Lee, and H. Suk, Mobile gateway for ubiquitous healthcare system using zigbee and Bluetooth, in 2014 Eighth InternationalConference on Innovative Mobile and Internet Services in UbiquitousComputing (IMIS). IEEE, 2014, pp. 139-145.
[25] A Phone-centred body sensor grid system platform: Cost, energy efficiency and user interface. Institute of Electrical and Electronics Engineers,Inc., April 2006. [Online]. Available: http://research.microsoft. com/apps/pubs/default.aspx?id=103003.
[26] K. Park and J. Pak, An integrated gateway for various P.hds in uhealthcareenvironments, BioMed Research International, 2012.
[27] A. M. Rahmani, N. K. Thanigaivelan, T. N. Gia, J. Granados, B. Negash, P. Liljeberg, and H. Tenhunen, Smart e-health gateway: Bringing intelligenceto internet-of-things based ubiquitous healthcare systems, in Annual IEEE Consumer Communications and Grid systemConference. IEEE, 2015, pp. 826-834.
[28] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, Fog computing: Aplatformfor internet of things and analytics, in Big Data and Internetof Things: A Roadmap for Smart Environments. Springer, 2014, pp.169-186.
[29] N. Bessis and C. Dobre, Big data and internet of things: a roadmap forsmart environments. Springer, 2014.
[30] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, Fog computing and itsrole in the internet of things, in Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012, pp. 13-16.
[31] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, The case forVM-based cloudlets in mobile computing, IEEE Pervasive Comput.,vol. 8, no. 4, pp. 14-23, Oct./Dec. 2009.
[32] OpenFog Architecture Overview. OpenFogConsortiumArchitecture Working Group. Accessed on Dec. 7, 2016.[Online]. Available: http://www.openfogconsortium.org/wp-content/uploads/OpenFog-Architecture-Overview-WP-2-2016.pdf.
[33] K. Ha et al., Towards wearable cognitive assistance, in Proc. 12th Annu. Int. Conf. Mobile Syst. Appl. Services, Bretton Woods, NH, USA,2014, pp. 68-81.
[34] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, Clone Cloud: Elastic execution between mobile device and cloud, in Proc. 6th Conf. Comput. Syst., Salzburg, Austria,2011, pp. 301-314.
[35] Atta-ur-Rahman, J. Alhiyafi, Health Level Seven Generic Web Interface", J. Comput. Theor. Nanosci. 15 (4), pp. 1261-1274, 2018.
[36] D. Laney, 3D Data Management: Controlling Data Volume, Velocity, and Variety, Technical Report, Application Delivery Strategies by META Group Inc., 2001.
[37] M. Beyer, Solving Big Data Challenge Involves More Than Just managing Volumes of Data, 2016. http://www.gartner.com/newsroom/id/ 1731916.
[38] J. Granados, A.M. Rahmani, P. Nikander, P. Liljeberg, H. Tenhunen, Towards energy-efficient HealthCare: An internet-of-things architecture using intelligent gateways, in: Proc. of International Conference on Wireless Mobile Communication and Healthcare, 2014, pp. 279-282.
[39] M.L. Hilton, Wavelet and wavelet packet compression of electrocardiograms, IEEE Trans. Biomed. 44 (5) (1997) 394-402.
[40] Z. Lu, D. Youn Kim, W.A. Pearlman, Wavelet compression of ECG signals by the set partitioning in hierarchical trees algorithm, IEEE Trans. Biomed. 47 (7) (2000) 849-856.
[41] R. Benzid, A. Messaoudi, A. Boussaad, Constrained ECG compression algorithm using the block-based discrete cosine transform, Digit. Signal Process. 18 (1) (2008) 56-64.
[42] F. Touati, R. Tabish, U-healthcare system: State-of-the-art review and challenges, J. Med. Syst. 37 (3) (2013).
[43] H.F. Durrant-Whyte, Sensor models and multisensor integration, Int. J. Robot. Res. 7 (6) (1988) 97-113.
[44] Health Level Seven Int'l, Introduction to HL7 Standards, 2017,www.hl7.org/ implement/standards. (Accessed on 30 July 2017).
[45] Netfilter/iptables -nftables project, http://netfilter.org/projects/nftables/. (Accessed on 24 July 2015).
[46] G. Kambourakiset, E. Klaoudatou, S. Gritzalis, Securing medical sensor environments: The codeblue framework case, in Proceeding of the Second International Conference on Availability, Reliability and Security, 2007, pp. 637-643.
[47] J. Ko, J.H. Lim, Y. Chen, R. Musvaloiu-E, A. Terzis, G.M. Masson, T. Gao, W. Destler, L. Selavo, R.P. Dutton, MEDiSN: medical emergency detection in sensor grid systems, ACM Trans. Embed. Comput. Syst. 10 (1) (2010) 11:1-11:29.
[48] S.R. Moosavi, T.N. Gia, A. Rahmani, E. Nigussie, S. Virtanen, H. Tenhunen, J. Isoaho, SEA: A secure and efficient authentication and authorization architecture forIoT-based healthcare using smart gateways, in: Proceeding of 6th International Conference on Amb ient Systems, Grid systems and Technologies, 2015, pp. 452-459.
[49] S.R. Moosavi, T.N. Gia, E. Nigussie, A. Rahmani, S. Virtanen, H. Tenhunen, J. Isoaho, Session resumption-based end-to-end security for healthcare internet of-things, in: Proceeding of IEEE International Conference on Computer and

Information Technology, 2015, pp. 581-588.
[50] Z. Shelby, C. Bormann, 6LoWPAN: The Wireless Embedded Internet, Wiley, UK, 2009.
[51] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, B. Patil, Proxy Mobile 2 IPv6, Internet Engineering Task Force, 2008.
[52] C. Otto, A. Milenkovi?, C. Sanders, E. Jovanov, System architecture of a wireless body area sensor grid systemfor ubiquitous health monitoring, J. Mob. Multimedia 1 (4) (2006) 307-326.
[53] T. Nguyen Gia, N.K Thanigaivelan, A.M. Rahmani, T. Westerlund, P. Liljeberg, H. Tenhunen, Customizing 6LoWPAN grid systems towards Internet-of-Things based ubiquitous healthcare systems, in: Proceeding of NORCHIP, 2014, pp. 1-6.
[54] Kumari., S. Tanwar., S. Tyagi, N. Kumar, M. Maasberg, K. K. R. Choo, Multimedia Big Data Computing and Internet of Things Applications: A Taxonomy and Process Model, Journal of Network and Computer Applications, 124:169-195, 2018.
[55] Kumari A., Tanwar S., Tyagi S., Kumar N., Fog Computing for Healthcare 4.0 Environment: Opportunities and Challenges, Computers and Electrical Engineering, Volume 72, 2018, Pages 1-13 .
[56] J. Vora, P. Devmurari, S. Tanwar, S. Tyagi, N. Kumar, M. S. Obaidat, Blind Signatures Based Secured E-Healthcare System, International Conference on Computer, Information and Telecommunication Systems (IEEE CITS-2018), Colmar, France, 11-13 July 2018, pp. 177-181.
[57] J. Vora, P. Italiya, S. Tanwar, S. Tyagi, N. Kumar, M. S. Obaidat, and K-F. Hsiao, Ensuring Privacy and Security in E-Health Records, International Conference on Computer, Information and Telecommunication Systems (IEEE CITS-2018), Colmar, France, 11-13 July 2018, pp. 192-196
[58] S. Tanwar, P. Patel, K. Patel, S. Tyagi, N. Kumar, M. S. Obaidat, An Advanced Internet of Thing based Security Alert System for Smart Home, International Conference on Computer, Information and Telecommunication Systems (IEEE CITS-2017), Dalian University, Dalian, China, 21-23 July 2017, pp. 25-29.
[59] S. Tanwar, Tyagi S and Kumar S, The Role of Internet of Things and Smart Grid for the Development of a Smart City, Intelligent Communication and Computational Technologies (Lecture Notes in Networks and Systems: Proceedings of Internet of Things for Technological Development, IoT4TD 2017, Springer International Publishing, vol 19, pp. 23-33.
[60] A. Rahman, S Dash, M Kamaleldin, A Abed, A Alshaikhhussain, H Motawei, N Al. Amoudi, W Abahussain, A Comprehensive study of mobile computing in Telemedicine, A. K. Luhachet. al. (Eds):ICAICR 2018,CCIS 956, pp. 413-425, 2019. Springer, Singapore
[61] A Rehman, S Dash, K Sultan, MA Khan, Management of Resource Usage in Mobile Cloud Computing, International Journal of Pure and AppliedMathematics, Volume 119 No. 16, 2018, 255-261.
[62] A. Solanki A., Nayyar A. Green Internet of Things (G-IoT): ICT Technologies, Principles, Applications, Projects, and Challenges. In Handbook of Research on Big Data and the IoT (pp. 379-405). IGI Global, 2019
[63] S.P. Singh, Nayyar, A., Kumar, R., Sharma, A. Fog computing: from architecture to edge computing and big data processing. The Journal of Supercomputing, 1-36, 2018
[64] S. Yi, Z. Hao, Z. Qin, and Q. Li, Fog computing: Platform andapplications, in Proc. 3rd IEEE Workshop Hot Topics Web Syst.Technol. (HotWeb), Washington, DC,USA, 2015, pp. 73-78.

Edited by: Anand Nayyar
Received: Jan 9, 2019
Accepted: Mar 17, 2019

DOI 10.12694/scpe.v20i2.1510
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 207–222. http://www.scpe.org © 2019 SCPE

TRIANGULATION RESOURCE PROVISIONING FOR WEB APPLICATIONS IN CLOUD COMPUTING: A PROFIT-AWARE APPROACH

PARMINDER SINGH [*], POOJA GUPTA [†] AND KIRAN JYOTI [‡]

Abstract. The elasticity feature of cloud attracts the application providers to host the application in a cloud environment. The dynamic resource provisioning arranges the resources on-demand according to the application workload. The over-utilization and under-utilization of resources can be prevented with autonomic resource provisioning. In literature, the Service Level Agreement (SLA) based, load-aware, resource-aware and user-behavior aware solutions have been proposed. The solutions are rigid for a particular metric which provides benefit either to end users or to the application providers. In this article, we proposed a Triangulation Resource Provisioning (TRP) technique with a profit-aware surplus VM selection policy. This policy ensures the fair resource utilization in hourly billing cycle while giving the Quality of Service (QoS) to the end-users. The proposed technique used time series workload forecasting, CPU utilization and response time in the analysis phase. The experiment results show that the TRP resource provisioning technique is a profit-aware approach for the application providers and ensure the QoS to the end-users.

Key words: Cloud computing, autonomic computing, resource provisioning, workload prediction, web applications, auto-scaling.

AMS subject classifications. 68M20, 68N19

1. Introduction.

Cloud computing paradigm provides the Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [34]. The Cloud Service Providers (CSPs) provides the VMs through IaaS with different pricing models such as reserved, on-demand and spot-instances such as Amazon EC2 [2]. Furthermore, the Application Service Providers (ASPs) rent the VMs from CSPs and host the applications. These applications provide SaaS such as web applications for job portal. The end users access web applications through internet and web services [26]. The end users requests are dynamic in nature, and expecting the Quality of Service (QoS). The careful design of resource provisioning techniques can overcome the issue of over-provisioning and under-provisioning called resource oscillation [51].

It is crucial to decide the right amount of resources for the cloud applications. The resource estimation is possible during their execution because it depends upon the incoming workload [41]. The workload fluctuation is quite often in web applications due to irregular access to web services. The network forensics can find initial evidence from web access pattern and further investigation carried for the digital forensics [43]. The over-provisioning state is providing more resources as compared to the required resources, thus lead to high renting cost to ASP. While, in under provisioning state the deployed resources are less as compare to required resources, thus cause the Service Level Agreement (SLA) violation [23]. Therefore, the profit-aware resource provisioning mechanism is required for the less renting cost, while proper utilization of resources in order to meet SLA requirements.

The above-mentioned problems can be solved with dynamic resource provisioning technique. It is an effective approach to provide the resources as per historical data of input workload and current resource utilization. The main goal of ASP is to make maximum profit by minimizing the renting cost and SLA penalty.

In our previous work, the workload prediction model has been developed for the web application workload in cloud computing [46]. The prediction of workload is an important parameter for the estimation of resources. In this paper, we devised a triangulation resource provisioning technique for web applications with the combination of proactive and reactive technique to estimate the resources. The main contribution of this article is as follows:

- The resource provisioning mechanism is designed for ASPs for web applications in the cloud.
- The surplus VM selection algorithm is developed with the profit-aware approach.
- The series of the experiment conducted on real workload weblogs on different metrics.

2. Background.

The resource provisioning without human intervention required a self-configuration, self-optimization, self-healing and self-protective mechanism [29, 19]. Therefore, the autonomic resource provisioning

[*]Lovely Professional University, Phagwara, India. (parminder.16479@lpu.co.in).
[†]Lovely Professional University, Phagwara, India. (pooja.19580@lpu.co.in).
[‡]Guru Nanak Dev Engineering College, Ludhiana, India. (kiranjyotibains@yahoo.com).

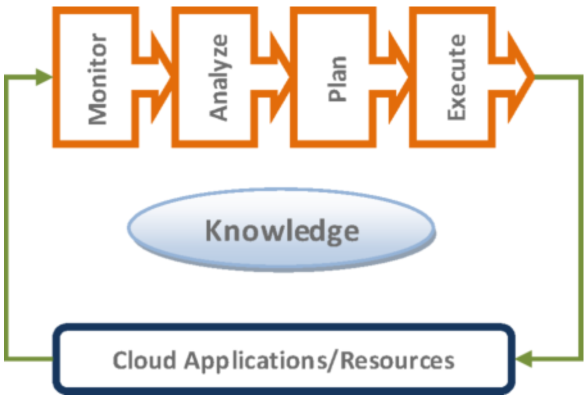


FIG. 2.1. *MAPE-K based cloud resources management.*

is focused on the four phases: Monitoring, Analyzing, Planning and Execution (MAPE) [33] shown in Figure 2.1. The monitoring phase collects the information from the high level and low-level parameters of the cloud environment [17]. Furthermore, the gathered information is analyzed as per reactive, proactive or hybrid approach. Afterward, the planning phase decides the scaling action as scale up/down or scale in/out or either do nothing. Different auto-scaling techniques have been proposed in literature such as threshold rules, fuzzy logic, time-series based, agent-based, etc. [31].

Monitor. The monitor phase fetches the information from the cloud environment about user behavior, SLA violations, and resources utilization [21, 40]. The user decides the threshold, monitoring and scaling intervals, instances bounds, etc. The monitoring performed on the application level, hypervisor level, and VM management.

Analyze. The information collected from the monitoring phase processed for useful inferences to take scaling decision. The reactive and proactive auto-scaler developed by many authors. The reactive approach takes scaling decision on the behalf of current system state. The threshold values are fixed for scale in/out decisions [35]. The predictive approach forecast future CPU utilization or incoming workload, so that the resources could be prepared before the actual workload. The proactive approach could overcome the delayed VM startup problem in the reactive approach.

Plan. The planning phase takes the scaling decisions on the basis of analysis phase parameters. The policies such as resource aware, cost aware, QoS aware, SLA aware and load aware have designed for web application per tier provisioning [27, 11].

Execution. The execution phase took the scaling decision from the planning phase and processed as per the upper and lower resources limit. The VM startup delayed discussed with the client at the time of SLA. The authors [9] designed the executor for web applications in the cloud and, surplus VM selection algorithms have been designed for the scale down decision.

Knowledge. The knowledge is a common repository to be shared with all the four phases of the MAPE loop. It contains the historical logs, policies, and information of resources status [52]. The information is utilized by the autonomic manager.

3. Related Work.

In related work section, the brief overview of resource provisioning techniques are discussed.

Huebscher and MacCann [22] published a survey on MAPE knowledge (MAPE-K) and the autonomic computing. The application of autonomic computing introduced with motivation and basic concepts. Singh et al. [47] published a survey and describe the role of cloud in the fog and edge computing. The author mentioned that fog and edge computing can become more beneficial with the carefully designed resource provisioning mechanism in cloud computing. The dynamic cloud-provisioning model [42] proposed for two-tier applications using the MAPE approach. Maurer et al. [32] devised the autonomic control MAPE loop for the adaptive resource provisioning technique for cloud infrastructure. This technique foundation is threshold rule-based, while in our approach time series prediction and reactive rules used for scaling decisions. The control MAPE loop used to design the adaptive framework to configure the scientific applications in an optimized manner [30]. The author [18] designed the resource provisioning approach with using MAPE-K loop.

The linear regression and neural network applied for analysis of resource utilization. The author also discussed the resource provisioning issues in cloud computing [24]. The proactive scaling applied using Double Exponential Smoothing (DES) along with scaling indicator to estimate the utilization of resources [21]. Bankole and Ajila [1, 10] applied a neural network on the analysis phase. The authors analyze the response time, resource utilization and throughput. The web usage mining can applied via machine learning and pattern recognition to find useful inferences [44, 45, 3]. Kaur and Chann [27] designed the analysis and planning phase for resource provisioning. The decision tree has been proposed in order to filter the effective indicators. Herbst et al. [35] devised a MAPE cycle based scaling policies with a reactive approach. The problem of reconfiguration is a challenge in a reactive approach. The author [20] designed the monitoring phase and proactive analysis phase. The prediction method with minimum error selected on the basis of prediction method categorization from simple to complex attributes. The clustering is crucial mechanism for data mining and, without this its becomes difficult to analyze the big data [38]. Singh and Channa [48] performed the workload clustering and allocation of resources for the different QoS feature for the different kind of workload. De Assun et al. [14] devised a scheduling and auto-scaling strategies to find the user patience of end-users in terms of response time to deliver QoS. Toosi et al. [50] devised an auto-scaling algorithm based on threshold-based reactive policy to utilize renewable energy in a load balancing technique.

A rule-based planning phase designed for resources estimation with 5 scaling rules presented for different architectures [13]. The planning phase improved with SLA-aware policy [16]. The proposed model designed to improve the QoS with minimum cost. A planning phase devised using Machine Learning (ML) approach in order to analyze the resource capacity [40]. Fallah et al. [15] applied ML to learn automata in the planning phase with a reactive and proactive auto-scaling approach to plan the resource allocation. The hybrid planning phase designed for auto-scaling with horizontal and vertical scaling techniques. The application cost optimized with autonomic resource provisioning framework [11]. Molto et al. [37] developed the horizontal and vertical scaling hybridization approach with live migration and over-subscription techniques. A framework has been proposed based on the MAPE loop for resource provisioning [5]. The planning phase designed with learning automata technique for resource utilization forecasting. Aslanpour and Dashti [6] proposed the hybrid technique with the combination of reactive and proactive auto-scaling technique. The planning phase has a two-sided policy having one side with SLA violation and the second side is resource utilization. The author [36] analyzed the surplus VMs decisions with cost and cost-aware policies. Author found the response time and resource utilization are two important factors need to consider to release the surplus VMs.

The state-of-the-art articles focused on SaaS resource provisioning. The algorithms, models, mechanism, framework, and architecture have been designed and developed. The resource provisioning technique designed with two methods: Reactive and Proactive. The reactive approach is providing the resources based on threshold-based rules. The major concern in reactive approach is VM startup and setup time, which usually varies from 5 to 15 minutes [27, 4, 8]. This elevates the SLA violation. The proactive method estimates the resources and provides the resources before the incoming workload. Therefore, the proactive technique could decrease the SLA violation.

In this article, the resource provisioning architecture and technique designed for application providers. The proposed mechanism is a combination of reactive and proactive approaches.

4. Proposed Approach.

In this section, the proposed resource provisioning approach is discussed in detail. The devised approach followed the IBMs K-MAPE loop and execute the modules at specific scaling interval. The triangulation approach used a hybrid approach by using proactive and reactive method for resource provisioning. The proposed model is the trade-off between SLA and cost, thus saving the cost while giving the desired QoS. The goal of the proposed technique is to give maximum profit to ASP in term of cost. The cloud architecture for proposed resource provisioning technique is shown in Figure 4.1. The architecture is considered ASP and CSP as part of a cloud environment.

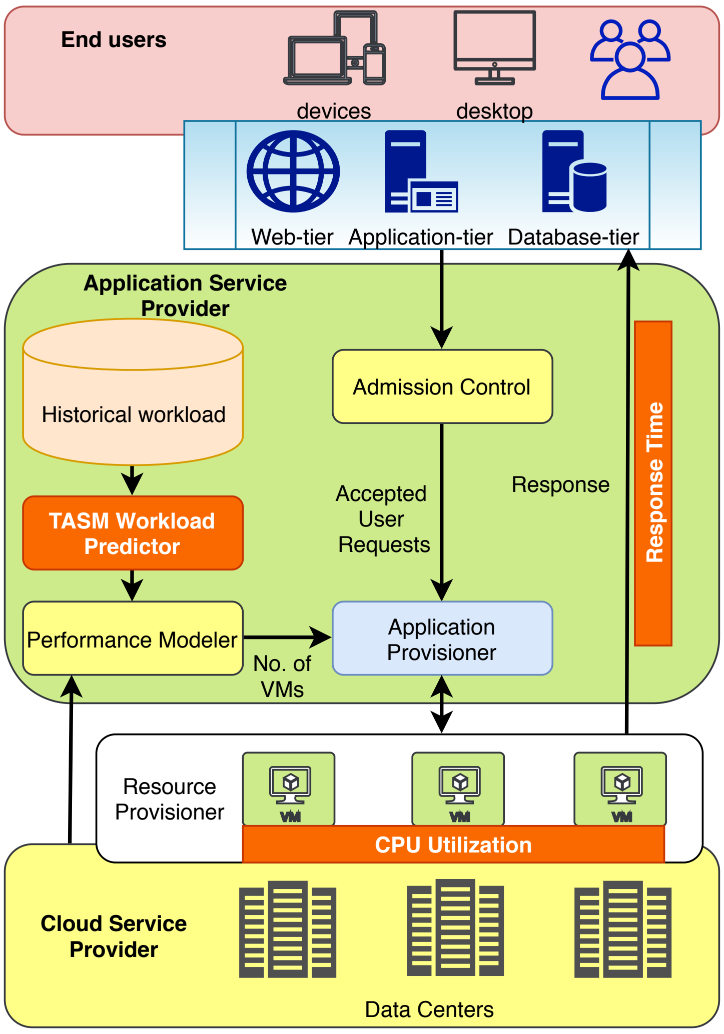


Fig. 4.1. *The cloud resource provisioning architecture.*

4.1. Monitoring Phase. The monitoring phase is generally stored the information about the resource utilization such as CPU utilization called resource-aware approach [51, 29] and, the technique which stores the response time usually called SLA-aware approach. The monitoring of resources, SLA and the incoming requests can take provisioning decision in an effective manner. Figure 4.1 shows the monitoring phase parameters as CPU utilization, response time and incoming workload in each scaling interval.

4.2. Analysis Phase. The analysis phase gives the scaling decision by analyzing the resources, SLA and user behavior, this combination name given as triangulation resource provisioning approach. The performance modeler estimates the number of VMs is working in the analysis phase. The TRP mechanism proposed for effective utilization of resources while giving the response in desired time as mentioned in SLA agreement with the user. Otherwise, the ASP has to pay penalty for the delayed services. The TRP model shown in Figure 4.2. The detailed working of each module of proposed TRP model as per following:

4.2.1. Workload prediction. In our previous work, we developed the time series based Technocrat ARIMA and SVR Model (TASM) [46] for predicting the web application workload. The TASM gives better accuracy as compare to other state-of-the-art prediction models with higher efficiency. In this paper, we used the TASM prediction model to predict the incoming workload. The analytical process for workload forecasting

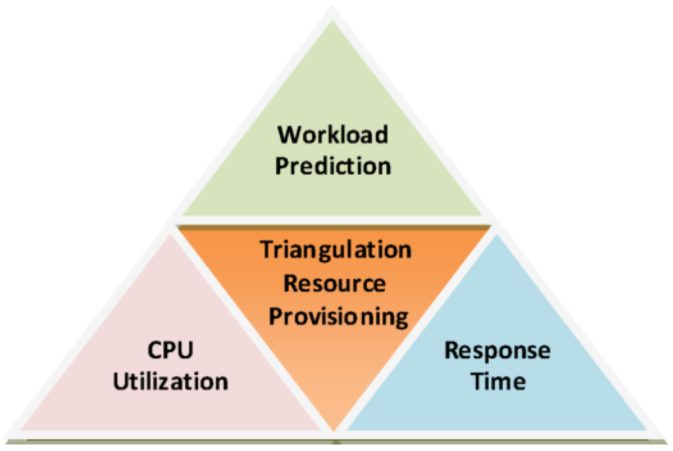


FIG. 4.2. *The proposed triangulation resource provisioning (TRP) approach.*

shown in Figure 4.3. The monitoring phase record the arrival rate and store in the repository named historical workload. Further, the arrival rate is compiled to a discrete time series. The classification on time series performed to select appropriate LR, ARIMA and SVR model. The non-seasonal and seasonal forecasting method applied and fetches the predicted value. The article [46] can be referred for more detail working of the model. In our previous work, short term prediction of 10 minutes is processed. In this work, 1 minute short term prediction has been used to predict the number of requests for a minute.

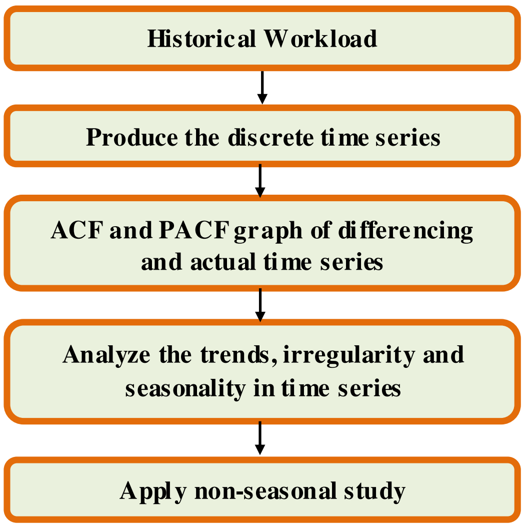

Historical Workload
Produce the discrete time series
ACF and PACF graph of differencing and actual time series
Analyze the trends, irregularity and seasonality in time series
Apply non-seasonal study

FIG. 4.3. *The TASM analytical process for workload forecasting.*

4.2.2. CPU utilization. The resource utilization can be observed from the CPU utilization of each VM deployed for the service of the web application in cloud infrastructure. The average CPU utilization has been calculated as per Eq. 4.1.

$$AverageCpuUtilization = \frac{\sum_{i=1}^{OnlineVms} VM_i CPUutilization}{OnlineVMs} \tag{4.1}$$

where VM_i utilization is the CPU utilization of VM_i and, $OnlineVMs$ are the total VMs including in-service VMs and startup VMs.

$$VM_iCPUutilization = \frac{\sum_{j=1}^{currentcloudlets}(CloudletPEs_j * CloudletLength_j}{PE * MIPS} \tag{4.2}$$

where the number of incoming requests is $CurrentCloudlets$ in the scheduler of VM. The number of processors required is $CloudletPEs$ and number of instruction in each VM in $cloudletLength$. The VM processing elements in number are represented with PE and Million Instruction Per Second ($MIPS$) is the processing capacity of each PE.

4.2.3. Response Time. The response time during the last minute is calculated using Eq. 4.3. It is an important criterion for evaluation of SLA violation. It is further used to select the surplus VM in profit aware approach.

$$AverageResponseTime = \frac{\sum_{i=1}^{TotalProcessedCloudlets} ResponseTime_i}{TotalProcessedCloudlets} \tag{4.3}$$

where $TotalProcessedCloudlets$ are the finished cloudlets in past minute. $ResponseTime_i$ is the delay in i^{th} request's response. The delay is further calculated by using Eq. 4.4.

$$ResponseTime_i = FinishTime_i - ProcessTime_i - ArrivalTime_i \tag{4.4}$$

where the $ArrivalTime_i$ is cloudlet accepted time to the cloud and, $ProcessTime_i$ is the processing time of the cloudlet. $FinishTime_i$ is time recorded from the Clock at finished request.

4.3. Planning Phase. The planning phase applied the rule-based technique to obtain the scaling decision. The compiled values form analysis phase such as predicted workload in scaling interval, average CPU utilization and response time is used in this phase. The threshold values are set for CPU utilization e.g. $20\% - 80\%$ and response time $0.2s - 1.0s$ for lower and upper threshold values respectively.

In the first case, the comparison of the response time and CPU utilization with the defined upper-threshold values, if any observed value greater than upper-threshold values then again forecast values are compared with the current workload. If the current number of *cloudlets* are less as compared to forecast the number of *cloudlets* than $scale - up$.

In the second case, if response time and CPU utilization both are less than lower-threshold values then further evaluate the current cloudlets with forecast cloudlets. If current cloudlets are greater than $scale - down$, otherwise $do - nothing$.

This decision of $scale - up$, $scale - down$ and $do - nothing$ further input to the execution phase.

4.4. Execution Phase. This phase takes the scaling decision scale-up, scale-down or do nothing as an input from the planning phase. The scale-down policy ensure de-provisioning of extra resources in cloud infrastructure. The profit-aware approach for selecting the surplus VM is presented in this paper. The proposed method designed as per the billing cycle of Amazon EC2 instance, which is providing hourly billing cycle. Algorithm 1, show the detailed working of profit-aware surplus VM selection algorithm. The description of notations used in the article is present in Table 4.1.

According to the algorithm 1, the load on the VM and pending minutes in current scaling interval have been considered. The line no. $(4 - 11)$, prepare the $onDemandSurplusVMs$, which are the potential VMs per scaling interval that can scale down. In line no. $(12 - 13)$, if there is only 1 VM select that VM for scale down. In line no. $(14 - 15)$, if there is no VM in the current scaling interval, means every VM having pending service time in current billing hour more than $\triangle s$ than no VM selected as surplus VM. Furthermore, if the number of VMs in $ondemandSurplusVMs$ list is greater than 1, than we have to choose that VM for scale down, whose pending minutes are equals to the processing time required to process the request in the scheduler of that VM. This result in saving the cost as well give the QoS to the end user. In order to do this, line no. $(17 - 29)$ find the VM with least workload. The function *calcuatePendingST* is calculated the service time of each cloudlet in the scheduler. The line no. $(24 - 27)$ select the VM with least workload. In line no. (30), the algorithm returns the $surplusVM$ to the resource provisioner in CSP.

Table 4.1
Description of notations

Parameter	Description
$Clock$	Timer for Simulation
$\triangle s$	Scaling Interval
OnDemandVM	Combined List of in-service and pending VMs
VM^P	List of VMs about to ready or in ready state
VM^S	List of VMs in-service
$OnDemandSurplusVMs$	List of potential surplus VMs in scaling interval
LST	VM's left service time in current billing hour
$cloudletList$	List of incoming request called cloudlets
SurplusVM	In algorithm 1, the VM finally selected for scale down

Algorithm 1 The pseudo code of proposed profit-aware surplus VM selection policy

```
1: Input: OnDemandVM List ($VM^S + VM^P$)
2: Output: surplusVM
3: Variables: $availableTime, remainingTime$, $onDemandSurplusVMs \leftarrow null$, $minLST \leftarrow anHour$
4: for vm in onDemandVmList do        ▷ Prepare the VMs list having left service time less than or equals to scaling interval
5:     $LST \leftarrow null$;
6:     $availableTime \leftarrow Clock - VM.getRequestTime()$
7:     $LST \leftarrow anHour - (availabletime\%anHour)$
8:     if LST<= $\triangle s$ then
9:         onDemandSurplusVMs.add(vm);
10:    end if
11: end for
12: if onDemandSurplusVMs.legnth() == 1 then        ▷ If list containing only 1 VM
13:     $surplusVM \leftarrow onDemandSurplusVMs.pop()$        ▷ Select the first VM as surplusVM
14: else if onDemandSurplusVMs.empty() then        ▷ If list is empty
15:     surplusVM ← $null$
16: else        ▷ If list containing more than 1 VMs
17:     for vm in onDemandSurplusVMs do
18:         $vm.LST \leftarrow 0$
19:         $cloudletList \leftarrow vm.getScheduler().getCloudletList()$
20:         for cloudlet in cloudletList do
21:             $pendingServiceTime \leftarrow calculatePendingST(cloudlet)$
22:             $vm.LST \leftarrow vm.LST - pendingServiceTime$
23:         end for
24:         if $abs(vm.LST) < minLST$ then        ▷ abs stands for absolute value
25:             $minLST \leftarrow vm.LST$
26:             surplusVM ← vm
27:         end if
28:     end for
29: end if
30: return surplusVm;
```

5. Experiment Evaluation. In this section, the proposed model TRP simulation performed using CloudSim [12], a simulation toolkit for cloud computing infrastructure. R Tool used to implement the prediction model. Furthermore, R Caller library integrates the R Scripts in the CloudSim.

5.1. Experiment Setup. The ClarkNet [49] weblog has been used in the experiment. The various patterns are present in the time series to evaluate the performance of the present algorithm in normal and peak workloads.

Table 5.1
Detail of ClarkNet dataset

Parameter	Value
Log Files	Aug28log, access1
No. of Requests	3,328,587
Duration	2 weeks
Discrete Time series	1 minute

The detail of the workload presented in Table 5.1.

The VMs of bigger size provisioned in case of heavy workload, while small VMs provisioned in case of light workload. The time taken in VM startup considered as 5 minutes. The time shared scheduling policy has been applied in the experiment.

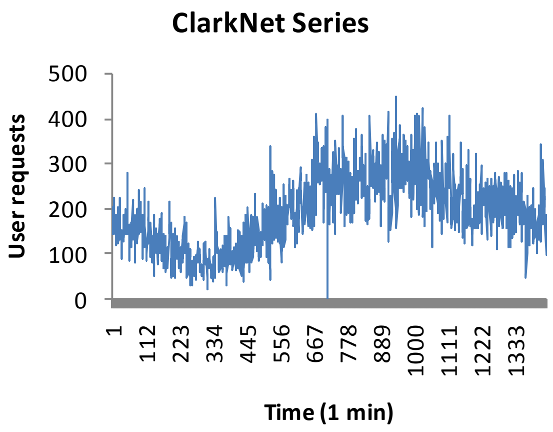


Fig. 5.1. *ClarkNet workload of the* 6^{th} *day in week.*

5.2. Performance Evaluation. The following metrics calculated for performance evaluation:

Response Time: The delay in answering the request is known as response time. The delay in response time is agreed at the time of catering the cloud services. The minimum response time recorded in the cloud environment is $200ms$. Some articles [35, 11] have considered the response from $700ms$ to $1200ms$. In our experiment, we considered the $1000ms$ to ensure the QoS to the end user. The recorded response time utilized for scaling decision in the proposed TRP technique.

SLA Violation: The SLA violation is considered as the delay in response time which is mentioned in the SLA. In our experiments, the desired response time is $1000ms$. The percentage of SLA violation calculated to charge the penalty to ASP.

Cost: The resource provisioning mechanism objective is to reduce the cost of application deployment while giving the QoS to the end user. The techniques are able to reduce the cost but ASP has to pay the penalty cost. The sum of rental cost and penalty cost imposed the loss to the ASP. The cost of ASP calculated as the sum of renting cost and SLA penalty.

5.3. Results and Discussion. The performance and efficiency of the proposed TRP mechanism presented in this section.

5.3.1. Workload Prediction. The request arrival rate predicted using the TASM [46] prediction approach. In our previous work, TASM has been proposed and find the prediction accuracy of TASM for the web application is satisfactory as compared to the existing time series prediction model. The TASM is an adaptive prediction model which is the combination of LR, ARIMA and SVR model. These models are selected based on the workload pattern classification. The weblogs first compiled to the discrete time series model, in our previous work the 10 minutes short time prediction has been performed. In this approach, 1-minute short time prediction has been performed and further applied in proposed TRP mechanism. The prediction of ClarkNet series using TASM shown in Figure 5.2.

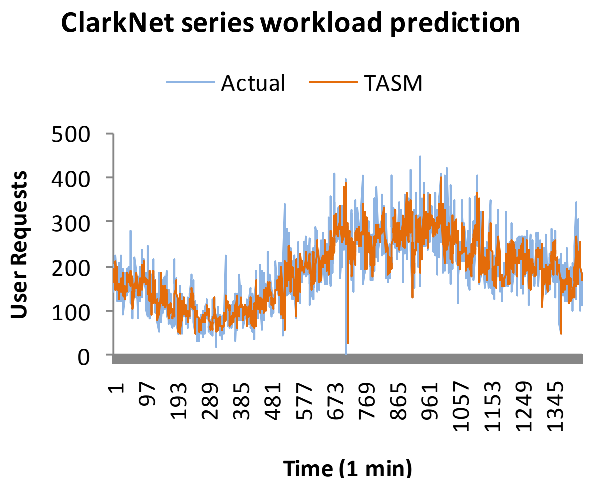


FIG. 5.2. *Workload prediction of ClarkNet series using TASM prediction model.*

$$MAPE = \frac{1}{n}\sum_{s=1}^{n}\left|\frac{actualRequests_s - predictedRequests_s}{actualRequests_s}\right| \times 100\% \tag{5.1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{s=1}^{n}(actualRequests_s - predictedRequests_s)^2} \tag{5.2}$$

The prediction models accuracy has been calculated with Mean-Absolute-Percentage-Error (MAPE) in Eq. 5.1 and Root-Mean-Square-Error (RMSE) in Eq. 5.2. The experiment results show the RMSE is 42.24 and MAPE as 20.63% of TASM prediction model as compare to LR values are 64.37 and 34.52% and ARMA values are 51.24 and 32.98% receptively for ClarkNet series.

5.3.2. Triangulation Resource Provisioning Mechanism. The TRP mechanism in detail explained in section 4.2. and Figure 4.2. The TRP approach consist of three important parameters to perform the provisioning for web applications in cloud computing. The first approach is related to the resource behavior of the provisioned VMs (singular approach). The monitoring phase records the average CPU utilization after every scaling interval. The threshold based techniques are set upper and lower threshold for provisioning of VM e.g. 80% upper threshold for scale up and 20% lower threshold to scale down decision. The second approach (double approach) add another parameter response time along with CPU utilization. The response time is related to SLA behavior, with which evaluation of SLA violation would be compiled and necessary actions could be taken. Furthermore, the response time is also evaluated on the basis of threshold values. The proposed model is based on the 3-D approach designed by the Aslanpour et al. [8]. The proposed model is known as triangulation resource provisioning approach (TRP) added the third parameter as user-behavior analysis. The future incoming workload is forecasted and considered as the third parameter. The LR, ARIMA and SVR models are used in a TASM prediction model to select the best prediction model based on workload classification. A new profit aware surplus VM selection policy has been proposed to balance the cost to application provider and QoS to the end user. The experiment results shows the performance evaluation of singular-double-TRP mechanisms for resource utilization, QoS and Cost.

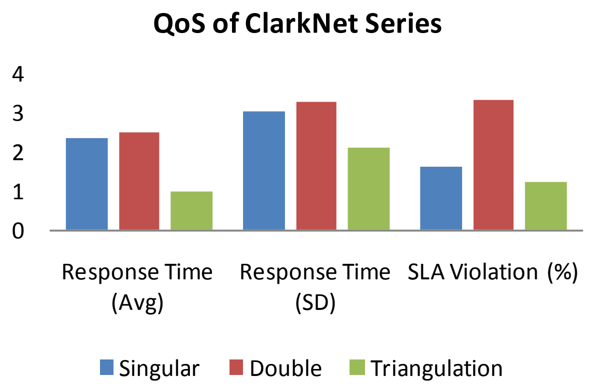


Fig. 5.3. *The QoS of singular, double and proposed triangulation mechanism for ClarkNet Series.*

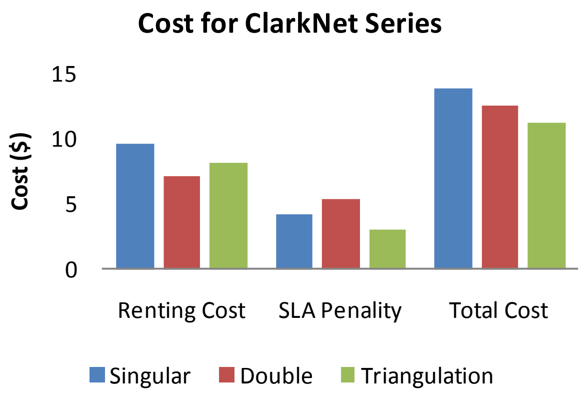


Fig. 5.4. *The cost of singular, double and proposed triangulation mechanism for ClarkNet Series.*

The Figure 5.3 and Figure 5.4 shows the experiment results of Singular, Double and Triangulation resource provisioning. The response time of singular and double technique response time is greater than 1 second which is higher than the required SLA parameter. The proposed triangulation technique is giving QoS as per SLA. The Standard Deviation (SD) of response time depicted by the purposed model assure QoE to the end users thus reduce overall SLA violation percentage. The cost of provisioning is not only the renting cost, but it also combined with the SLA penalty. The purposed triangulation resource provisioning with workload prediction reduces the overall provisioning cost of ClarkNet workload. The proposed TRP model able to reduce the SLA violation and 20% of overall provisioning cost.

5.3.3. Profit-aware surplus VM selection policy. The execution phase in Amazon EC2 randomly selects the VM in the scale-down decision. The policy in state-of-the-art are designed and implement such as Last Up First Down (LUFD) [6], First Up First Down (FUFD) [7], cost-aware and load-aware[9, 8] policies. The super professional executor has been used in the experiment [9]. A new profit-aware surplus VM selection policy presented in this paper. The cost-aware policy selects the VM for scale-down having maximum service time in the current hour regardless the number of requests processed on that VM. If the number of jobs will not complete in the scaling interval than all the jobs will be failed and thus lead to SLA penalty and lead to poor QoE to the end user and, ASP has to pay the penalty cost also. The second technique proposed is load aware policy which selects the VM with the least workload to ensure QoS to the end user. This method can

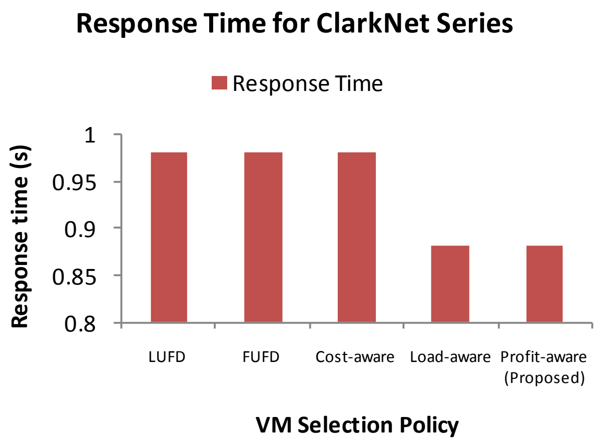


Fig. 5.5. *The response time of surplus VM selection policies for ClarkNet Series.*

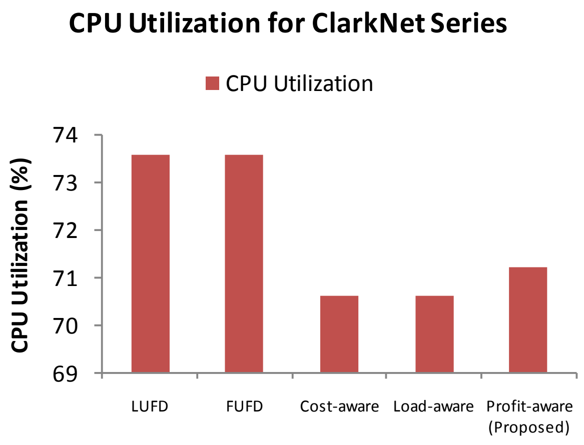


Fig. 5.6. *The CPU utilization of surplus VM selection policies for ClarkNet Series.*

select any machine to de-provision with the least workload may select the VM with maximum minutes pending in current hourly billing cycle thus lead to a more renting cost to the ASP.

In this paper, the profit-aware approach first shortlists the VMs having pending service time less than or equal to the scaling interval. For example, if the scaling interval is 10 minutes, then the VMs having pending service ranges from 0 to 10 minutes has been shortlisted. Afterward, out of the shortlisted VMs, the VM with minimum workload has been selected for the scale down. This process repeated in every scaling interval. The experiment results show that the proposed surplus VM selection policy gives the prominent benefit to the ASP and end user.

We have conducted the experiment with two scenarios on ClarkNet dataset. Both the scenarios are conducted with the proposed TRP mechanism but with the different executors. The first experiment is conducted with ClarkNet workload. The default techniques LUFD and FUFD have been tested with the simple executor. The cost-aware, load-aware and proposed profit-aware surplus VM selection policies are tested with super-professional executor (suprex) [9], which provides the quarantined VM option. The suprex executor will not immediately de-provision the VM selected for scale down. It moves to the quarantined VM list and de-provision

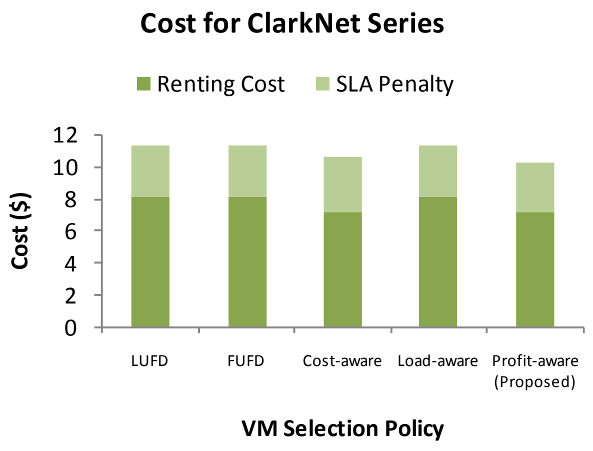


FIG. 5.7. *The cost of surplus VM selection policies for ClarkNet Series.*

when the time period of the hourly billing cycle is complete. In between if VM is required; it can be recalled as in-service VM.

The Figure 5.5, Figure 5.6 and Figure 5.7 shows the comparison of state-of-the-art and proposed profit-aware surplus VM selection policies for response time, CPU utilization and cost metrics respectively. The load-aware and proposed profit-aware policies response time is minimum as compare to other techniques, thus gives better QoE to the end-users with quick response. The LUFD and FUFD techniques gives highest CPU utilization, even the proposed technique gives better CPU utilization from cost-aware and load-aware policies. The LUFD and FUFD is not supported in super-professional executer, which results higher renting cost even due to higher CPU utilization. The proposed VM selection policy gives upto 16% of cost benefit as compare to other surplus VM selection policies. The experiment result shows that the proposed technique gives better QoS with minimum cost.

5.3.4. Comparison of resource provisioning mechanisms. The proposed approach compared with the two baseline approaches named as Cost-aware (LRM) [25] presents prediction based dynamic resource prediction model. The author applied Linear Regression (LR) model for workload forecasting. The second approach under consideration is cost-aware (ARMA) [39]. The second order autoregressivemoving-average (ARMA) model applied for workload forecasting. The third approach is the proposed profit-aware (TRP) resource provisioning model. The TASM prediction model has been used to predict the incoming requests and TRP mechanism is used for resource provisioning and surplus VMs are selected by the profit-aware approach. The strategies under consideration are worked on a proactive resource provisioning model under the MAPE loop.

The CPU utilization of three approaches shown in Figure 5.8 for ClarkNet series at each scaling interval. The simulation result shows the CPU utilization of cost-aware (LRM), cost-aware (ARMA) and proposed profit-aware (TRP). The CPU utilization in some scaling intervals is more than 100%, because the ASP does not have enough resources to process the incoming requests. These conditions lead to SLA violation due to under-provisioning of resources. The result has shown that no technique is able to give 100% CPU utilization. The ARMA and LR have more spikes as compare to profit-aware (ARMA), which clearly indicates that proposed resource provisioning mechanism is successfully overcome the over-provisioning issues for web application in the cloud computing environment.

The delay in request response is shown in Figure 5.9 for three approaches at every scaling interval. The delay in response gives poor QoS to the end users and leads to SLA violation. The ASP has to pay penalty for delay more than decided in SLA agreement. This happens due to under-provisioning and, over-provisioning state having an extra burden of cost to the ASP. The SLA violation minimizes the profit of ASP. In our experiment,

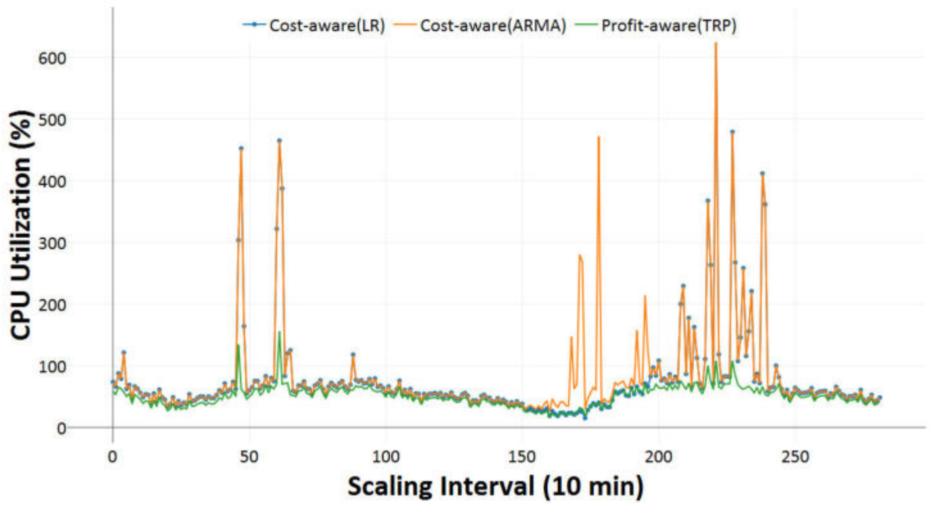


Fig. 5.8. *The comparison of CPU utilization for ClarkNet Series.*

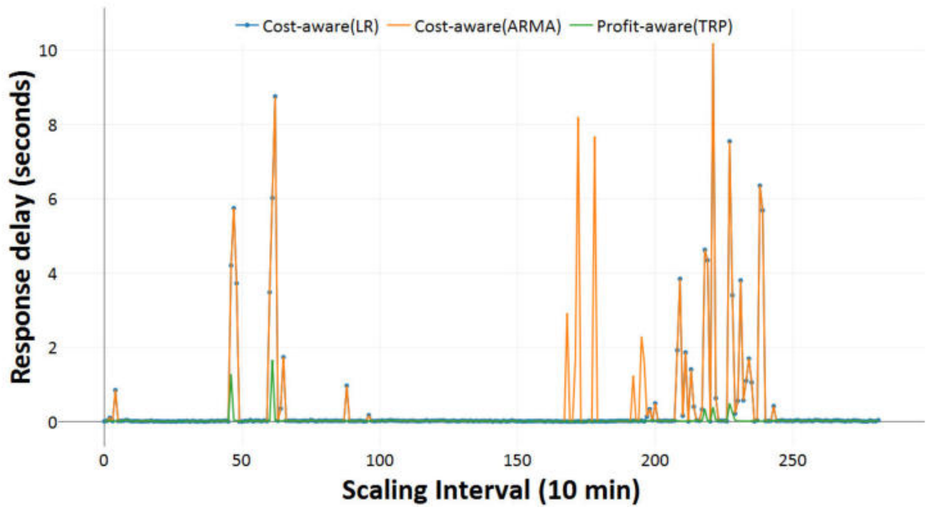


Fig. 5.9. *The comparison of response delay for ClarkNet Series.*

we consider response time as per SLA as $1s$. The Figure 5.9 shows the delay more than $1s$. The comparison result observed the high spikes, which indicates the higher delay in response to the user. The profit-aware approach average delay is equal to 0, only a few small spikes observed. It strongly indicates that the proposed profit-aware (TRP) mechanism is able to give QoS to the end users and reduce the SLA violation.

The VM allocations are shown in Figure 5.10 for three mechanisms for ClarkNet time series at each scaling interval. The cost-aware (LRM) and cost-aware (ARMA) approaches have more under-utilization and over-provisioning states as compare to proposed profit-aware (TRP) approach.

The renting cost and SLA penalty cost of three mechanisms for is shown in Figure 5.11. The renting cost of profit-aware(TRP) approach is higher than cost-aware (LR) and cost-aware(ARMA) model, but when we compared with the SLA penalty, it experienced least SLA penalty. The overall provisioning cost would be less than other provisioning mechanisms. It has been observed that the proposed approach provides the QoS to end-users and comparatively offer 12% more profit to application providers.

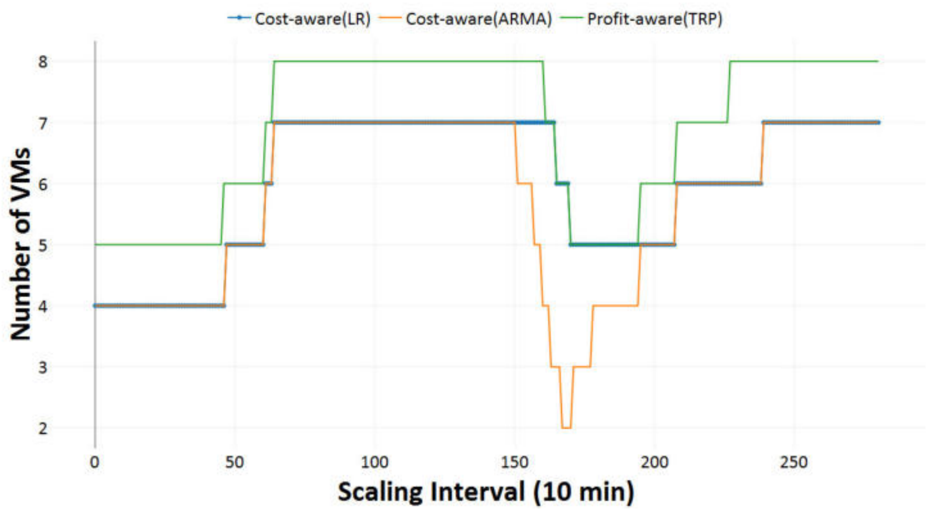


FIG. 5.10. *The comparison of VM allocation for ClarkNet Series.*

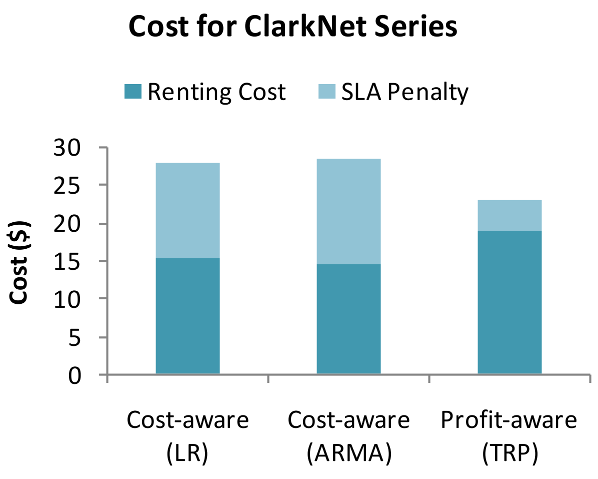


FIG. 5.11. *The comparison of cost for ClarkNet Series.*

6. Conclusion and Future Work. In this article, resource provisioning mechanism presented for web application in cloud computing. The triangulation resource provisioning (TRP) approach has been designed using three parameters: Workload prediction, CPU utilization and response time. We also presented the profit-aware surplus VM selection policy in the scale-down decision in horizontal scaling. The presented model is supporting the MAPE control loop. The workload fluctuations are forecast with the TASM prediction model. The response time, CPU utilization and predicted request are applied in the analysis and planning phase for scaling decisions. The profit-aware surplus VM selection policy used in the execution phase for select the appropriate VM for scale-down. The real workload traces ClarkNet weblog used to evaluate the performance of existing and proposed provisioning mechanisms. CloudSim and R tool used to simulate and obtaining the comparison results. The result shows that the proposed model for web applications provides fair utilization of resources with minimum cost, thus provides maximum profit to application provider and QoE to the end users.

In the future, authors can design new prediction models with higher efficiency. The vertical scaling can be combined with horizontal scaling to mitigate the VM start-up delay issue. The new surplus VM selection policies can be designed for a multi-cloud environment.

REFERENCES

[1] Ajila, S. A. and Bankole, A. A.(2013). Cloud client prediction models using machine learning techniques. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 134–142. IEEE.

[2] Al-Roomi, M., Al-Ebrahim, S., Buqrais, S., and Ahmad, I. (2013). Cloud computing pricing models: a survey. *International Journal of Grid and Distributed Computing*, 6(5):93–106.

[3] Alzubi, J., Nayyar, A. and Kumar, A.(2018). Machine learning from theory to algorithms: an overview. *Journal of Physics: Conference Series*, 1142(1):012012.

[4] Antonescu, A.-F. and Braun, T.(2016). Simulation of sla-based vm-scaling algorithms for cloud-distributed applications. *Future Generation Computer Systems*, 54:260–273.

[5] Arani, M. G., Shamsi, M., et al. (2015). An extended approach for efficient data storage in cloud computing environment. *International Journal of Computer Network and Information Security*, 7(8):30.

[6] Aslanpour, M. S. and Dashti, S. E.(2016). Sla-aware resource allocation for application service providers in the cloud. In *Web Research (ICWR), 2016 Second International Conference on*, pages 31–42. IEEE.

[7] Aslanpour, M. S. and Dashti, S. E.(2017). Proactive auto-scaling algorithm (pasa) for cloud application. *International Journal of Grid and High Performance Computing (IJGHPC)*, 9(3):1–16.

[8] Aslanpour, M. S., Dashti, S. E., Ghobaei-Arani, M., and Rahmanian, A. A.(2018). Resource provisioning for cloud applications: a 3-d, provident and flexible approach. *The Journal of Supercomputing*, 74(12):6470–6501.

[9] Aslanpour, M. S., Ghobaei-Arani, M., and Toosi, A. N.(2017). Auto-scaling web applications in clouds: A cost-aware approach. *Journal of Network and Computer Applications*, 95:26–41.

[10] Bankole, A. A. and Ajila, S. A. (2013). Cloud client prediction models for cloud resource provisioning in a multitier web application environment. In *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, pages 156–161. IEEE.

[11] Beltrán, M.(2015). Automatic provisioning of multi-tier applications in cloud computing environments. *The Journal of Supercomputing*, 71(6):2221–2250.

[12] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R.(2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50.

[13] Casalicchio, E. and Silvestri, L.(2013). Mechanisms for sla provisioning in cloud-based service providers. *Computer Networks*, 57(3):795–810.

[14] de Assunção, M. D., Cardonha, C. H., Netto, M. A., and Cunha, R. L.(2016). Impact of user patience on auto-scaling resource capacity for cloud services. *Future Generation Computer Systems*, 55:41–50.

[15] Fallah, M., Arani, M. G., and Maeen, M.(2015). Nasla: Novel auto scaling approach based on learning automata for web application in cloud computing environment. *International Journal of Computer Applications*, 113(2).

[16] García, A. G., Espert, I. B., and García, V. H.(2014). Sla-driven dynamic cloud resource management. *Future Generation Computer Systems*, 31:1–11.

[17] Ghanbari, H., Simmons, B., Litoiu, M., and Iszlai, G. (2011). Exploring alternative approaches to implement an elasticity policy. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 716–723. IEEE.

[18] Ghobaei-Arani, M., Jabbehdari, S., and Pourmina, M. A.(2018). An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach. *Future Generation Computer Systems*, 78:191–210.

[19] Gill, S. S., Chana, I., Singh, M., and Buyya, R.(2019). Radar: Self-configuring and self-healing in resource management for enhancing quality of cloud services. *Concurrency and Computation: Practice and Experience*, 31(1):e4834.

[20] Herbst, N. R., Huber, N., Kounev, S., and Amrehn, E.(2014). Self-adaptive workload classification and forecasting for proactive resource provisioning. *Concurrency and computation: practice and experience*, 26(12):2053–2078.

[21] Huang, J., Li, C., and Yu, J.(2012). Resource prediction based on double exponential smoothing in cloud computing. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 2056–2060. IEEE.

[22] Huebscher, M. C. and McCann, J. A.(2008). A survey of autonomic computingdegrees, models, and applications. *ACM Computing Surveys (CSUR)*, 40(3):7.

[23] Iqbal, W., Dailey, M. N., Carrera, D., and Janecek, P.(2011). Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems*, 27(6):871–879.

[24] Islam, S., Keung, J., Lee, K., and Liu, A.(2012). Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155–162.

[25] J. Yang, C. Liu, Y. Shang, B. Cheng, Z. Mao, C. Liu(2014) A cost-aware auto-scaling approach using the workload prediction in service clouds. In *Information Systems Frontiers*, 16:7–18.

[26] Joshi, N. and Shah, S.(2019). A comprehensive survey of services provided by prevalent cloud computing environments. In *Smart Intelligent Computing and Applications*, pages 413–424. Springer.

[27] Kaur, P. D. and Chana, I.(2014). A resource elasticity framework for qos-aware execution of cloud applications. *Future Generation Computer Systems*, 37:14–25.

[28] Kaneriya, S., Tanwar, S., Nayyar, A., Verma, J.P., Tyagi, S., Kumar, N., Obaidat, M.S. and Rodrigues, J.J.(2018). Data Consumption-Aware Load Forecasting Scheme for Smart Grid Systems. *2018 IEEE Globecom Workshops (GC Wk-*

shps), pages 1–6. IEEE.

[29] Kim, H., el Khamra, Y., Jha, S., and Parashar, M.(2009). An autonomic approach to integrated hpc grid and cloud usage. In *e-Science, 2009. e-Science'09. Fifth IEEE International Conference on*, pages 366–373. IEEE.

[30] Koehler, M.(2014). An adaptive framework for utility-based optimization of scientific applications in the cloud. *Journal of Cloud Computing*, 3(1):4.

[31] Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J. A.(2014). A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of grid computing*, 12(4):559–592.

[32] Maurer, M., Brandic, I., and Sakellariou, R.(2013). Adaptive resource configuration for cloud infrastructure management. *Future Generation Computer Systems*, 29(2):472–487.

[33] Maurer, M., Breskovic, I., Emeakaroha, V. C., and Brandic, I.(2011). Revealing the mape loop for the autonomic management of cloud infrastructures. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 147–152. IEEE.

[34] Mell, P., Grance, T., et al.(2011). The nist definition of cloud computing.

[35] Mohamed, M., Amziani, M., Belaïd, D., Tata, S., and Melliti, T.(2015). An autonomic approach to manage elasticity of business processes in the cloud. *Future Generation Computer Systems*, 50:49–61.

[36] Moldovan, D., Truong, H.-L., and Dustdar, S.(2016). Cost-aware scalability of applications in public clouds. In *Cloud Engineering (IC2E), 2016 IEEE International Conference on*, pages 79–88. IEEE.

[37] Moltó, G., Caballer, M., and de Alfonso, C.(2016). Automatic memory-based vertical elasticity and oversubscription on cloud platforms. *Future Generation Computer Systems*, 56:1–10.

[38] Nayyar, A. and Puri, V.(2017). Comprehensive Analysis & Performance Comparison of Clustering Algorithms for Big Data. Review of Computer Engineering Research. *Review of Computer Engineering Research*, 4(2):54–80.

[39] N. Roy, A. Dubey, and A. Gokhale(2011). Efficient autoscaling in the cloud using predictive models for workload forecasting. In *International Conference on Cloud Computing (CLOUD)*, pages 500–507. IEEE.

[40] Qavami, H. R., Jamali, S., Akbari, M. K., and Javadi, B.(2013). Dynamic resource provisioning in cloud computing: a heuristic markovian approach. In *International Conference on Cloud Computing*, pages 102–111. Springer.

[41] Qu, C., Calheiros, R. N., and Buyya, R.(2018). Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Computing Surveys (CSUR)*, 51(4):73.

[42] RahimiZadeh, K., AnaLoui, M., Kabiri, P., and Javadi, B.(2015). Performance modeling and analysis of virtualized multi-tier applications under dynamic workloads. *Journal of Network and Computer Applications*, 56:166–187.

[43] Saxena, A.,Shrivastava, G., Sharma, K.(2012). Forensic investigation in cloud computing environment. *International Journal of forensic computer science*, 2:64–74.

[44] Sharma, K., Shrivastava, G. and Kumar, V.(2011). Web mining: Today and tomorrow *2011 3rd International Conference on Electronics Computer Technology*, pages 399–403. IEEE.

[45] Sharma, K. and Bhatnagar, V.(2011). Private and Secure Hyperlink Navigability Assessment in Web Mining Information System *International Journal on Computer Science and Engineering*, 3(6):2245–2250.

[46] Singh, P., Gupta, P., and Jyoti, K.(2018). Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. *Cluster Computing*, pages 1–15.

[47] Singh, S.P., Nayyar, A., Kumar, R. and Sharma, A.(2018). Fog computing: from architecture to edge computing and big data processing. *The Journal of Supercomputing*, pages 1–36.

[48] Singh, S. and Chana, I.(2016). Resource provisioning and scheduling in clouds: Qos perspective. *The Journal of Supercomputing*, 72(3):926–960.

[49] Stephen Balbach ClarkNet web server logs. In *Available: http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html*, Accessed on: 2019, 20 February.

[50] Toosi, A. N., Qu, C., de Assunção, M. D., and Buyya, R.(2017). Renewable-aware geographical load balancing of web applications for sustainable data centers. *Journal of Network and Computer Applications*, 83:155–168.

[51] Xiao, Z., Song, W., Chen, Q., et al.(2013). Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Trans. Parallel Distrib. Syst.*, 24(6):1107–1117.

[52] Zareian, S., Veleda, R., Litoiu, M., Shtern, M., Ghanbari, H., and Garg, M.(2015). K-feed-a data-oriented approach to application performance management in cloud. In *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, pages 1045–1048. IEEE.

Edited by: Anand Nayyar
Received: Feb 4, 2019
Accepted: Mar 12, 2019

DOI 10.12694/scpe.v20i2.1513
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 223–235. http://www.scpe.org © 2019 SCPE

ONTOLOGY DRIVEN SOCIAL BIG DATA ANALYTICS FOR FOG ENABLED SENTIC-SOCIAL GOVERNANCE

AKSHI KUMAR[*] AND ABHILASHA SHARMA[†]

Abstract. Conventional e-government has many practical infrastructure development and implementation challenges. The recent surge of SMAC (Social media, Mobile, Analytics, Cloud) technologies re-defines the e-governance ecosystem. Cloud-based e-governance has numerous operational challenges which range from development to implementation. Moreover, the contemplation and vocalization of public opinion about any government initiative is quintessential to be cognizant of how citizens perceives and get benefitted from cloud/fog enabled governance. This research puts forward a semantic knowledge model for investigating public opinion towards adaption of fog enabled services for governance and comprehending the significance of two s-components (sentic and social) in aforesaid structure that specifically visualize fog enabled Sentic-Social Governance. The results using conventional TF-IDF (Term Frequency-Inverse Document Frequency) feature extraction are empirically compared with ontology driven TF-IDF feature extraction to find the best opinion mining model with optimal accuracy. The results depict that the implementation of ontology driven opinion mining for feature extraction in polarity classification outperforms the traditional TF-IDF method validated over baseline supervised learning algorithms. An average of 7.3% improvement in accuracy and approximately 38% reduction in features has been reported.

Key words: Fog Computing, Government Intelligence Cloud , Opinion Mining, Feature Selection, Ontology

AMS subject classifications. 68M14, 97R50, 91D30

1. Introduction.

Government-to-Citizen (G2C) is a concept [1] that speaks about the connectivity between public administration organizations and citizens of a country. The relationship specifies ICT (Information and communication technologies) based solution that streamlines the interaction and association between governing bodies and citizens. The ultimate goal of G2C [2] governance is to serve the society by offering various ICT services with the use of cutting edge technologies in a more productive and profitable way. It also plays up to increase people participation in governance both in terms of quantity and quality. G2C interactions empower citizens by apprising them with the government policies, practices, rules, regulations, strategies and services. Several countries have developed their respective framework of G2C technology for enhancing public participation in governmental proceedings.

Digital India [3] is a flagship programme launched by Government of India to transform the nation into a digitally empowered society and knowledge economy. The objective is to explore innovative ideas and practical solutions by exploiting digital technologies such as cloud computing, mobile applications, Internet of Things (IoT) etc. It is a "citizen-centric" campaign that concentrates on three prime parameters [4]: digital infrastructure as a utility to every citizen, governance and services on demand, and digital empowerment of citizens. The initiative includes coupling of multiple government ministries and departments with several thoughts and ideas transforming into a single comprehensive vision but its practical implementation is very demanding and challenging. Large development and implementation cost, lack of internet accessibility, lack of computing skills among citizens, insufficient technology requirements, requisite of email addresses, lack of privacy are certain issues [5] that makes it difficult for governing bodies to reach out citizens. The nine growth pillars of Digital India [6] that contribute towards the economic and electronic dissemination of government information in public are represented in figure 1.1. Several policies, schemes, campaigns, and initiatives are undertaken by different government officials with the purpose to reinforce these growth pillars. Thus, the holistic development of these growth pillars is the need of the hour.

Electronic governance or e-governance [7], an application of ICT and framework for G2C communication, plays a significant role in information exchange of government services to citizens. It is an electronic execution of governance [8] that facilitates a simple, rapid, efficient and highly transparent process of information broadcasting, delivery of government services in order to promote good governance. It signifies the reformation of government with the implementation of technology in government processes and functions. This technological revolution in governance can modernize the modus operandi of any society.

[*]Department of Computer Science & Engineering, Delhi Technological University, Delhi-42, India (akshi.kumar@gmail.com).
[†]Department of Computer Science & Engineering, Delhi Technological University, Delhi-42, India (abhi16.sharma@gmail.com)

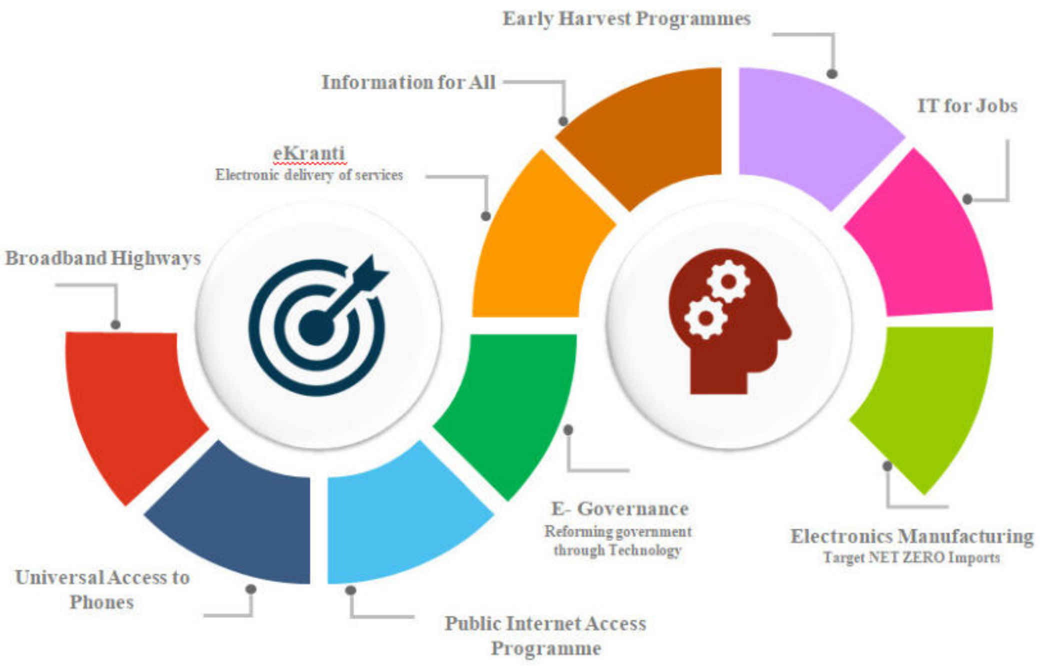

Early Harvest Programmes
Information for All
IT for Jobs
eKranti
Electronic delivery of services
Broadband Highways
E- Governance
Reforming government through Technology
Electronics Manufacturing
Target NET ZERO Imports
Universal Access to Phones
Public Internet Access Programme

FIG. 1.1. *Nine Growth Pillars of Digital India [6]*

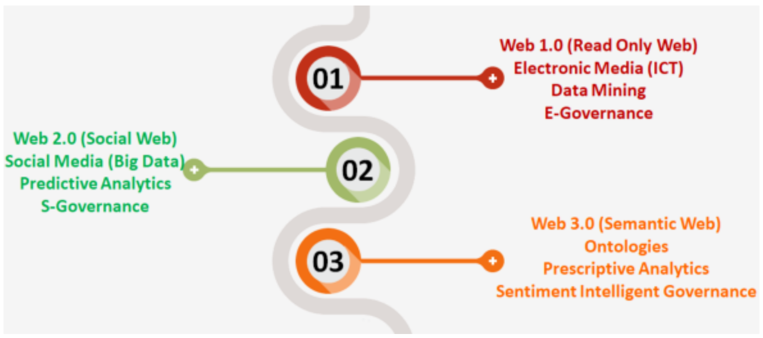


FIG. 1.2. *Evolution of Media, Analytics and Governance with Web*

Digital information and communication technologies have come forth as wave maker that can transform an entire economy, build up smart citizens and realize a well established form of socio-political organizations across globe. In view of this, convergence of four technologies namely Social media, Mobile, Analytics, Cloud (SMAC paradigm) [9] is the driving force of governance ecosystem. A voluminous amount of data (big data) has been generated through various sources of *social media* and *mobile* applications that require optimized techniques for data *analytics*. Cloud computing characterizes the on-demand delivery of computing services over the internet. This virtualization technology offers self-service capability, scalability, flexibility, and affordability. The evolving web has changed the traditional model of governance into digital governance model turning the [10] paradigm shift from e-governance to s-governance (social governance). It also turns the corners of social media as well as data analytics. The evolution timeline of all four factors has been depicted in figure 1.2.

Cloud computing [11] makes a big shift from the conventional methods of governance due to various reasons such as infrastructure cost reduction, quick provision of computing resources, scalability in terms of delivering right amount of IT services, improves productivity and performance, procuring more secure environment and much more. It proffers a simple way to access servers, storage, databases and a broad set of application services

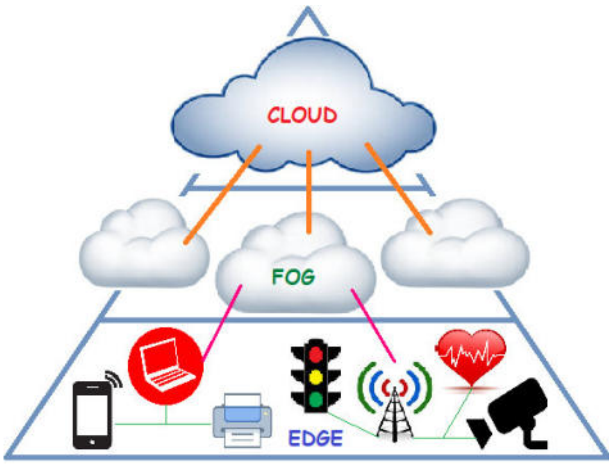


Fig. 1.3. *Big Data Processing Layer Stack [13]*

over the Internet. Cloud computing, the name is so because information accessed is found in ”the cloud” and provides user a anywhere, anytime access. Fog computing, dew computing, cloudlet and edge computing are extended variants that lies at different levels in hierarchy of cloud computing architecture. Figure 1.3 represents how the [12] storage, accessibility and maintenance of huge amount of data i.e. big data available over web is being maintained and processed by cloud, fog and edge computing.

Fog computing brings the benefits, services and power of cloud to the edge of the network. It has closer proximity to end users i.e. performing short term data analytics at the edge and larger geographical distribution. It acts as an interface between cloud and edge device layers [14] deciding what data needs to be pushed to cloud and what needs to be analyzed locally at the edge. The goal of fogging is to improve efficiency, reducing data transportation to cloud, security and compliance. All these factors put accent on the adoption of fog layered architecture for adapting governance infrastructure. The role and need of cloud/fog based governance has been studied across literature. Various literary resources are available online which discuss the practical and potential application of cloud in the governance ecosystem. Certainly, fog enabled governance is the non-trivial buzzword within the contemporary paradigm for government intelligence. *MeghRaj: the national cloud* is one such the government intelligence cloud initiative [15] launched by Government of India.

The contemplation and vocalization of public opinion about any governmental processes or practices is quintessential as they are entitled to enjoy all the legal rights and privileges granted by the government. It is equally important to be cognizant of how citizens perceives and get benefitted from cloud/fog enabled governance. Consequently, addition of one more s-factor (sentiment based) in s-governance reforms the existing model into a *sentic-social* governance (S-governance) with a view to make it more sustainable. Opinion mining facilitates this implementation and formalization of the S-governance model. It is referred to as the computational study [16,17] to extract and analyze public opinion/sentiment about any topic, subject, event or entity for better decision making process by applying various intelligent techniques over a large volume of user generated data (social big data) over web. The use of different intelligent learning techniques such as machine learning, lexicon-based, hybrid and concept based has been reported across pertinent literature studies [18] within the domain. As an attempt to comprehend public opinion on the induction and adoption of cloud-based governance model, this research put forwards an optimized predictive learning model based on real-time data.

The vital sub-task of the polarity classification (opinion polarity: positive, negative, neutral) process is feature extraction, which converts the input data (unstructured textual data indicative of opinion), into an array of representative features. Commonly, the feature extraction task is done using intrinsic filtering methods which are fast, classifier-independent methods that rank features according to predetermined numerical functions based on the measure of the importance of the terms. A variety of scoring functions such as, tf-idf, chi-square, mutual information, information gain, cross-entropy etc., have been used as statistical measures to pick features with the highest scores [19].The accuracy of the classifier strongly depends on the selection of high quality data features that is the training dataset. Moreover, the training sets are typically prepared manually. Past literature conforms that an optimal feature selection [20] improves the classifier performance (in terms of speed, predictive power and simplicity of the model), reduces dimensionality, removes noise and helps visualizing the data for model selection. In feature selection the features are kept intact and n best features are chosen among them, removing the redundant and co-linear features. This sub-task of selecting the relevant subset of features and discarding the non-essential ones is computationally challenging and expensive task. Motivated by these issues, in this research we propose a semantic knowledge based polarity classification process.

An ontology [21,22] which is specification of conceptualization is utilized as a filtering method for finding important as well as hidden features. Ontologies are primarily tools which can drive the feature engineering process by:

- Structuring semantic information as concepts, properties, instances and hierarchies for feature identification.
- Extracting explicit features to build the feature space using relationship between concepts.
- Uncover important features using concept hierarchy defined by the ontology.

Hence, to optimize the feature space without sacrificing remarkable classification accuracy in this work, we put forward an intelligent ontology based data analytics solution for opinion prediction in social big data concerning fog enabled governance. The proffered solution is put to test for the sentiment classification tasks on tweets pertaining to *"Meghraj: The National Cloud"*, a government intelligence cloud initiative launched by Government of India. The conventional classification process is done using TF-IDF (Term Frequency-Inverse Document Frequency) feature extraction method on the cleaned dataset. Five supervised machine learning classifiers namely Nave Bayesian (NB), Support Vector Machines (SVM), Multilayer Perceptron (MLP), k-Nearest Neighbour (k-NN), and Decision Tree (DT) are empirically compared. Ontology based feature optimization is then performed to semantically analyze the concept and make rise in reusability, interoperability, knowledge acquisition and its verification. Thus, the contribution of this research is to build an optimal opinion mining model as follows:

- To implement five supervised learning algorithms to classify opinion polarity using tf-idf feature extraction: NB, DT, SVM, kNN & MLP
- To build a domain ontology for optimal feature extraction: Domain Ontology for Meghraj (DOM)
- To implement five supervised learning algorithms using ontology guided feature extraction method: TF-IDF on ontological features
- Performance analysis on the basis of efficacy measures

The objective of this paper is to implement and evaluate an opinion mining model for analysing public opinion on government cloud initiative. It is characterized as a semantic knowledge (ontology) based model of fog enabled services offered by government and consequently comprehends the significance of two s-components (sentic and social) within the Fog enabled Sentic-Social Governance.

The subsequent sections are lined up as follows: Section 2 discusses the background work related to fog enabled services in governance and ontology driven opinion mining. Section 3 abstracts the information about Meghraj as a techno typhoon for government prosperity and its scope in the landscape of Indian governance. Section 4 explicates the proffered model for ontology driven opinion prediction of fog enabled governance It also illustrates the dataset details and implementation process. Section 5 determines the opinion classification of the chosen concept as per tweets polarity. It substantiates the ontological model by comparing classifier performance for optimal feature extraction with baseline supervised learning techniques to analyse public opinion about the program. At last, section 6 sums up the inferences drawn from the results and discusses the future work as an open scope of this research work.

2. Related Literature Review of Cloud/Fog enabled Services for Digital Governance. Conventional e-governance faces many challenges in terms of cost, software and hardware requirements, network, security, business & policy adoption and implementation etc. Recent years have shown cloud computing as a technology to solve these problems. Pokharel et al. [23] proposed cloud computing as the future solution for e-governance. Mukherjee et al. [24] put forward a future framework for e-governance based on cloud computing consisting of three layers. Sharma et al. [25] enlisted the applications of cloud computing in e-governance. Work done by authors Cellary et al. [26] discussed about cloud computing and service-oriented architecture for e-governance. In the research done by Yeh et al. [27], cloud computing has been used in e-governance to change its function towards service, push forward the green technology and promote industrial upgrading. Author Rastogi [28] proposed a model-based framework to implement cloud computing in e-governance. Tripathi & Parihar in 2011, Alshomrani & Qamar in 2013 [29,30] analysed cloud computing and its applications in e-governance and explained how cloud may lead to cost savings.

The term, *Opinion Mining*, also known as sentiment analysis or sentiment mining, was initially witnessed in the published work by Dave et al. [31] in 2003 and since then both primary and secondary studies have been reported across pertinent literature [18,32,33]. As discussed, various techniques have been used to perform the task of opinion mining namely: machine learning, concept based, lexicon based, and hybrid. Pertinent literature reveals [18] various application areas of opinion mining namely, business intelligence, information and security analysis, market intelligence, sub component technology, government intelligence, smart society services etc. Government intelligence (GI) has been the least explored application area with only few relevant studies done within the domain, which include GI using concept based opinion mining, lexicon-based opinion mining , hybrid approach , and opinion mining using machine learning. Recently, Kumar & Sharma [34, 35] proposed few models/frameworks for opinion mining in the different application area of digital governance. To the best of our knowledge, no related work on mining public opinion to understand adoption and acceptance process of a cloud based e-governance initiative has been done.

Feature extraction primarily transforms raw data into features suitable for modelling. For example textual features include n-grams, word2vec and TF-IDF etc [36]. The training set is high-dimensional and the classifiers accuracy strongly depends on the quality of hand-crafted features. To deal with the dimensionality of training feature sets for improved classifier performance, various techniques have been successfully applied. Feature selection [37] and dimensionality reduction [38] approaches have been used to select features with the highest "importance"/influence on the target variable, from a set of existing features. The higher the number of features, more challenging and computationally expensive it gets to visualize the training set. Ontology guided feature extraction can thus be used as a feasible and to optimize the feature space representative of the training dataset. Domain ontologies can be used in selection of features for improved feature-based opinion mining. Pealver-Martnez et al. [39] have discussed the concept of ontological feature based opinion mining based on recent studies. The two major dimensions were given attention namely, opinion classification and feature based opinion mining. They reviewed the improvement process of feature level opinion mining by incorporating the concept of ontology. Also, a proposed approach for ontology-supported polarity mining (OSPM) has been examined for the sake of enhancing the process of opinion classification.

3. *Meghraj: The National Cloud* - An Initiative by Indian Government for Proliferation of Fog/Cloud in Governance. In May 2016, National e-Governance Plan (NeGP) [40] was initiated by Government of India for better accessibility of all government services to citizens. The vision is to access these services in the citizens own locality in a cost-effective manner via common service delivery outlets. As a step towards this different states have placed and setup their respective ICT infrastructure. The deployment and installation of application software has been done through assistance and funding provided by central government. Although, the whole process has been outsourced but still the entire endeavour was strapped for time due to inadequate application development initiatives, detailed and exhaustive procurement process, shortage of in-house experts for managing large procurements etc. The government was looking forward for a common shared platform to gain momentum in the implementation process. This infused the need for the adoption of a government private cloud environment that would expedite the ICT enabled service enrichment process with an affordable cost in and among various government departments at state or central level. Figure 3.1 represents the hierarchy of adoption of cloud [41] in Indian governmental structure.

Meghraj, the GI cloud initiative [15,42] was launched by Government of India under the Department of Electronics and Information Technology (DeitY), Ministry of Communications and Information Technology. The major objective(s) is to (a) accelerate the development and deployment of eGov applications (b) ensure optimal use of resources and infrastructure (c) make standardized and certified applications available (d) replicate successful applications in order to reduce effort, time and cost across states and much more. The functional flow of GI cloud system [42] is illustrated in figure 5. The architecture of GI cloud follows specific standards, guidelines and a set of protocols issued by Government of India. The task of GI cloud services directory is

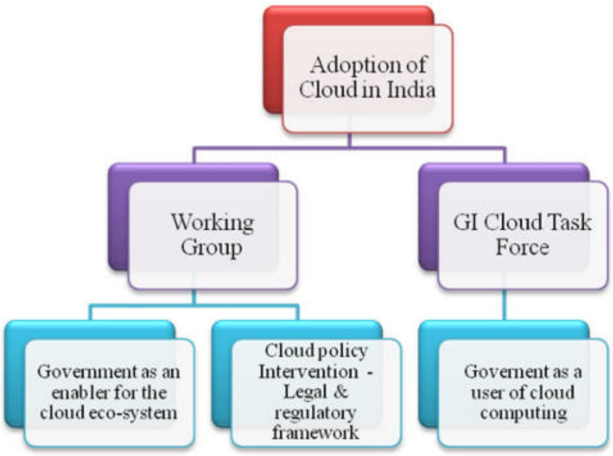

Adoption of Cloud in India
Working Group
GI Cloud Task Force
Government as an enabler for the cloud eco-system
Cloud policy Intervention - Legal & regulatory framework
Govenment as a user of cloud computing

Fig. 3.1. *The National Cloud Initiative [41]*

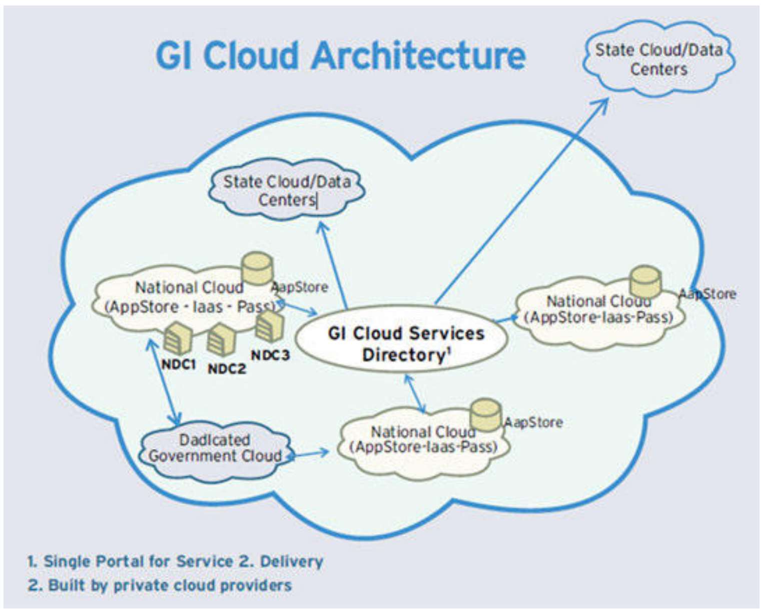

GI Cloud Architecture
State Cloud/Data Centers
State Cloud/Data Centers
National Cloud (AppStore - Iaas - Pass)
AapStore
NDC1
NDC2
NDC3
GI Cloud Services Directory¹
National Cloud (AppStore-Iaas-Pass)
AapStore
Dadlcated Government Cloud
National Cloud (AppStore-Iaas-Pass)
AapStore
1. Single Portal for Service 2. Delivery
2. Built by private cloud providers

Fig. 3.2. *GI Cloud System Flow [42]*

to publish the services proffered by GI cloud. It comprises of discrete and multiple private cloud computing environments at national and state levels with a view to provide high level blueprint of ICT enabled government functions. The architecture allows the state data centres (SDCs) or state clouds of different states to associate with GI cloud either by acting as independent cloud environments or by lending their IT Infrastructure as part of GI Cloud but these states can have their own data centres as well that exists outside the GI cloud environment. All such states are motivated to jump on GI cloud resources as and when they exhausted their own. SDCs of 21 states have been made operational and running numerous applications such as e-Procurement, Bhoomi, commercial tax, mandi board etc. SDCs of 4 states are in advanced stage of implementation and cloud adoption.

National Cloud was launched in February 2014 under Meghraj initiative and implemented by National Informatics Centre (NIC) [43] offers various services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Storage as a Service (STaaS). Table 3.1 enlists various potential

TABLE 3.1
Key drivers and issues of GI cloud

Benefit(s)	**Hazard(s)**
Optimal usage of available resources and infrastructure	Application design approaches and cloud standards
Rapid deployment and Reusability	Intensive change management and lack of skilled resources
Security, Scalability, Manageability and Maintainability	Vendor lock-in and data location
Reduced time, effort and cost in managing technology	Loss of portability and control
Increased standardisation and user mobility	Licensing and funding model

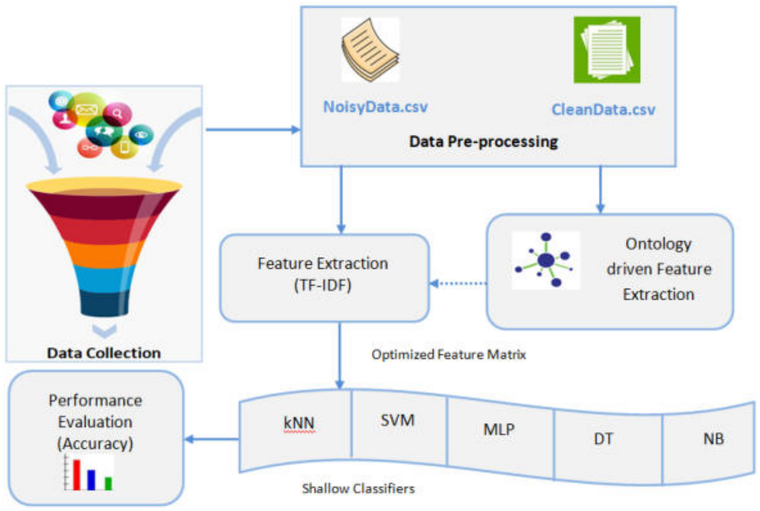


FIG. 4.1. *System Architecture of Ontology based Opinion Prediction Model*

benefits and risks of GI cloud.

There are three adoption phases for the establishment of GI cloud [44] namely, (i) Strategy, policy and guidelines establishment (ii) Implementation and (iii) Monitoring, management and ongoing improvement. The first phase focuses over the formation of practices, strategies, rules and guidelines in order to prepare a development plan. Implementation phase executes the prepared program in a well planned manner and the last phase monitors the present workings, evaluates the shortfalls and works towards continuous improvements. In this paper, the evaluation of GI cloud initiative has been done by capturing opinionated response of citizens for predictive analytics. Domain Ontology has been built to model the feature space and it is optimized using the conventional statistical feature extraction TF-IDF filter. The proposed framework is proposes in the following section.

4. Ontology based Opinion Predictive Analytics.

The proposed framework comprises of four phases namely, data collection, data pre-processing, ontology driven optimal feature extraction and classification. Figure 4.1 depicts the system architecture of ontology based opinion prediction model.

4.1. Data Acquisition.

Collection of data is the foremost step in the process of opinion mining. Various social media portals are available for capturing real-time data of public perception about any topic, subject, event or area of concern. Analysis of real-time data sets relevant to any phenomena provides more accurate results or evaluation of the system and hence improves the overall performance. Twitter, being the most popular micro blogging site [45] has been used in this research work to capture the public sentiments about Meghraj. It is one of the most prominent social media platforms exercised by government for facilitating

Table 4.1
Weekly record of tweets collected

Duration	**Tweet Count**	**Duration**	**Tweet Count**
Week 1	192	Week 4	328
Week 2	194	Week 5	160
Week 3	402	Week 6	140

Table 4.2
Tweets Distribution based on Polarity

Week	**N**	**Nu**	**P**	**Total**
Week 1	29	14	149	192
Week 2	23	16	155	194
Week 3	19	76	307	402
Week 4	11	65	252	328
Week 5	4	79	77	160
Week 6	1	85	54	140
Total	87	335	994	1416

direct government citizen interaction. A huge volume of users share their views, ideas, suggestions and update themselves with the information posted on twitter. The diversity of opinion in various public communities attributed to their regional, cultural, social, economical and educational backgrounds is considered in order to obtain a realistic view of the system. Provision of various APIs (Application Program Interfaces) accelerate the process of tweets extraction over a specific topic using #(hash tag) leading with the topic name(#topicname). The process of tweet extraction has been done by running scripts using an application developed in python. The search query comprises and executes with various hash tag words such as #meghraj, #gicloud, #meghrajcloud, #nationalcloud. A .csv file has been created with the tweets relevant to #topicname as a result of search query. Tweets associated with hashtags for national cloud have been collected near the launch date of program for a duration 6 weeks. A count of 1416 tweets have been gathered in this duration in order to figure out the public inclination about the program. Table 4.1 enlists the weekly status of tweets collected.

The tweets are classified into three polarity categories namely, positive, negative and neutral. Table 4.2 reflects the tweets distribution based on their opinion polarity.

For the selected duration, MeghRaj able to capture a count of 1416 tweets that consists of 950 positive tweets, 335 neutral tweets and 126 negative tweets. A graphical depiction of the aforesaid stats of weekly polarity distribution of tweets is represented in figure 4.2. Approximately 70% of tweets were positive, whereas the percentage of neutral and negative tweets was 23.6% and 6.1% respectively. Deprecation in the count of positive tweets has been reported after week 3 whereas an average rise of tweet count can be observed in neutral tweets after week 2. However, there is a continuous deterioration in the frequency of negative tweets. Week 3 was reported as a critical week being the maximum number of positive tweets whereas week 6 reflects the maximum number of neutral tweets. The percentage of negative tweets was high in the initial phases due to certain technical limitations in implementing procedure of this scheme. Also, the gradual increase in the neutral tweets by the time is due to the rapid occurrence of informational tweets posted by government, allied agencies, and media or civic. This information was all about the campaign features, functionality, latest updates and future processing's and hence making the polarity of tweet neutral. Python packages like Scipy, nltk, Numpy, Scikit-learn etc., machine learning and python scripts (version 3.6) have been used for implementation.

4.2. Pre-processing of data. The outcome of data collection is a .csv file that contains data with a lot of noise. The second phase, data pre-processing, is another essential step performed in order to transform the existing file into a new one for feature selection by cleansing the collected data. This conversion of NoisyData.csv file to CleanData.csv file and the pre-processing procedure includes following steps:

- Removing tweets replication, stop words, number in tweets with placeholders, mentions etc.
- Replacement of URLs.
- Eliminating figures, special characters such as , #.
- Natural Language tool-Kit (NLTK) [46,47] for tokenization.

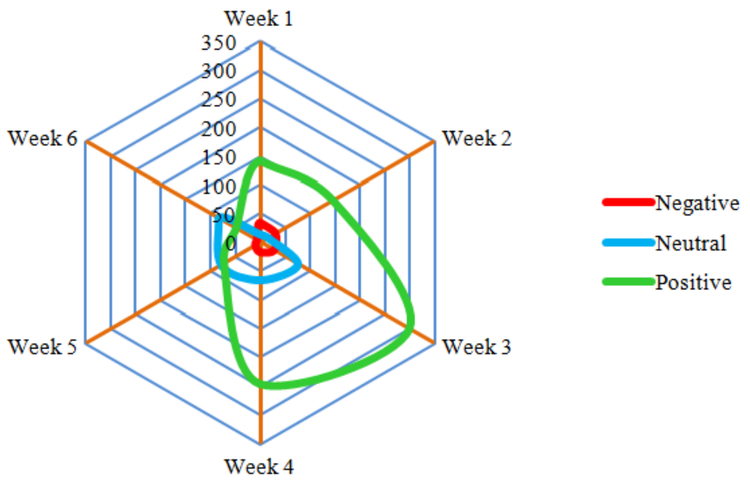


Fig. 4.2. *Polarity distribution of tweets on weekly basis for Meghraj*

- Porter's stemmer [48,49] for stemming to the root word.
- Removal of non-ASCII English character.

This cleansed data is then utilised for feature extraction. Some keywords for the selected attributes are: projects, e-services, IT giants, operational, future computing, cloud solution, departments, deployment, infrastructure, server, administration etc.

4.3. Feature Extraction. Feature extraction is one of the critical and complex tasks in opinion mining. The objective is to recognize the entity (person, service or an object) that is being referred in opinion. The automation of the process of feature identification in opinion analysis with the use of NLP (Natural language processing) techniques makes it harder to comprehend. In this paper, the two methods used for feature extraction are as follows:

- **Conventional Feature Extraction (TF-IDF based)**
  TF-IDF stands [50,51] for Term Frequency - Inverse Document Frequency. It is a weight statistically measured to evaluate the importance of a word to a document in a corpus. The importance of a word increases as its frequency increases in a document but is offset by the frequency of word in corpus. The Term Frequency, TF(t, d) simply counts the frequency of a term in a document as follows:

$$TF(t,d) = \left(\frac{No.\ of\ times\ term\ t\ appears\ in\ a\ document\ d}{Total\ no.\ of\ terms\ in\ the\ document} \right)$$

  The Inverse Document Frequency, IDF (t, D) checks whether the word is rare or common in the corpus so as to measure how much information is provided by a specific word. It is calculated as follows:

$$IDF(t,D) = log_e \left(\frac{Total\ no.\ of\ documents}{No.\ of\ documents\ with\ term\ t\ in\ it} \right).$$

  Thus, TF-IDF is calculated as:

$$TF - IDF(t,d,D) = TF(t,d) * IDF(w,D)$$

  where t denotes the terms; d denotes each document and D denotes the collection of documents.

- **Optimal Feature Extraction (Ontology driven TF-IDF)**
  Ontology is specifically defined as [22,52] a conceptual reference model that describes the semantics of a system or domain. It represents the relationship between concepts; both in human comprehensible

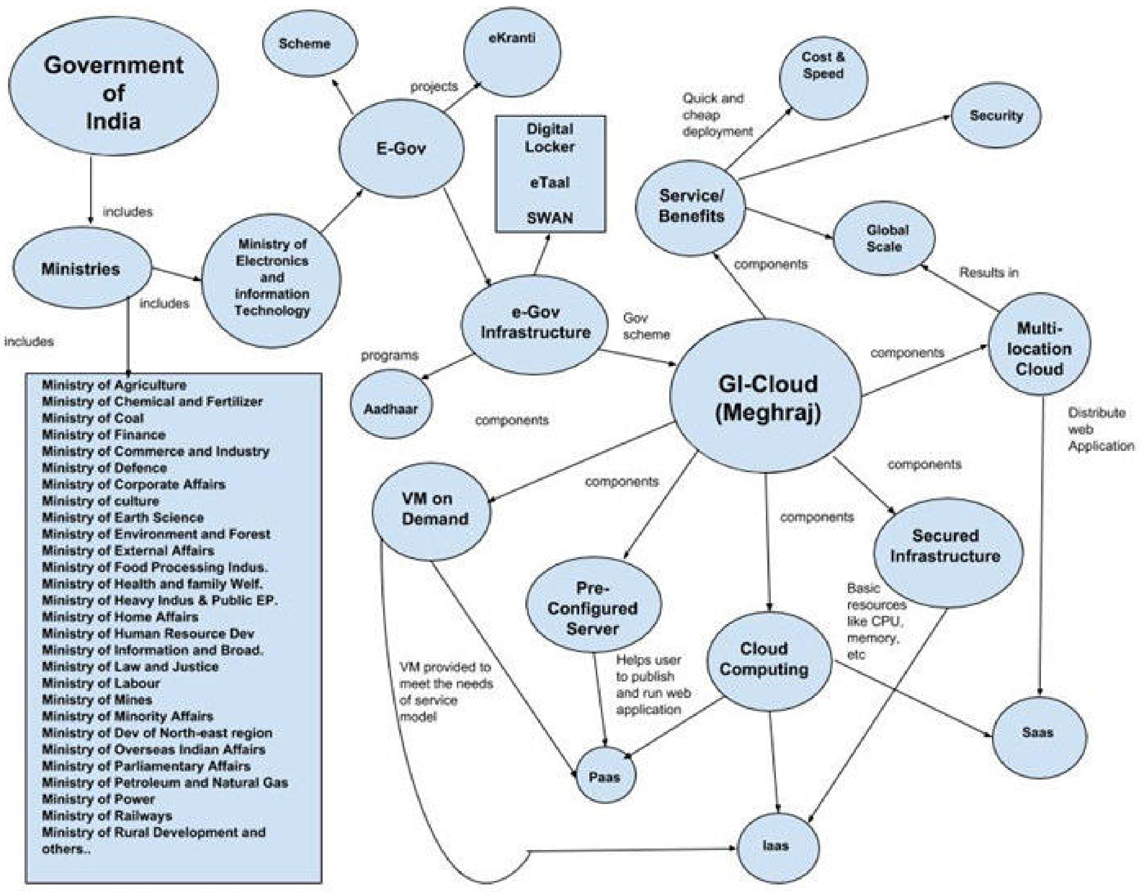

Government of India
includes
Ministries
includes
includes
Ministry of Electronics and information Technology
Ministry of Agriculture
Ministry of Chemical and Fertilizer
Ministry of Coal
Ministry of Finance
Ministry of Commerce and Industry
Ministry of Defence
Ministry of Corporate Affairs
Ministry of culture
Ministry of Earth Science
Ministry of Environment and Forest
Ministry of External Affairs
Ministry of Food Processing Indus.
Ministry of Health and family Welf.
Ministry of Heavy Indus & Public EP.
Ministry of Home Affairs
Ministry of Human Resource Dev
Ministry of Information and Broad.
Ministry of Law and Justice
Ministry of Labour
Ministry of Mines
Ministry of Minority Affairs
Ministry of Dev of North-east region
Ministry of Overseas Indian Affairs
Ministry of Parliamentary Affairs
Ministry of Petroleum and Natural Gas
Ministry of Power
Ministry of Railways
Ministry of Rural Development and others..
Scheme
eKranti
projects
E-Gov
Digital Locker
eTaal
SWAN
e-Gov Infrastructure
Gov scheme
programs
Aadhaar
components
Service/ Benefits
Quick and cheap deployment
Cost & Speed
Security
Global Scale
Results in
components
Multi-location Cloud
components
GI-Cloud (Meghraj)
Distribute web Application
components
components
components
VM on Demand
Pre-Configured Server
Secured Infrastructure
Cloud Computing
Basic resources like CPU, memory, etc
Helps user to publish and run web application
VM provided to meet the needs of service model
Paas
Saas
Iaas

Fig. 4.3. *Domain Ontology of Meghraj (DOM): An initiative towards fog enabled governance*

and machine processable manner. It represents a concept or categories of a particular subject area that exhibits the characteristics and relationship between them. The ontological representation of the concept of Meghraj (DOM - an initiative taken by government of India to provide fog enabled services for a sentic-social facet of governance) is illustrated in figure 4.3 that represents the existing entities, how they are grouped and related in a hierarchy and sub classified based on their similitude & dissimilarity. The ontology represents various ministries that come under the umbrella of government of India. Among those, Ministry of Electronics and Information Technology provides the facility of electronic delivery of services i.e. e-Governance to facilitate citizens. Various components that comprise e-Governance are policies; projects (such as e-Kranti) and infrastructure (include programs such as aadhar, services such as digital locker, eTaal, SWAN etc., schemes). Meghraj, the first Indian GI cloud is one of the prime governmental schemes enhancing the e-Governance infrastructure. Different sub components that combines to form Meghraj are cloud computing, secured infrastructure, multi-location cloud, pre-configured server and VM (virtual machine) on demand. The figure depicts the sub components of cloud computing, elements of services offered by MeghRaj and the inter-relationship between these components along with remaining elements of ontology.

4.4. Opinion Polarity Classification. In this phase, polarity of opinion is classified into three pre-defined categories namely, positive, negative and neutral. The optimal feature set generated in the earlier step are used to build the training and testing sets of the classifier. In this paper, five supervised learning based classifiers namely, Support Vector Machine (SVM), Decision Trees (DT), Nave Bayesian (NB), k-Nearest Neighbour (k-NN) and Multi-Layer Perceptron (MLP) have been implemented. All the machine learning techniques are

TABLE 4.3
Accuracy obtained using conventional and optimal feature extraction

Technique Used	Conventional Approach (TF-IDF) Accuracy (%)	Optimal Approach (Ontology + TF-IDF) Accuracy (%)	Increase In Accuracy (%)
NB	82.2	91.9	9.7
DT	89.8	96.6	6.8
SVM	92.6	98.5	5.9
k-NN	91.3	97.7	6.4
MLP	90.2	98.3	8.1

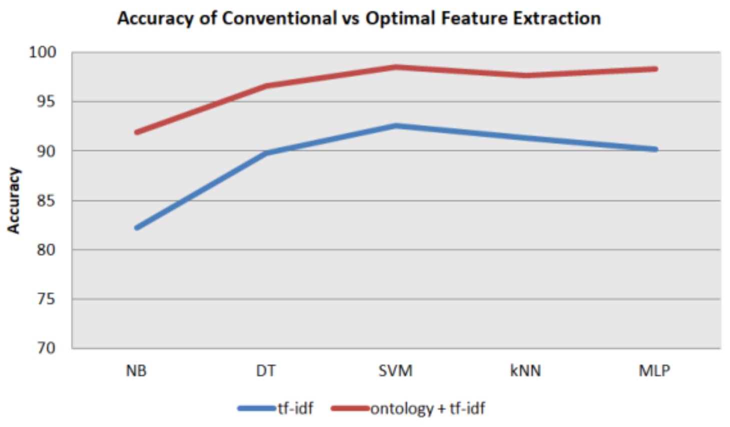


FIG. 4.4. *Accuracy obtained using conventional and optimal approach*

described in detail [34,53] across relevant literature.

4.5. Results and Discussions. This section discusses about the performance of classifiers for selected machine learning techniques (NB, DT, SVM, kNN & MLP) based on accuracy. It also compares and contrasts the difference between conventional feature extraction (using TF-IDF) and optimal feature extraction (ontological TF-IDF) techniques. The results along with percentage increase in accuracy have been listed in Table 4.3.

Result states that the best accuracy with conventional feature extraction over collected data set has been obtained by Support Vector Machine (SVM) algorithm with 92.6%. Next is k-Nearest Neighbour (kNN) with a classification accuracy of 91.3% followed by Multi-Layer Perceptron (MLP) with 90.2% and Decision Trees (DT) with 89.8. Amongst all, Naive Bayesian (NB) attained the lowest accuracy of 82.2%. The best accuracy using ontology driven feature extraction is achieved by SVM, i.e. 98.5% followed by MLP with 98.3%. k-NN and DT have occupied the next level with 97.7% and 96.6% respectively. Amongst all, NB showed the lowest accuracy of around 91.9%. The graphical comparison is represented in figure 4.4.

The maximum accuracy gain was obtained by NB (9.7%) followed by MLP (8.1%) while SVM, k-NN showed an appreciable gain in accuracy (approximately 6%). The average accuracy gain is of 7.38%.

The count of features selected in both the approaches is listed in Table 4.4. In conventional approach all the classification algorithms used the same number of features (866). After applying ontology for feature extraction the minimum number of features selected was 507 (SVM) which is 58.5% selection and maximum was 574 (NB) which is 66.2% selection. The Table 4.4 reflects an average of 62.02% features was selected.

5. Conclusion. User-generated big data from online social portals is a goldmine for extracting and analyzing stance and opinion. This knowledge discovery framework within the unstructured web data setting defines a novel socially aware and sentiment driven governance. Based on this, the work proffered in this research made two primary contributions to evaluate the response of citizens towards government initiatives, schemes or policies. Firstly, the role of social media analytics in government intelligence was investigated. Public opinion in tweets for the national cloud initiative of Government of India was examined. This opinionated information

Table 4.4
Number of features obtained using conventional and optimal approach

Technique	Conventional Approach (TF-IDF) #Features (%)	Optimal Approach (Ontology + TF-IDF) #Features (%)	Features Selected (%)
NB	866	574	66.2
DT	866	526	60.7
SVM	866	507	58.5
k-NN	866	547	63.1
MLP	866	534	61.6

can be an imperative phase in a government stratagem for public policy evaluation. Secondly, the learning for the predictive model of opinion mining was driven using optimal semantics-driven feature space generation. Domain ontology for Meghraj (DOM) was built and used for feature extraction with the intrinsic TF-IDF filtering method. The results demonstrated an average accuracy gain of 7.3%. SVM outperformed all the other classifiers (NB, DT, k-NN, MLP) for both conventional and ontology-drive model training. The use of ontology also built an optimal feature space automatically with only 62% of the features selected.

REFERENCES

[1] E-governance, `https://en.wikipedia.org/wiki/E-governance`
[2] What is Government-to-Citizen (G2C), `https://www.igi-global.com/dictionary/government-to-citizen-g2c/12392`
[3] Introduction of Digital India, `https://digitalindia.gov.in/content/introduction`
[4] Vision of Digital India, `https://digitalindia.gov.in/content/vision-and-vision-areas`
[5] Digital India, `https://www.mygov.in/group/digital-india/`
[6] How Digital India will be realized: Pillars of Digital India., `https://digitalindia.gov.in/content/programme-pillars`
[7] Dawes, S. S, *The evolution and continuing challenges of egovernance*, Public Administration Review., 68 (2008), S86-S102.
[8] Lee-Geiller, S. and Lee, T. D., *Using government websites to enhance democratic E-governance: A conceptual model for evaluation.*, Government Information Quarterly., (2019)
[9] SMAC - The paradigm shift - Creating future of the enterprise., `https://home.kpmg/in/en/home/insights/2014/09/smac-theparadigmshift.html`
[10] Kumar, A. and Sharma, A., *Paradigm shifts from e-governance to s-governance*, The Human Element of Big Data: Issues, Analytics, and Performance., 213(2016)
[11] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., et al., *A view of cloud computing.*, Communications of the ACM., 53(4) (2010), 50-58
[12] Vaquero, L. M. and Rodero-Merino, L., *Finding your way in the fog: Towards a comprehensive definition of fog computing.*, ACM SIGCOMM Computer Communication Review, 44(5)(2014), 27-32.
[13] Fog computing vs edge computing., `https://erpinnews.com/fog-computing-vs-edge-computing`
[14] Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K. and Buyya, R, *iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments.*, Software: Practice and Experience, 47(9) (2017), 1275-1296.
[15] About MeghRaj., `https://cloud.gov.in/about.php`
[16] Liu, B., *Sentiment analysis and opinion mining.*, Synthesis lectures on human language technologies, 5(1) (2012), 1-167.
[17] Pang, B. and Lee, L., *Opinion mining and sentiment analysis.*, Foundations and Trends in Information Retrieval, 2(12) (2008), 1-135.
[18] Kumar, A. and Sharma, A., *Systematic Literature Review on Opinion Mining of Big Data for Government Intelligence.*, Webology, 14(2) (2017).
[19] Chandrashekar, G. and Sahin, F., *A survey on feature selection methods.*, Computers & Electrical Engineering, 40(1) (2014), 16-28.
[20] Kumar, A., Khorwal, R. and Chaudhary, S, *A survey on sentiment analysis using swarm intelligence.*, Indian Journal of Science and Technology, 9(39) (2016).
[21] Gruber, T., *Ontology.*, Encyclopedia of database systems.1963-1965.
[22] Maedche, A. and Staab, S., *Ontology learning for the semantic web.*, IEEE Intelligent systems, 16(2) (2001), 72-79.
[23] Pokharel, M. and Park, J.S., *Cloud computing: future solution for e-governance.*, In Proceedings of the 3rd international conference on Theory and practice of electronic governance (pp. 409-410) (2009). ACM.
[24] Mukherjee, K. and Sahoo, G., *Cloud computing: future solution for e-governance.*, nternational Journal of Computer Applications, 7(7) (2010), pp.31-34.
[25] Sharma, M.K. and Thapliyal, M.P., *G-Cloud(e-Governance in Cloud).*, International Journal Engg. TechSci, 2(2) (2011), pp.134-137.

[26] Cellary, W. and Strykowski, S., *E-government based on cloud computing and service-oriented architecture.*, In Proceedings of the 3rd international conference on Theory and practice of electronic governance (2009)(pp. 5-10). ACM.
[27] Yeh, C., Zhou, Y., Yu, H. and Wang, H., *Analysis of E-government service platform based on cloud computing.*, In Information Science and Engineering (ICISE), 2010 2nd International Conference (2010) (pp. 997-1000). IEEE.
[28] Rastogi, A., *A model based approach to implement cloud computing in e-Governance.*, International Journal of Computer Applications, 9(7) (2010), pp.15-18.
[29] Tripathi, A. and Parihar, B., *E-governance challenges and cloud benefits.*, In Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on (Vol. 1, pp. 351-354). IEEE. 2011, June
[30] Alshomrani, S. and Qamar, S., *Cloud based e-government: benefits and challenges.*, International Journal of Multidisciplinary Sciences and Engineering, 4(6) (2013), pp.1-7.
[31] Dave K, Lawrence S, Pennock DM, *Mining the peanut gallery: Opinion extraction and semantic classification of product reviews.*, Proceedings of the 12th international conference on World Wide Web. ACM. 2003; 519-528.
[32] Kumar, A. and Jaiswal, A, *Systematic literature review of sentiment analysis on Twitter using soft computing techniques.*, Concurrency and Computation: Practice and Experience, e5107.
[33] Gamal, D., Alfonse, M., M El-Horbaty, E. S. and M Salem, A. B., *Analysis of Machine Learning Algorithms for Opinion Mining in Different Domains.*, Machine Learning and Knowledge Extraction, 1(1) (2019), 224-234.
[34] Kumar, A. and Sharma, A., *Socio-Sentic framework for sustainable agricultural governance.*, ustainable Computing: Informatics and Systems.(2018)
[35] Kumar, A. and Sharma, A., *Opinion Mining of Saubhagya Yojna for Digital India.*, In International Conference on Innovative Computing and Communications (2019) (pp. 375-386). Springer, Singapore.
[36] Asghar, M. Z., Khan, A., Ahmad, S. and Kundi, F. M., *A review of feature extraction in sentiment analysis..*, Journal of Basic and Applied Scientific Research, 4(3) (2014), 181-186.
[37] Chandrashekar, G. and Sahin, F., *A survey on feature selection methods.*, Computers & Electrical Engineering, 40(1) (2014), 16-28.
[38] Sorzano, C. O. S., Vargas, J. and Montano, A. P., *A survey of dimensionality reduction techniques.*, arXiv preprint arXiv:1403.2877. (2014)
[39] Pealver-Martnez, I., Valencia-Garca, R. and Garca-Snchez, F., *Ontology-guided approach to feature-based opinion mining.*, In International Conference on Application of Natural Language to Information Systems (2011, June) (pp. 193-200). Springer, Berlin, Heidelberg.
[40] National e-Governance Plan, `https://meity.gov.in/divisions/national-e-governance-plan`
[41] Indian Government Cloud Initiative (GI Cloud - MeghRaj), `https://www.slideshare.net/nasscom/meghraj-nasscom`
[42] GI Cloud (MeghRaj), `https://meity.gov.in/content/gi-cloud-meghraj`
[43] GI Cloud Initiative - Meghraj, `http://vikaspedia.in/e-governance/national-e-governance-plan/gi-cloud-initiative-meghraj`
[44] Government of India's GI Cloud (Meghraj) strategic Direction Paper, `https://meity.gov.in/writereaddata/files/GI-Cloud\%20Strategic\%20Direction\%20Report\%281\%29_0.pdf`
[45] A. Kumar and T.M. Sebastian, *Sentiment analysis on twitter.*, IJCSI International Journal of Computer Science, vol. 9, no. 4, pp. 372-378, Jul. 2012.
[46] Natural Language Toolkit, `https://www.nltk.org/`
[47] Natural Language Toolkit, `https://en.wikipedia.org/wiki/Natural_Language_Toolkit`
[48] Porter Stemmer, `http://people.scs.carleton.ca/~armyunis/projects/KAPI/porter.pdf`
[49] Porter's Algorithm., `http://people.ischool.berkeley.edu/~hearst/irbook/porter.html`
[50] Aizawa, A., *An information-theoretic perspective of tfidf measures.*, Information Processing & Management, 39(1) (2003), 45-65.
[51] Jing, L. P., Huang, H. K. and Shi, H. B., *A Improved feature selection approach TFIDF in text mining.*, In Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on (Vol. 2, pp. 944-946). IEEE.
[52] Cristani, M. and Cuel, R., *A survey on ontology creation methodologies.*, International Journal on Semantic Web and Information Systems (IJSWIS), 1(2) (2005), 49-69.
[53] Kumar, A. and Jaiswal, A., *Empirical Study of twitter and tumblr for sentiment analysis using soft computing techniques.*, In Proceedings of the World Congress on Engineering and Computer Science (2017) (Vol. 1).

Edited by: Anand Nayyar
Received: Feb 18, 2019
Accepted: Mar 20, 2019

DOI 10.12694/scpe.v20i2.1515
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 237–258. http://www.scpe.org © 2019 SCPE

HYBRID BALANCED TASK CLUSTERING ALGORITHM FOR SCIENTIFIC WORKFLOWS IN CLOUD COMPUTING

AVINASH KAUR [*], POOJA GUPTA [†] AND MANPREET SINGH [‡]

Abstract. Scientific Workflow is a composition of both coarse-grained and fine-grained computational tasks displaying varying execution requirements. Large-scale data transfer is involved in scientific workflows, so efficient techniques are required to reduce the makespan of the workflow. Task clustering is an efficient technique used in such a scenario that involves combining multiple tasks with shorter execution time into a single cluster to be executed on a resource. This leads to a reduction of scheduling overheads in scientific workflows and thus improvement of performance. However available task clustering methods involve clustering the tasks horizontally without the consideration of the structure of tasks in a workflow. We propose hybrid balanced task clustering algorithm that uses the parameter of impact factor of workflows along with the structure of workflow. According to this technique, tasks can be considered for clustering either vertically or horizontally based on the value of the impact factor. This minimizes the system overheads and the makespan for execution of a workflow. A simulation based evaluation is performed on real workflows that shows the proposed algorithm is efficient in recommending clusters. It shows improvement of 5-10% in makespan time of workflow depending on the type of workflow used.

Key words: Metadata; Scientific Data Management; Data Sharing; Data Integration; Computer Supported Collaborative Work

AMS subject classifications. 68M20, 91C20

1. Introduction.

In past years of scientific discovery [16], the computational workflow continues to be popular among various disciplines of science, including astronomy[36], physics[11], biology [23, 33], seismology [25], chemistry [45] and others.

A workflow is a series of activities representing business processes or computational science with existing dependencies between them. These dependencies need to be satisfied for the achievement of a goal. Business workflow is a control-flow driven activity including constructs for specifying conditions, paths and also involve human interaction. It implements the company's services or products. Scientific workflow involves large scale data and/or complex computations, therefore, utilizes computing resources and storage capacities[32]. It does not involve control-flow but is data-driven, while there are exceptions such as Askalon [34].

Large amount of data processing is required by scientific workflows that consist of millions of uncommon tasks[6]. For example, the Cybershake workflow [26] containing 800,000 tasks is executed on TeraGrid [41]. These loosely coupled applications in all represent a significant amount of computation and data [11]. Existing applications such as Condo r[20] do not consider overheads in system, fault occurrence or restructuring of a workflow [40, 31].

To improve the scalability of the system and to reduce system overheads, workflow restructuring techniques such as task clustering is introduced [37, 18, 48]. It is a process where smaller tasks are merged into a larger job[37] that is the single execution unit in a workflow management system. After application of task clustering on tasks, the execution units are reduced and in turn, it leads to an increase in application computation, thus reducing system overheads.

However, various existing methods use trial-and-error approach for optimization of workflow structures. For example, Horizontal Clustering (HC) [37] merge different tasks at the same level of workflow horizontally. For a single task, the horizontal level is defined as the largest distance from start task of the Directed Acyclic Graph (DAG) to this task. The user controls the granularity of clustering that is the number of tasks within a cluster and defines either the number of jobs clustered per horizontal level or a number of tasks per clustered job. This kind of techniques ignored the dynamic properties of distributed environments [38].

Many methods are introduced for reducing the system overhead and clustering the tasks either horizontally or vertically but none of the technique employs both kinds of clustering simultaneously. The structure of the workflow plays a significant role in clustering of tasks of a workflow.

[*]Lovely Professional University, Phagwara, India (avinash.14557@lpu.co.in).
[†]Lovely Professional University, Phagwara, India (pooja.19580@lpu.co.in).
[‡]GNDEC, Ludhiana, India (mpreet78@gmail.com).

The work proposes hybrid balanced clustering algorithm which takes into consideration both the structure of workflow and number of jobs available for tasks to be clustered. Also, the tasks with the parent-child relationship are clustered together vertically and tasks without the relationship are considered horizontally. Hence, this helps in reducing systems overheads and faster execution of tasks and minimum wastage of resources. The important points considered in proposed work are

- Minimum tasks overheads: The overheads are reduced to the minimum while the tasks with a single parent-child relationship are clustered into one cluster. Hence, the dependency time of tasks reduces to a significant level.
- Minimum resource wastage: Algorithm ensures that the dependent tasks are provided with the data required as early as possible in order to avoid increase in waiting time and wastage of resources.

The rest of the paper is organized in the following way. The overview of related work is outlined in Section 2. Section 3 describes the workflow management system where the clustering techniques are applied. Section 4 describes the proposed algorithm. Section 5 reports the performance evaluation,results of proposed technique along with available basic clustering techniques. Section 5 ends the manuscript with conclusion and future scope.

2. Literature Review.

2.1. Load imbalance. Load balancing is a topic of significance in the area of distributed computing. To balance the computational load dynamically among different resources, transfer of some jobs is required from one resource to another in a period of time. This is called task reallocation [42]. A reallocation algorithm is proposed for tuning the parallel jobs submitted to the resource manager [42]. The batch scheduler sends submission and cancellation requests. The reallocation algorithm in a multi-cluster environment is presented. To dynamically migrate processes from overloaded computational nodes to less loaded nodes a premptive process migration method is proposed in [47]. However, it exhibits the limitation to maintain balance only with some idle nodes. In our case, we consider more tasks then available compute resources. To handle load imbalance, [15] presented techniques split tasks dynamically and consolidate them to fill idle compute nodes. Similarly, Ying et al. [46] present a load balancing algorithm based on collaborative task clustering. Level based autonomic Workflow-and-Platform Aware(WPA) task clustering technique [35] is proposed that considers the factors of workflow structure and underlying resource set size for task clustering. It aims to reduce the overhead in systems and wastage of resources.

In comparison to the techniques discussed above, our work merges the tasks based on their runtime distribution also considering the data dependencies. The overheads in our work are not introduced during runtime. Also, an approach to dynamically select the order of clustering whether vertical or horizontal is proposed.

2.2. Granularity. In scientific workflows, the techniques to control the granularity of tasks is also addressed. A label-based and level-based clustering approach is proposed by Singh et al. [37]. According to level-based clustering, considering the same horizontal level tasks are clustered. In it user specifies the number of tasks to be considered in a single cluster. In another approach of label-based clustering, the labeling of tasks is accomplished manually by the user. This method is more prone to error due to manual interaction. A task grouping and ungrouping algorithm are proposed, where information of application and resources is not known in advance [13]. Their work does not consider data dependencies but reduces queuing and scheduling overhead. An algorithm is proposed by Muthuvelu et al. [29] that group tasks based on their runtime to a resource capacity. Then, they proposed a technique [28] to determine the granularity of task based on CPU time, resource constraints and task file size. An online scheduling algorithm [27, 30] that cluster tasks based on application deadline, user's budget and resource network utilization was introduced. Ang et al.[2] and Ng et al.[21] aimed to increase the performance in the scheduling of tasks by introducing a factor of bandwidth. Further, Liu and Liao [24] proposed a technique for executing fine-grained jobs by grouping tasks considering the processing capacity of available resources and bandwidth. An approach for reusing and repurposing workflow is presented. It uses the metric of semantic similarity between layer hierarchies of a workflow. It adopted a graph skeleton based clustering technique for grouping layer hierarchies into clusters. This technique ranked the clusters. The similarity computation used is dependent on syntactic variations [49].

As there is a large number of processes involved in the execution of a scientific workflow. This may lead to high level of failures. A general task failure model is proposed that uses maximum likelihood based estimation for improving the execution time performance of scientific workflows [7]. This framework fails to take advantage of the combination of horizontal and vertical clustering.

2.3. Structural similarity. In [22] SFLA technique is proposed for encoding of workflows through workflow representations by exploiting set descriptors. In [50] author proposes a method for reusing and repurposing of a workflow by calculating the semantic similarity between layers of different workflows.These hierarchies are grouped into clusters. The author proposed a SimiFlow architecture for supporting clustering of workflows based on similarity.

2.4. Data dependency. Although the proposed techniques significantly decrease the impact of queuing time overhead and scheduling, they did not consider the factor of data dependencies. As clustering, the tasks horizontally increases the problem of dependency imbalance and runtime imbalance. To overcome these problems, three new methods of clustering Horizontal Runtime balancing, Horizontal impact factor balancing and Horizontal Distance Balancing are proposed in[10]. In these algorithms, only horizontal clustering is performed. In a workflow, there can be tasks with single parent single child relationship. In these kinds of tasks, vertical clustering can prove to be more advantageous then horizontal clustering technique. A general task failure model is proposed that uses maximum likelihood based estimation for improving the execution time performance of scientific workflows [7]. This framework fails to take advantage of the combination of horizontal and vertical clustering considering the single parent and child relationship in the nodes of a workflow. The deciding factor is unexplainable in research so as to choose whether to cluster tasks vertically or horizontally so as to maintain parallelism.

The existing work suffers from one or the following drawbacks

- The data dependencies between tasks not considered
- The runtime imbalance and dependency imbalance not considered
- The structure of workflow not considered
- The maximum parallelism between tasks not exploited

This work proposes Hybrid balanced task clustering technique to perform clustering while maintaining the parallelism of the system considering the single parent single child relationships in the nodes of workflow. Hybridizations of horizontal and vertical clustering technique has been achieved using impact factor based clustering technique. The cluster size is dynamically set as per job runtime to maintain the parallelism. Dependency variance is calculated using distance metrics. Horizontal and vertical clustering is performed as per the available resources, so that the parallelism is not affected. Hybrid clustering improves the performance of scientific workflow in cloud computing and a further benefit to data placement.

3. System Architecture.

3.1. Workflow. The Workflow Management Coalition (WfMC) defined a workflow as the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [17]. By the definition given by WfMC, the work process is a progression of organized exercises and calculations of a business procedure which is a unique depiction of the undertakings to be executed in that business process. They are utilized to join a few diverse computational procedures into a solitary lucid procedure. The business applications can now be seen as perplexing work processes, which comprise of different changes performed on the information fancied in accomplishing the goal. Workflows offer awesome points of interest in isolating capacities from applications and in this manner offering data framework to be segmented based by first arranging them and then incorporating them.

WfMC introduces its reference model in recognizing the interfaces inside of this structure which empowers items to operate interactively at an assortment of levels. It characterizes a work management framework and the most critical interfaces of a system as in Figure 3.1.

- **Workflow Engine:** A software service that gives the run-time environment with a specific end term goal to make, oversee and execute work process cases.
- **Process Definition:** Specifies the information about the process and the workflows related to it.

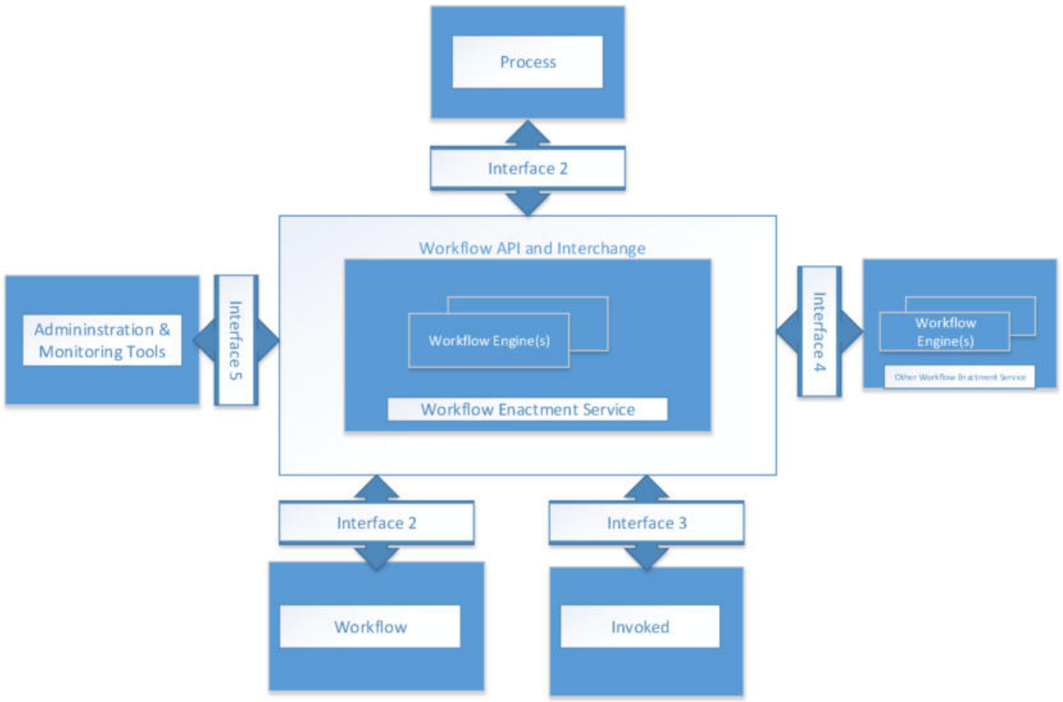

Process
Interface 2
Workflow API and Interchange
Admininstration & Monitoring Tools
Interface 5
Workflow Engine(s)
Workflow Enactment Service
Interface 4
Workflow Engine(s)
Other Workflow Enactment Service
Interface 2
Interface 3
Workflow
Invoked

FIG. 3.1. *WfMC reference model [17]*

- **Workflow Interoperability:** This interface makes interoperability possible between different processes of a workflow.
- **Invoked Applications:** Interfaces to bolster cooperation with an assortment of IT applications.
- **Workflow Client Applications:** Interfaces to bolster connection with the client interface.
- **Administration and Monitoring:** Interfaces to give a framework observing and metric capacities to encourage the administration of composite work process application situations.

3.2. Workflow model. A workflow application $Wf{=}(V_i, E_i)$ is represented as a directed acyclic graph (DAG) where $V_i{=}v_{i1}{,}v_{i2}..v_{in}$ is a set of vertices representing tasks and E_i represents edges control or data dependencies between them.A dependency e_{ij} is the precedence constraint of the form $(v_{i1}{,}v_{j1})$,where $v_{i1}{,}v_{j1}$ $\in V_i$ and $v_{i1} \neq v_{j1}$.This refers to that the child task can only complete its execution if the parent task has completed the execution. An example workflow is shown in Figure 3.2.

3.3. Workflow execution environment. A workflow is submitted to a workflow management system for execution that resides on a submit hosts which is user interaction machine. The execution machine for a workflow can be a grid, physical cluster [12], a dedicated parallel system[26], a virtual environment such as the cloud[4] or it can also be a local machine. Figure 3.3 shows a typical execution environment for scientific workflows. The components of this environment are listed below:

Workflow Mapper responsible for the generation of an executable workflow on the basis of an abstract workflow as submitted by a composition system or a user of the workflow. It finds the appropriate computational resources and data required for execution of a workflow. For optimization of performance, Workflow Mapper also restructures the workflow and add transformation for provenance information generation and data management.

Workflow Engine is responsible for the execution of jobs as per dependencies defined by the workflow. It manages the jobs by tracking their status. The jobs whose parent jobs have completed are given to the Job Scheduler.

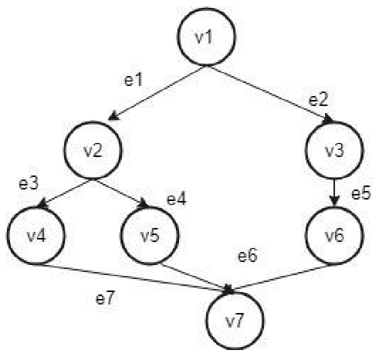


FIG. 3.2. *Workflow model*

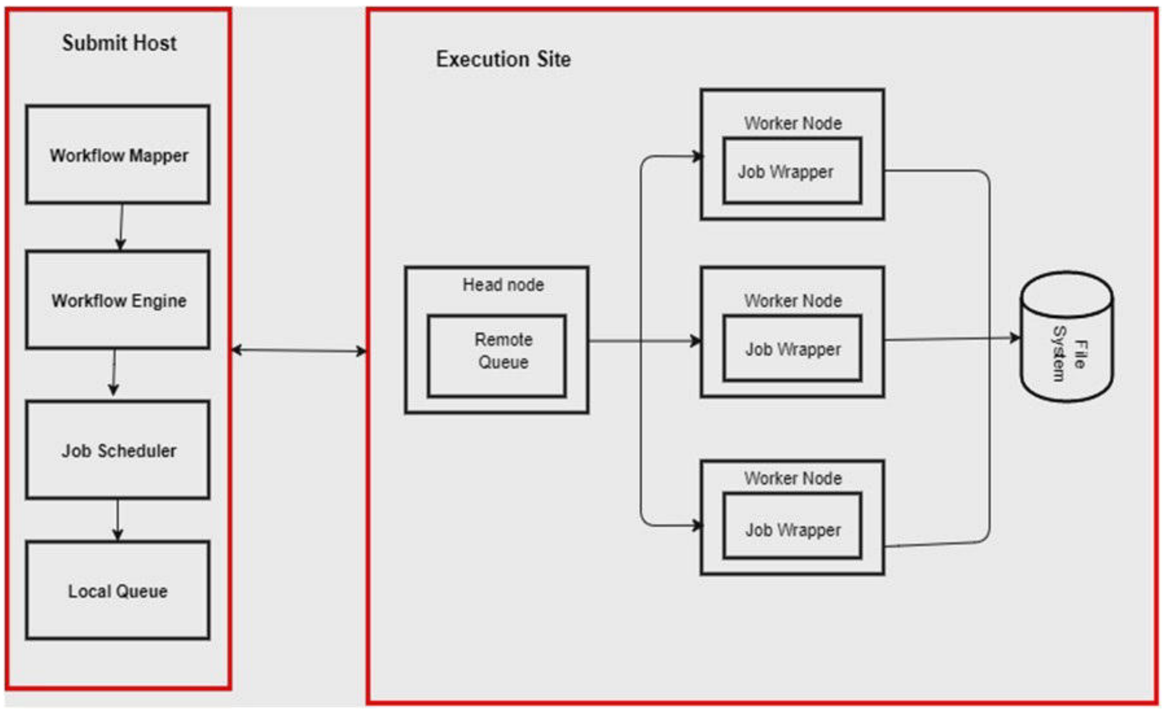


FIG. 3.3. *Workflow management system*

Local Queue and **Job Scheduler** manages workflow jobs individually and observer their execution on remote and local resources. Different scheduling algorithms such as HEFT[43] and MinMin[5] can be applied to the Job Scheduler and improve the overall runtime of workflow executions.

Job Wrapper unwraps tasks from clustered jobs. Then these are executed at the worker nodes. All of these components work cooperatively with each other to perform workflow preparation and execution.

4. Hybrid Balanced Task Clustering algorithm.

In this, the proposed hybrid balanced clustering algorithm used for task clustering is discussed which is not dependent on the input of the user. It is able to cluster the tasks vertically with single parent single child relationship and the tasks horizontally as well. The system overhead is reduced while involving best utilization of resources. The flowchart of proposed technique is depicted in Figure 4.1. The symbols used in this work are explained in Table 3.1.

4.1. Background.

The two major issues that are undertaken by the clustering algorithms, dependency imbalance, and runtime imbalance. Runtime imbalance refers to the unequal distribution of runtime of tasks at

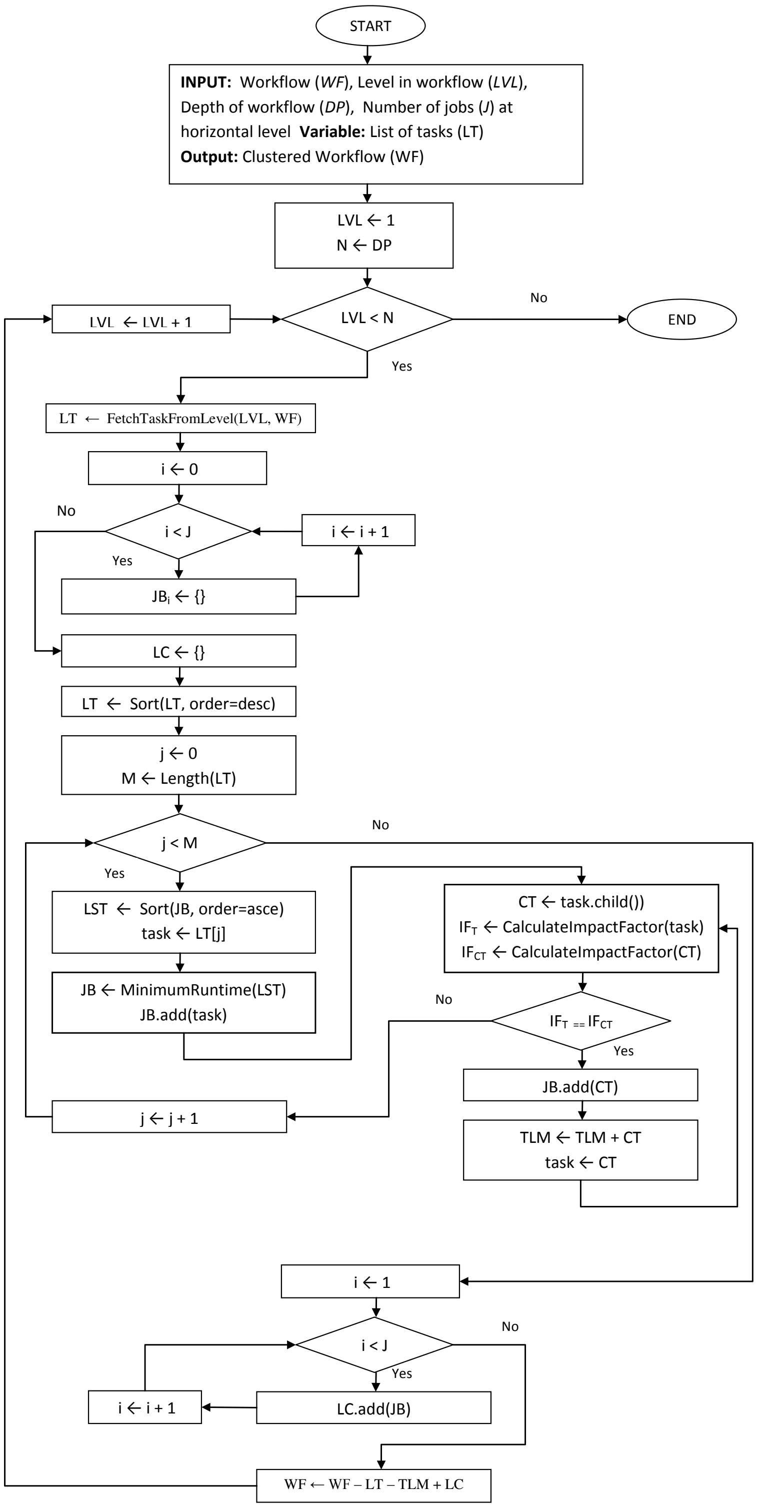

START
INPUT: Workflow (WF), Level in workflow (LVL), Depth of workflow (DP), Number of jobs (J) at horizontal level Variable: List of tasks (LT)
Output: Clustered Workflow (WF)
LVL ← 1
N ← DP
LVL ← LVL + 1
LVL < N
No
END
Yes
LT ← FetchTaskFromLevel(LVL, WF)
i ← 0
No
i < J
i ← i + 1
Yes
JBi ← {}
LC ← {}
LT ← Sort(LT, order=desc)
j ← 0
M ← Length(LT)
j < M
No
Yes
LST ← Sort(JB, order=asce)
task ← LT[j]
CT ← task.child())
IFT ← CalculateImpactFactor(task)
IFCT ← CalculateImpactFactor(CT)
JB ← MinimumRuntime(LST)
JB.add(task)
No
IFT == IFCT
Yes
JB.add(CT)
j ← j + 1
TLM ← TLM + CT
task ← CT
i ← 1
i < J
No
Yes
i ← i + 1
LC.add(JB)
WF ← WF – LT – TLM + LC

Fig. 4.1. *Flowchart of Hybrid Balanced Clustering Algorithm*

Table 3.1
Explanation of symbols used in this work

Abbreviation	**Definition**
WF	Workflow
LVL	Level in a workflow
J	Number of jobs at a horizontal level
LT	List of tasks at a level
DP	Depth of workflow
JB_i	Empty Job
LC	Empty list of clustered jobs
LST	Sorted List of jobs
CT	Child Task
IF	Impact Factor
TLM	Merged Task List

the same horizontal level. While dependency imbalance refers to clustering tasks at a level without considering the factor of dependency between tasks. This increases the waiting time for input at the next level and thus the delay in execution. The problem is also referred to as data locality. Generally, runtime imbalance leads to dependency imbalance and dependency imbalance leads to runtime imbalance. The structure of workflow is also an important factor while clustering the tasks. As horizontal clustering is always performed for tasks that may increase the problems of runtime imbalance and dependency imbalance. So instead of performing task clustering horizontally at each level of workflow, vertical clustering can also be performed of the tasks where single parent single child relationship exists between tasks besides performing clustering horizontally. This may lead to improvement in the execution of workflow while further decreasing the delays. Considering the above challenges of clustering, the introduced method aims to obtain appropriate hybrid clustering while merging the tasks vertically with the similar impact factor and remaining tasks horizontally according to number of available resources. Thus reducing the overall execution time. For the realization of the desired clustering, the proposed Hybrid balancing algorithm as follows.

Algorithm working: This technique prefers to cluster the tasks with single parent single child relationship. The problem of dependency imbalance is catered by measuring the impact factor of tasks in the workflow. The Impact Factor (IF) of task t_n is denoted by

$$IF(t_n) = \sum_{t_a \subset child(t_a)} \frac{IF(t_a)}{parent(t_a)} \tag{4.1}$$

where $child(t_a)$ denotes a set of child tasks of t_a and $parent(t_a)$ denotes set of parent tasks of t_a The impact factor is used to measure the similarity between the tasks or jobs[9].

The proposed algorithm ensures that the tasks with one parent-child relationship are clustered into one cluster. The remaining tasks are then clustered horizontally. The parent child relationship is depicted by matching the impact factor of tasks vertically. The tasks with the similar impact factor are grouped into one cluster. This help in reducing the execution time of workflow. For example figure 4.2 shows a five-level workflow composed of one task at level one, two tasks at level-two, four tasks at level-three, three tasks at level-four and one task at level-five.

Algorithm 1 shows the pseudo code of hybrid balancing algorithm that uses the combination of horizontal distance balancing [10], horizontal runtime balancing [10] and impact factor [10] of tasks. We assume the number of jobs as an input from the user. This algorithm addresses the problem of load imbalance and also considers the tasks with asymmetric structure. The algorithm begins with the first level of workflow as in Figure 4.2 by selecting tasks from each level (Lines 2-3). Tasks are clustered into a job and returned by merge procedure (Line 4).The merge procedure merges the tasks vertically and horizontally also. In merge procedure, the tasks are sorted according to decreasing order of runtime (Line 13). Considering the level three of a workflow. There are four tasks D(30s),E(10s),F(30s),G(20s). The task list LT is maintained as according to decreasing order

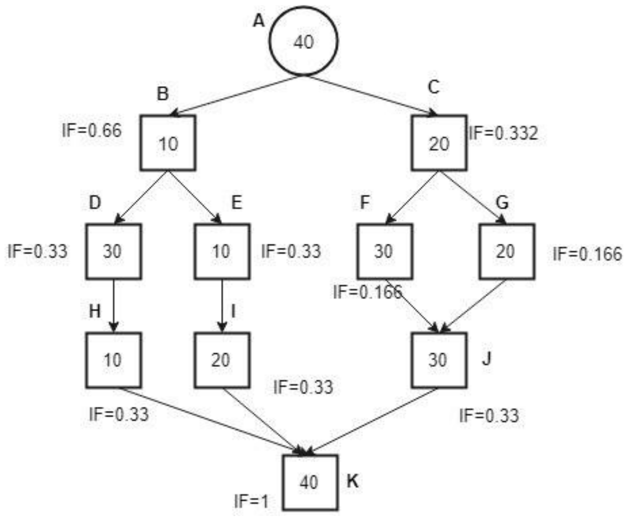


FIG. 4.2. *Example workflow*

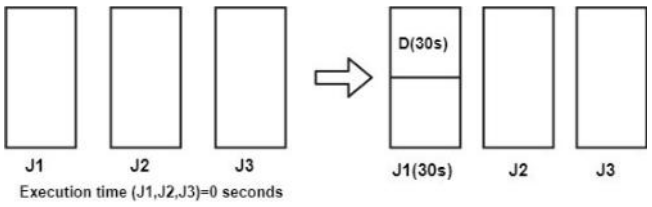


FIG. 4.3. *Merging of clusters in a job*

of runtime of tasks $LT{=}D, F, G, E$. Then horizontal runtime balancing adds the task D to the job with the shortest runtime as in Figure 4.3.

Step 2: The tasks are arranged as per the minimum distance with the task D. $Distance{=}0(D), 4(F), 4(G)$, $2(E)$. The impact factor of task D included in a job is matched vertically to depict parent-child relationship exists or not. If the relationship exists then the child task is added to the job where task D resides (Lines 20-22) as shown in Figure 4.4 and Figure 4.5.

Step 3: Now after the clustering of tasks D and H. The tasks in a task list(LT) are task F,G,E. The horizontal distance balancing is performed on tasks E, F and G. In this jobs J1, J2, J3 are sorted based on shortest distance between them and targeted task D. The distances are 4,4,2 respectively for tasks F, G, E. The candidate is selected with minimal distance 2 of task E. Hence impact factor of task E is matched vertically and if the impact factor matches with the below vertically then both are clustered into another cluster c2 as shown in Figure 4.6 and merged as a job J2 as shown in Figure 4.7.

Step 4: Now the tasks left in the task list LT are F and G. Similarly the process is repeated for the tasks F and G. But these tasks do not have a parent child relationship with any task. So they cannot be clustered vertically. Hence Horizontal clustering is performed for the tasks F, G forming cluster c3 as shown in Figure 4.8 and merging into job J3 as shown in Figure 4.9.

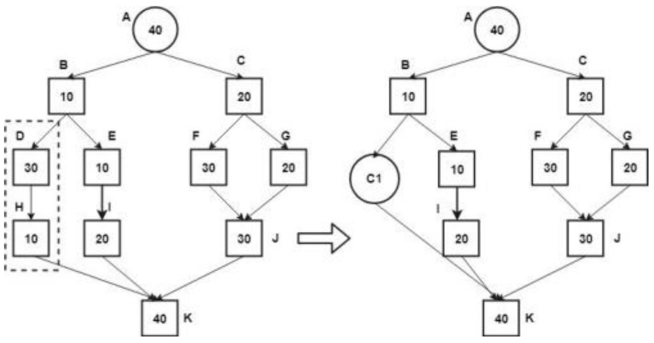


FIG. 4.4. *Clustering to avoid runtime imbalance and dependency imbalance*

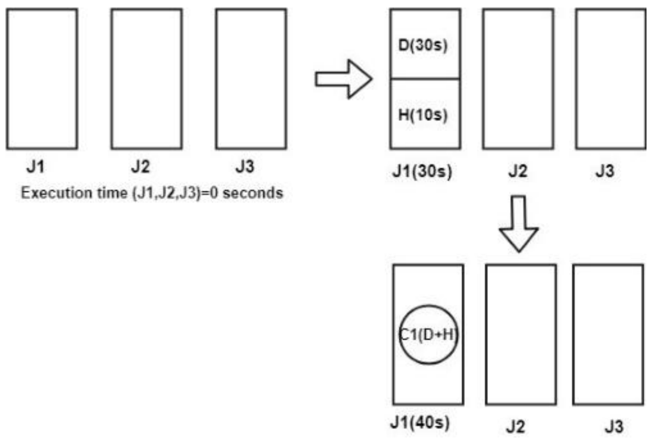


FIG. 4.5. *Merging clusters into a job*

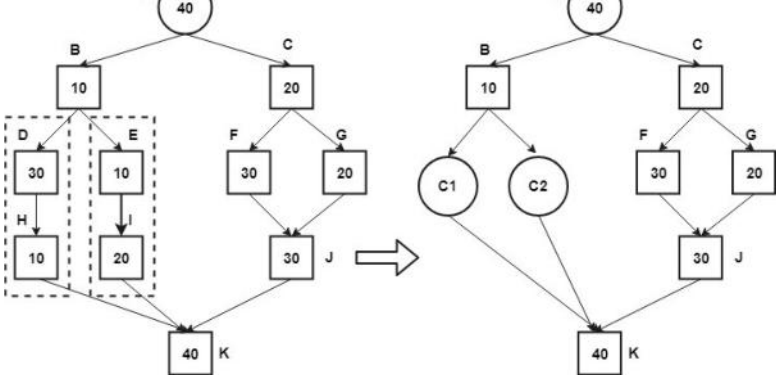


FIG. 4.6. *Clustering on runtime and dependencies*

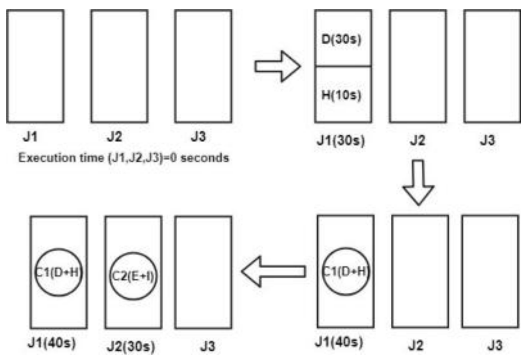


Fig. 4.7. *Merging clusters into a job*

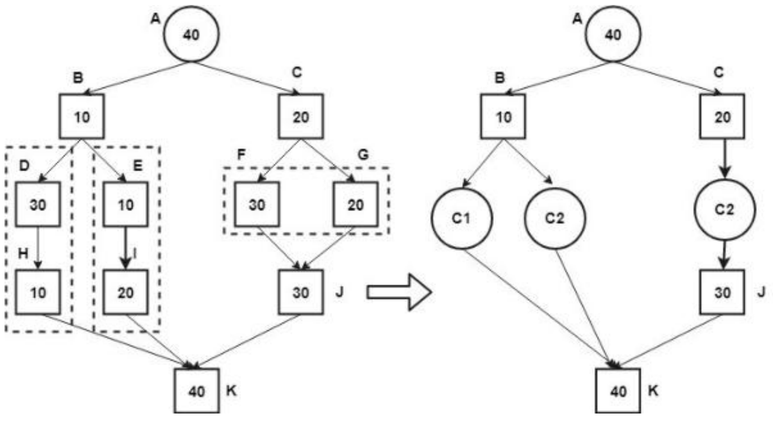


Fig. 4.8. *Clustering to avoid runtime imbalance and dependency imbalance*

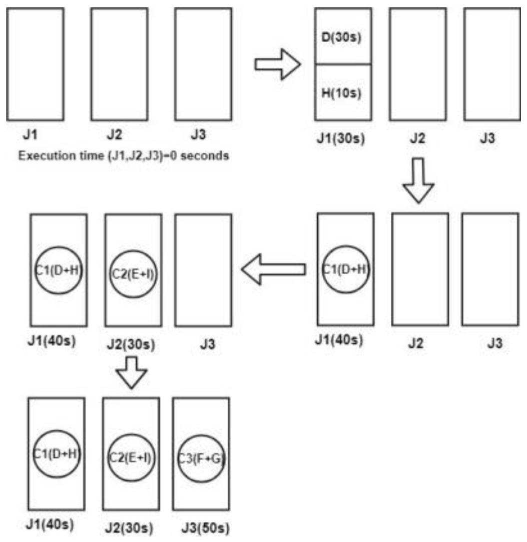


Fig. 4.9. *Merging clusters into a job*

4.2. Pseudocode for Hybrid Balanced clustering algorithm (HYB). Algorithm 1 presents it.

Algorithm 2 Hybrid Balanced Clustering algorithm.

```
Require: $WF$: workflow;$J$: number of jobs at horizontal level ;$LVL$:Level of workflow;$DP$:depth of workflow
 1: procedure HYCLUSTERING($Wf$)
 2:     for $LVL < DP(WF)$ do
 3:         $LT \leftarrow$ FETCHTASKSFROMLEVEL($WF, LVL$)        ▷ Divide $WF$ on the basis of depth
 4:         $TLM$,$LC \leftarrow$ MERGE($LT, J$)                 ▷ Group of clustered job returned
 5:         $WF \leftarrow WF - LT + LC - TLM$                   ▷ Dependency merging
 6:     end for
 7: end procedure
 8: procedure MERGE($LT, J$)
 9:     for $i < J$ do
10:         $JB_i \leftarrow \{\}$                                ▷ NULL JOB
11:     end for
12:     $LC \leftarrow \{\}$                                      ▷ NULL JOB LIST
13:     Sort $LT$ in descending of runtime
14:     for all $task$ in $LT$ do
15:         $LST \leftarrow$ sort list of $JB_i$ as per least distance with $task$
16:         $JB \leftarrow$ the job with minimum runtime in $LST$
17:         $JB$.add ($task$)
18:         while $IF_T = IF_{CT}$ do                             ▷ CT is child task
19:             $JB$.add(CT)
20:             $TLM \leftarrow TLM + CT$
21:             $task \leftarrow CT$
22:         end while
23:     end for
24:     for $i < J$ do
25:         $LC$.add( $JB$)
26:     end for
27:     return $LC$,$TLM$
28: end procedure
```

5. Performance Evaluation.

5.1. Scientific workflow applications. Montage [3] is one of the application of astronomy used for constructing large image mosaics of the sky. In it, the input images are projected onto a sphere and then the overlap for each input image is calculated. The input images are projected to the accurate orientation while maintaining the constant emission level of background for all images. At last, reprojected images are added into a final mosaic. The final resulting image provides a detailed description of the part of the sky under investigation. Figure 5.1 represents the Montage workflow. The size of the workflow is dependent upon the number of images used for constructing the desired mosaic of the sky.

Cybershake. CyberShake [14] is an application of seismology that is used for calculating Probabilistic Seismic Hazard curves for various geographic regions in Southern California. It is used in the identification of all ruptures within 200KM radius. It also changes rupture into multiple variations of rupture that differ in hypocenter locations and slip distributions. Then for each rupture variance, synthetic seismograms are calculated. After that, the peak intensity is extracted and added with the original rupture probability for production probabilistic hazards for the location. Figure 5.2 shows an illustration of the Cybershake workflow.

Epigenomics. The Epigenomics workflow [44] is a CPU-intensive application. Initially the data is obtained in the form of DNA sequence lanes from the Illumina-Solexa Genetic Analyzer. Every Solexa machine generates multiple DNA sequences. Then the mapping of DNA sequences to the accurate locations in a genome is performed by the workflow. This produces a map showing the density of sequence. A simplified structure of Epigenomics is shown in Figure 5.3.

SIPHT. The SIPHT workflow [19] is responsible for researching small untranslated RNAs (sRNAs). It is responsible for regulation of different processes such as virulence or secretion in bacteria. This predict ρ-independent transcriptional terminators. A simplified version of SIPHT workflow is depicted in Figure 5.4.

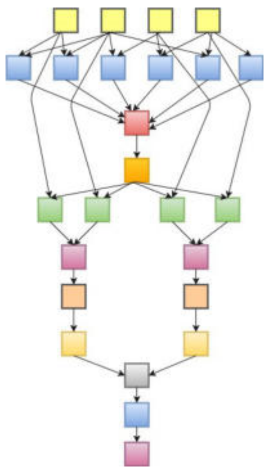


Fig. 5.1. *A simplified visualization of the Montage workflow[3].*

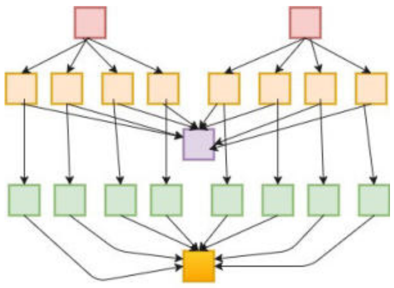


Fig. 5.2. *A simplified visualization of the CyberShake workflow [14].*

LIGO. In Laser Interferometer Gravitational Wave Observatory (LIGO)[1] workflows data is collected by large-scale interferometers for searching of gravitational wave signatures. The observers aim is to measure and detect waves as predicted by relativity. It is a data-intensive workflow. Figure 5.5 depict a version of workflow. In this workflow tasks are separated into different groups where a group consisting of interconnected tasks as shown in Figure 5.5.

5.2. Balanced Task Clustering algorithms. The task clustering techniques are classified into two different categories Horizontal clustering and vertical clustering. Further to overcome the limitations of these techniques, balanced clustering is introduced.

5.3. Description of Baseline Balanced Clustering Algorithms. In order to study the impact of the proposed clustering technique, the clustering methods considered are: Horizontal Impact Factor Balancing (HIFB) Horizontal Clustering (HC), Horizontal Distance Balancing (HDB) and Horizontal Runtime balancing (HRB) [8, 10].

Horizontal Clustering (HC): In this technique tasks at the same horizontal level of workflow is merged into a job. The horizontal level for a single task is considered as the largest distance from the first task of workflow starting from the first task. The first task is a task without a parent task.

As shown in Figure 5.6, the tasks t2, t3 and t4 are combined together into a cluster c1, thus creating just one job j1. The horizontal clustering is performed for the tasks at the same level. Thus reducing the overhead.

Vertical Clustering Clustering (VC): In this algorithm task at the same vertical level of a workflow are merged and make a single job. In this, the tasks with relationship between one parent and one child are merged.

As shown in Figure 5.7 the tasks t2, t5, t8 exhibit parent-child relationship as t2 is parent of t5 and further

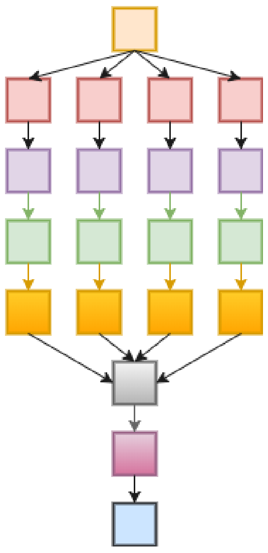


FIG. 5.3. *Epigenomics workflow with multiple branches [44].*

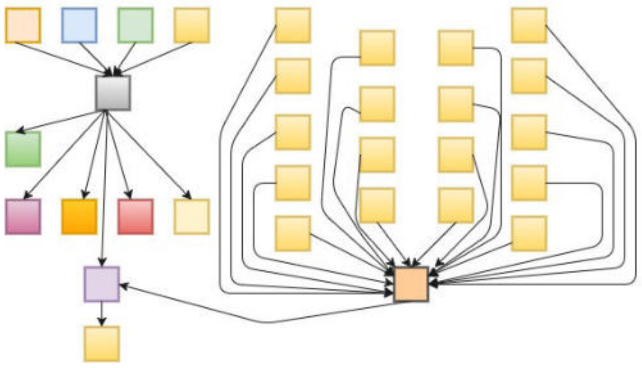


FIG. 5.4. *A simplified visualization of the SIPHT workflow [19].*

t5 is parent t8. So these tasks are clustered together into a cluster c1, thus creating a job1. Similarly t3, t6, t9 and t4, t7, t10 are clustered into clusters c2,c3 thus creating combined jobs job2, job3 respectively. The overheads are also combined into overheads o2, o3 and o4.

Horizontal Runtime Balancing (HRB): In this algorithm runtime of tasks is equally distributed between jobs. The problem of runtime variance is addressed at the same horizontal level. In this greedy method, the jobs are sorted according to the ascending order of runtime. The new task is joined to the job with a minimum runtime. This method further raises the dependency imbalance among tasks as the factor of data dependency is not considered while clustering.

As shown in Figure 5.8, there are four tasks t1, t2 and t3, t4 with runtime 20s and 30s respectively. According to this clustering method tasks t1, t3 are combined into one cluster and t2, t4 are combined into another cluster. This balances the runtime among tasks, but leads to dependency imbalance among tasks.

Horizontal Impact Factor Balancing (HIFB): It overcomes the limitation of Horizontal runtime balancing algorithm of dependency imbalance. In this algorithm, the jobs are first sorted by considering the similarity of impact factor (IF) in increasing order. Then the shortest job is selected using Horizontal Runtime Balancing. It groups the jobs sharing the same position in the workflow.

As shown in Figure 5.9 the tasks t1, t2 and t3, t4, t5 have same the impact factor. Then using HRB the tasks t1, t2 and t3, t4 are combined into clusters c1, c2.

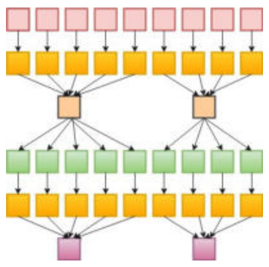


FIG. 5.5. *A simplified visualization of the LIGO Inspiral workflow [1].*

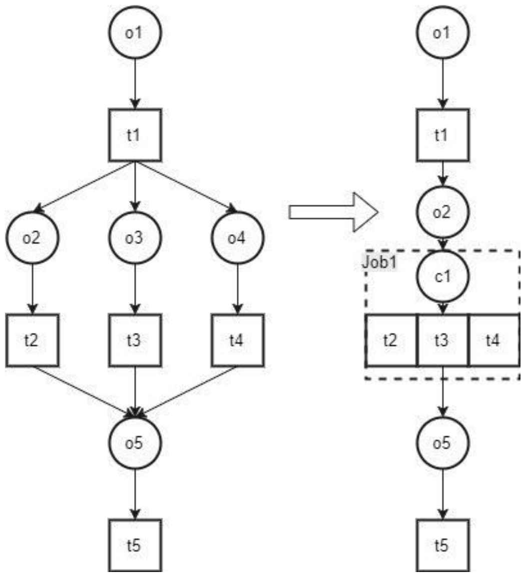


FIG. 5.6. *Horizontal Clustering.*

Horizontal Distance Balancing (HDB): In this clustering technique jobs are sorted considering the distance between them and the target job. After that Horizontal Runtime Balancing method is executed for selecting the shortest job. The tasks with minimum distance are merged thus reducing data transfer.

As in Figure 5.10 the tasks t1 and t2 are closest as compared to distance between t1, t3 and t1, t4. So t1, t2 are combined into one cluster and t3, t4 are combined into another cluster.

The performance of above discussed algorithms has been evaluated using workflowsim on various datasets e.g. Montage, LIGO.

5.4. Experiments and Results.

Experimental setup. The trace based simulation method is adopted for evaluation. The performance of the proposed technique is compared with the baseline algorithm. Real traces are used for evaluation of algorithms. In the environment, different system parameters used are a number of virtual machines, system overhead, different workflows and sizes of data.

For executing applications of workflow, open source workflow simulator, workflowsim is used. It is used for modeling an execution environment in the cloud. It is an extension of cloudsim. The DAG model of workflow is executed in it. It performs clustering and scheduling of tasks. It also performs provisioning of resources at the workflow level.

In the experiment, the simulator WorkflowSim is extended to implement the proposed hybrid balanced task clustering technique. A virtual machine cluster consisting of 20 single homogeneous core VMs is considered. This cluster is a quota for a user in some distributed environments such as Amazon EC2. Each VM detains

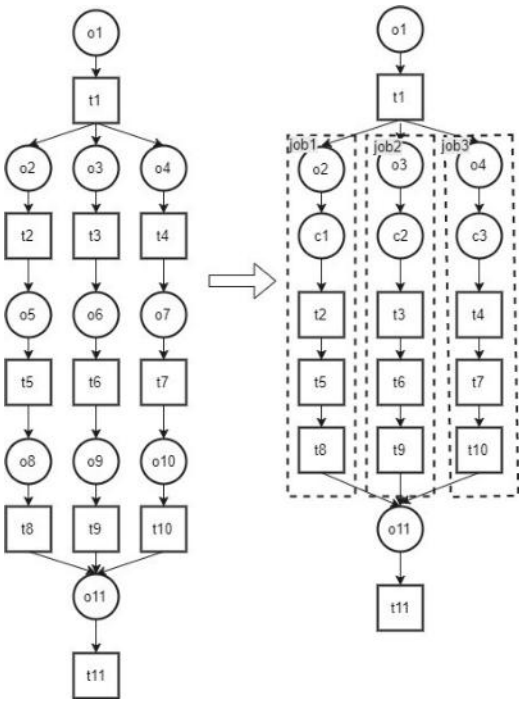


Fig. 5.7. *Vertical clustering.*

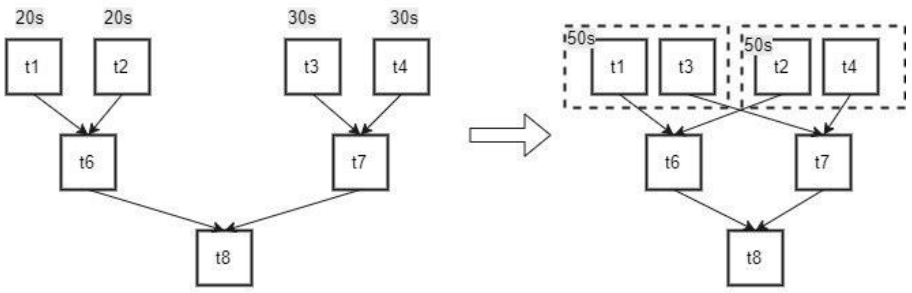


Fig. 5.8. *A sample of the HRB (Horizontal Runtime Balancing) method.*

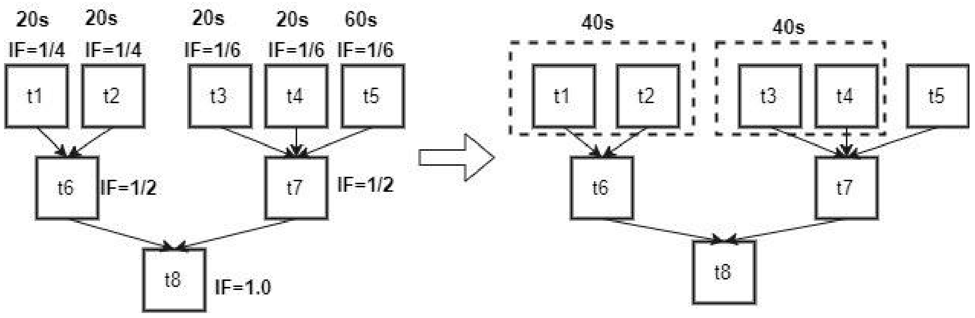


Fig. 5.9. *HIFB (Horizontal Impact Factor Balancing) method.*

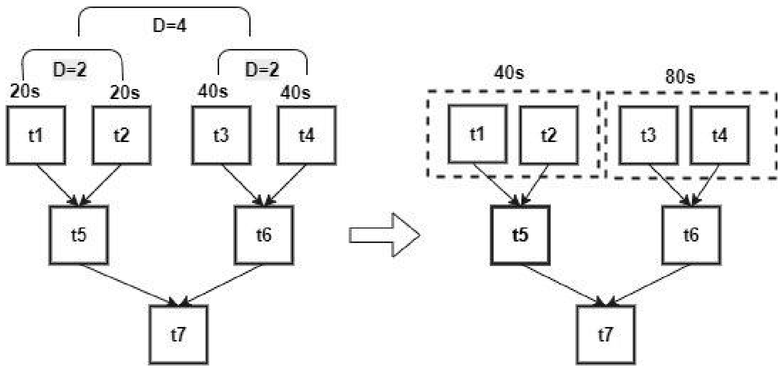


Fig. 5.10. *HDB (Horizontal Distance Balancing) method.*

Table 5.1
Configuration setup for the experiment

Parameter	**size**
No. of virtual machines	20
Memory Capacity	512 MB
Processing capacity	1000 MIPS
Network Bandwidth	15 MB/s

configuration of 512 MB. The processing capacity for each virtual machine is 1000 MIPS and by default, the network bandwidth was set to 15 MB/s.

The configuration setup for the experiment is described as in Table 5.1.

An evaluation of the proposed technique was performed for identification of its ability. The method was evaluated for the improvement in execution time of scientific workflow and reducing the load to minimum resources. The experiment is conducted on the following variables.

- **Execution time of workflow :** Total time taken by the workflow application to execute on the available resources.
- **Performance Gain (μ):** It is defined as an overall improvement in execution time of workflow by using the clustering algorithm over the execution of the workflow without clustering of a task. It can be evaluated using equation as follows:
  $\mu =$ (time taken without clustering-time taken with clustering)/(time taken without clustering)*100
  $\mu > 0$ for a clustering technique signifies that it leads to improvement of the execution time of a workflow . Whereas $\mu < 0$ refers to negative impact of clustering method. This negative impact lead to increase of execution time of worklfow.

Different scientific workflow applications are used in the experiments along with the fixed number of tasks for each as shown in Table 5.2.

5.5. Result and Analysis. Table 5.2 depicts the scientific workflow applications used for the experiment and the number of tasks considered for each.

Then the makespan time for each scientific application is calculated as depicted in Table 5.3.

Using the makespan time, performance gain of scientific applications cybershake, epigenomics, LIGO, Montage and SIPHT is calculated and compared as evaluated according to different baseline algorithms and proposed algorithm in Table 5.4.

Three set of experiments are conducted on the scientific workflow applications.

Experiment 1: Improvement in execution time The makespan time of balanced clustering algorithms on different scientific workflow applications cybershake, epigenomics, LIGO, Montage, SIPHT is obtained from the experiments performed. The makespan time is depicted in Table 5.3. According to the experimental evaluations the following results are depicted:

Table 5.2
Scientific workflow for experiment and number of tasks

Workflow	Number of tasks
Cybershake	1000
Epigenomics	997
Montage	1000
SIPHT	1000
LIGO	1000

Table 5.3
Makespan time of workflow with or without clustering algorithms

	HC	HRB	HIFB	HDB	HYB	without clustering
Cybershake	1511.46	1303.58	1833.68	1445.07	1352.92	2222.05
Epigenomics	238974.2	218583.8	241833.1	238754	206077.6	253462
LIGO	13188.79	12768.49	15261.21	13015.6	11635.49	17234.23
Montage	1210	924.71	936	947	923.61	1927.69
SIPHT	21154.99	9537.03	22778.28	18539.23	9499.07	23453.58

- **Cybershake Workflow Application**: In this application, there is 26% and 10% improvement in execution time by hybrid balanced clustering algorithm(HYB) then horizontal impact factor balancing(HIFB) and horizontal clustering(HC) respectively.
- **Epigenomics workflow application** : In this application, there is 14.78%, 13.68%, 13.76% improvement in execution time by Hybrid Balanced Clustering algorithm then horizontal impact factor balancing(HIFB), Horizontal Distance Balancing(HDB) and Horizontal Clustering(HC) respectively.
- **LIGO**: In this application there is 23.75%, 11.77% , 10.60% and 8.87% improvement in execution time by Hybrid Balanced Clustering algorithm then Horizontal impact factor balancing(HIFB), Horizontal clustering(HC), Horizontal Distance balancing(HDB) and Horizontal runtime balancing(HRB) respectively.
- **Montage Workflow Application** : In this application there is 23%, 2.46% improvement in execution time of workflow by Hybrid Balanced Clustering algorithm then Horizontal Clustering (HC) and Horizontal Distance Balancing respectively.
- **SIPHT** : In this application there is 58%,55% and 48% improvement in execution time by Hybrid Balanced Clustering algorithm then Horizontal Impact Factor Balancing(HIFB),Horizontal Clustering(HC) and Horizontal Distance Balancing(HDB) respectively.

Hence from the experiment 1, it is concluded that the execution time taken by Hybrid Balanced Task Clustering algorithm(HYB) is lesser as compared to other baseline algorithms. Proposed technique shows the overall improvement in execution time.

Experiment 2: Performance Gain In this experiment, the performance gain (μ) of the proposed technique is evaluated with the other clustering algorithms. In this experiment, the proposed technique is evaluated for identification of the amount to which it impacts the overall execution time of workflow.Figure 5.4 shows the performance gain μ for different clustering methods obtained by the performed experiment. From the experiments, it is depicted that all the methods of clustering retain a positive gain performance. This further improves the execution time of different workflow applications. According to obtained results Montage, CyberShake and SIPHT workflows have better performance gains with a minimum gain of 52%,41% and 59%. In comparison LIGO and Epigenomics having a minimum performance gain of 32% and 18% respectively. The variation in performance gain is due to the granularity of the average runtime of tasks in workflows. Considering the workflows the lowest value for the performance gain is for HIFB in SIPHT workflow. In all the scientific workflows, the technique Horizontal Runtime Balancing(HRB) and proposed Hybrid balanced Task clustering(HYB) method performs better than the other balancing techniques. The clustering method HDB(Horizontal Distance Balancing) and HIFB(Horizontal Impact Factor Balancing) lack in performance, because there runtime as well as

Table 5.4
Performance gain for various workflows

	Cybershake	**Epigenomics**	**LIGO**	**Montage**	**SIPHT**
HC	31.97903	5.715961	23.47329	37.23057	9.800593
HRB	39.11388	13.76072	25.91204	52.03015	59.33649
HIFB	17.478	4.588013	11.44826	51.44447	2.879305
HDB	34.96681	5.802842	24.4782	50.87384	20.95352
HYB	41.33435	18.69489	32.48616	52.08721	59.49842

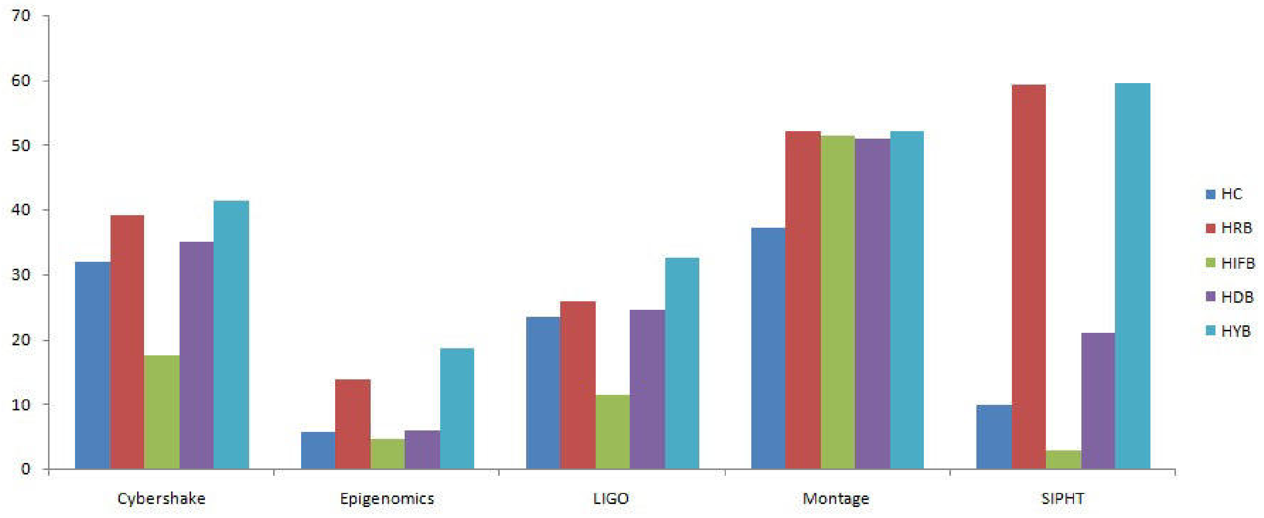


Fig. 5.11. *Comparison of performance Gain (μ) for experiment for various clustering methods.*

dependency imbalance between tasks. This groups the tasks into a cluster that should execute in parallel leading to an increase in execution time of workflow. The proposed hybrid balancing clustering method delivers a good performance since it first checks for parent-child relationship and then clusters the tasks. Thus increasing the performance gain and decreasing overall execution time of workflow as shown in Figure 5.11.

Experiment 3: Varying virtual machines In experiment 3, the number of virtual machines are varied while keeping the total number of tasks 1000 MIPS.

First, the experiment is performed by using a horizontal clustering technique while varying the number of virtual machines (VM) as shown in Figure 5.12.

The same experiment is conducted by using hybrid balanced clustering algorithm and the result is depicted as shown in Figure 5.13.

While comparing both the results, it is found that horizontal clustering technique performs in a similar manner inspite of more number virtual machines available. While in the proposed algorithm, the execution time decreases as the number of virtual machines increases. Hence leading to an increase in performance of execution of workflow applications. But at an instance the proposed technique performs as similar to baseline algorithms. This depicts the proposed technique also reaches a stagnation point but after more of the resource utilization in comparison to baseline algorithms.

As in Figure 5.12 the graph depicts the number of virtual machines increases the execution time decreases but at an instant the performance is constant and it does not matter on adding more virtual machines. The proposed technique is performing in a similar manner and adding up the virtual machines decreases the execution time. But at a point the performance becomes constant.

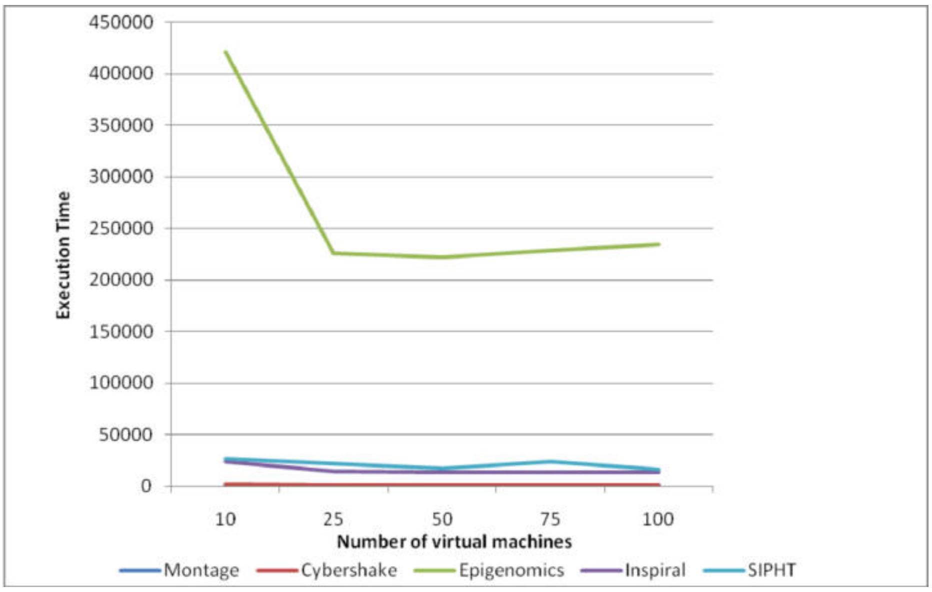


Fig. 5.12. *Execution time while varying virtual machines in Horizontal Clustering technique.*

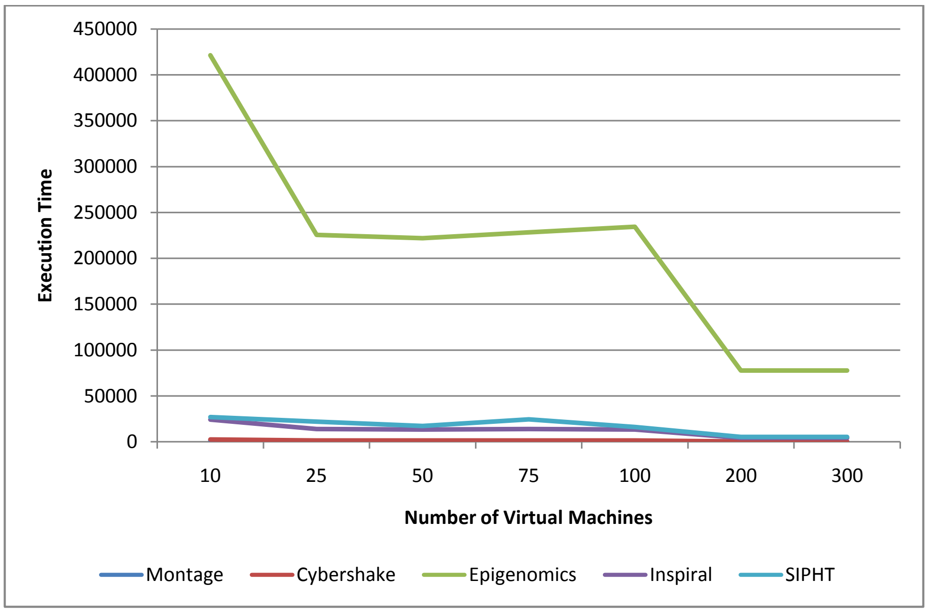


Fig. 5.13. *Execution time while varying virtual machines in Hybrid Balanced Clustering technique.*

6. Conclusion and Future Scope. The work proposes a hybrid balancing (HYB) task clustering method that reduces the full time of workflow execution and thus avoids waste of resources. The proposed approach considers distance variance and merges the tasks with similar impact factor in pipeline. A simulation experiment is conducted for evaluation of the proposed algorithm in comparison to four clustering methods: Horizontal Runtime Balancing (HRB), Horizontal Clustering (HC), Horizontal Distance Balancing (HDB) ,and Horizontal Impact Factor Balancing (HIFB). The experiment aimed to evaluate the execution time improvement. The results depict that the proposed clustering method is able to perform better than the other methods. Hence it can be used in different workflow structures.

In near future, it is intended that the size of the resource set can be identified for optimized execution of a workflow. Also, the proposed approach can be implemented in a workflow engine. So that this approach can be used for clustering tasks in a real environment.

REFERENCES

[1] Abramovici, A., Althouse, W. E., Drever, R. W., Gürsel, Y., Kawamura, S., Raab, F. J., Shoemaker, D., Sievers, L., Spero, R. E., Thorne, K. S., et al.(1992). Ligo: The laser interferometer gravitational-wave observatory. *Science*, 256(5055):325–333.

[2] Ang, T., Ng, W., Ling, T., Por, L., and Liew, C.(2009). A bandwidth-aware job grouping-based scheduling on grid environment. *Information Technology Journal*, 8(3):372–377.

[3] Berriman, G. B., Deelman, E., Good, J. C., Jacob, J. C., Katz, D. S., Kesselman, C., Laity, A. C., Prince, T. A., Singh, G., and Su, M.(2004). Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand. In *SPIE Conference on Astronomical Telescopes and Instrumentation*.

[4] Berriman, G. B., Juve, G., Deelman, E., Regelson, M., and Plavchan, P.(2010). The application of cloud computing to astronomy: A study of cost and performance. In *Workshop on e-Science challenges in Astronomy and Astrophysics*.

[5] Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., and Kennedy, K.(2005). Task scheduling strategies for workflow-based applications in grids. In *5th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '05)*.

[6] Callaghan, S., Maechling, P., Small, P., Milner, K., Juve, G., Jordan, T., Deelman, E., Mehta, G., Vahi, K., Gunter, D., Beattie, K., and Brooks, C. X.(2011). Metrics for heterogeneous scientific workflows: A case study of an earthquake science application. *International Journal of High Performance Computing Applications*, 25(3):274–285.

[7] Chen, W., Da Silva, R. F., Deelman, E., and Fahringer, T.(2016). Dynamic and fault-tolerant clustering for scientific workflows. *IEEE Transactions on Cloud Computing*, 4(1):49–62.

[8] Chen, W., Da Silva, R. F., Deelman, E., and Sakellariou, R.(2013a). Balanced task clustering in scientific workflows. In *2013 IEEE 9th International Conference on e-Science*, pages 188–195. IEEE.

[9] Chen, W., Deelman, E., and Sakellariou, R.(2013b). Imbalance optimization in scientific workflows. In *Proceedings of the 27th international ACM conference on International conference on supercomputing*, ICS '13, pages 461–462.

[10] Chen, W., Ferreira Da Silva, R., Deelman, E., and Sakellariou, R.(2015). Using imbalance metrics to optimize task clustering in scientific workflow executions. *Future Generation Computer Systems*, 46:69–84.

[11] Deelman, E., Kesselman, C., Mehta, G., Meshkat, L., Pearlman, L., Blackburn, K., Ehrens, P., Lazzarini, A., Williams, R., and Koranda, S.(2002). GriPhyN and LIGO: building a virtual data grid for gravitational wave scientists. In *11th IEEE International Symposium on High Performance Distributed Computing (HPDC '02)*.

[12] Fahringer, T., Prodan, R., Duan, R., Hofer, J., Nadeem, F., Nerieri, F., Podlipnig, S., Qin, J., Siddiqui, M., Truong, H.-L., et al.(2007). Askalon: A development and grid computing environment for scientific workflows. In *Workflows for e-Science*, pages 450–471. Springer.

[13] Ferreira da Silva, R., Glatard, T., and Desprez, F.(2013). On-line, non-clairvoyant optimization of workflow activity granularity on grids. In *Euro-Par 2013 Parallel Processing*, volume 8097 of *Lecture Notes in Computer Science*, pages 255–266. Springer Berlin Heidelberg.

[14] Graves, R., Jordan, T., Callaghan, S., Deelman, E., Field, E., Juve, G., Kesselman, C., Maechling, P., Mehta, G., Milner, K., Okaya, D., Small, P., and Vahi, K.(2010). CyberShake: A Physics-Based Seismic Hazard Model for Southern California. *Pure and Applied Geophysics*, 168(3-4):367–381.

[15] Guo, Z., Pierce, M., Fox, G., and Zhou, M.(2011). Automatic task re-organization in mapreduce. In *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, pages 335–343.

[16] Hey, T., Tansley, S., Tolle, K. M., et al.(2009). *The fourth paradigm: data-intensive scientific discovery*, volume 1. Microsoft research Redmond, WA.

[17] Hollingsworth, D. and Hampshire, U.(1995). Workflow management coalition: The workflow reference model. *Document Number TC00-1003*, 19.

[18] Hussin, M., Lee, Y. C., and Zomaya, A. Y.(2010). Dynamic job-clustering with different computing priorities for computational resource allocation. In *The 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*.

[19] Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., and Vahi, K.(2013). Characterizing and profiling scientific workflows. *Future Generation Computer Systems*, 29(3):682–692.

[20] Kalayci, S., Dasgupta, G., Fong, L., Ezenwoye, O., , and Sadjadi, S.(2010). Distributed and adaptive execution of

condor dagman workflows. In *Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE'2010)*.

[21] Keat, N. W., Fong, A. T., Chaw, L. T., and Sun, L. C.(2006). Scheduling framework for bandwidth-aware job grouping-based scheduling in grid computing. *Malaysian Journal of Computer Science*, 19(2):117–126.

[22] Koohi-Var, T. and Zahedi, M.(2017). Scientific workflow clustering based on motif discovery. *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, 7.

[23] Lathers, A., Su, M., Kulungowski, A., Lin, A., Mehta, G., Peltier, S., Deelman, E., and Ellisman, M.(2006). Enabling parallel scientific applications with workflow tools. In *Challenges of Large Applications in Distributed Environments (CLADE 2006)*.

[24] Liu, Q. and Liao, Y.(2009). Grouping-based fine-grained job scheduling in grid computing. In *First International Workshop on Education Technology and Computer Science*.

[25] Maechling, P., Deelman, E., Zhao, L., Graves, R., Mehta, G., Gupta, N., Mehringer, J., Kesselman, C., Callaghan, S., Okaya, D., Francoeur, H., Gupta, V., Cui, Y., Vahi, K., Jordan, T., and Field, E.(2007). SCEC CyberShake WorkflowsAutomating probabilistic seismic hazard analysis calculations. In Taylor, I., Deelman, E., Gannon, D., and Shields, M., editors, *Workflows for e-Science*, pages 143–163. Springer.

[26] Mats, R., Juve, G., Vahi, K., Callaghan, S., Mehta, G., and andEwa Deelman, P. J. M.(2012). Enabling large-scale scientific workflows on petascale resources using mpi master/worker. In *Proceedings of the 1st conference of the Extreme Science and Engineering Discovery Environment*.

[27] Muthuvelu, N., Chai, I., Chikkannan, E., and Buyya, R.(2010). On-line task granularity adaptation for dynamic grid applications. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 266–277. Springer.

[28] Muthuvelu, N., Chai, I., and Eswaran, C.(2008). An adaptive and parameterized job grouping algorithm for scheduling grid jobs. In *10th International Conference on Advanced Communication Technology (ICACT 2008)*, volume 2, pages 975 –980.

[29] Muthuvelu, N., Liu, J., Soe, N. L., Venugopal, S., Sulistio, A., and Buyya, R.(2005). A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global grids. In *Proceedings of the 2005 Australasian workshop on Grid computing and e-research*.

[30] Muthuvelu, N., Vecchiola, C., Chai, I., Chikkannan, E., and Buyya, R.(2013). Task granularity policies for deploying bag-of-task applications on global grids. *Future Generation Computer Systems*, 29(1):170 – 181. ¡ce:title¿Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures¡/ce:title¿.

[31] Nayyar, A.(2011a). Interoperability of cloud computing with web services. *International Journal of ElectroComputational World & Knowledge Interface*, 1(1).

[32] Nayyar, A.(2011b). Private virtual infrastructure (pvi) model for cloud computing. *International Journal of Software Engineering Research and Practices*, 1(1):10–14.

[33] Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M. R., Wipat, A., and Li, P.(2004). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054.

[34] Ostermann, S., Plankensteiner, K., Prodan, R., Fahringer, T., and Iosup, A.(2009). Workflow monitoring and analysis tool for askalon. In *Grid and Services Evolution*, pages 1–14. Springer.

[35] Sahni, J. and Vidyarthi, D. P.(2016). Workflow-and-Platform Aware task clustering for scientific workflow execution in Cloud environment. *Future Generation Computer Systems*.

[36] Sakellariou, R., Zhao, H., and Deelman, E.(2010). Mapping Workflows on Grid Resources: Experiments with the Montage Workflow. In Desprez, F., Getov, V., Priol, T., and Yahyapour, R., editors, *Grids P2P and Services Computing*, pages 119–132. Springer.

[37] Singh, G., Su, M., Vahi, K., Deelman, E., Berriman, B., Good, J., Katz, D. S., and Mehta, G.(2008). Workflow task clustering for best effort systems with pegasus. In *15th ACM Mardi Gras Conference*.

[38] Singh, S. P., Nayyar, A., Kumar, R., and Sharma, A.(2018). Fog computing: from architecture to edge computing and big data processing. *The Journal of Supercomputing*, pages 1–36.

[39] SIPHT (2018). SIPHT. `http://pegasus.isi.edu/applications/sipht`.

[40] Solanki, A. and Nayyar, A.(2019). Green internet of things (g-iot): Ict technologies, principles, applications, projects, and challenges. In *Handbook of Research on Big Data and the IoT*, pages 379–405. IGI Global.

[41] TeraGrid (2018). The TeraGrid Project. `http://www.teragrid.org`.

[42] Tomas, L., Caminero, B., and Carrioón, C.(2012). Improving grid resource usage: Metrics for measuring fragmentation. In *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '12)*, pages 352–359.

[43] Topcuoglu, H., Hariri, S., and Min-You, W.(2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274.

[44] USC Epigenome Center (2018). USC Epigenome Center. `http://epigenome.usc.edu`.

[45] Wieczorek, M., Prodan, R., and Fahringer, T.(2005). Scheduling of scientific workflows in the askalon grid environment. In *ACM SIGMOD Record*, volume 34, pages 56–62.

[46] Ying, H., Mingqiang, G., Xiangang, L., and Yong, L.(2009). A webgis load-balancing algorithm based on collaborative task clustering. *Environmental Science and Information Application Technology, International Conference on*, 3:736–739.

[47] Zhang, X., Qu, Y., and Xiao, L.(2000). Improving distributed workload performance by sharing both cpu and memory resources. In *Proceedings of 20th International Conference on Distributed Computing Systems, (ICDCS'2000*, pages 233–241.

[48] Zhao, H. and Li, X.(2009). Efficient grid task-bundle allocation using bargaining based self-adaptive auction. In *The 9th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*.

[49] Zhou, Z., Cheng, Z., Zhang, L.-J., Gaaloul, W., and Ning, K. (2018a). Scientific workflow clustering and recommendation leveraging layer hierarchical analysis. *IEEE Transactions on Services Computing*, 11(1):169–183.

[50] Zhou, Z., Cheng, Z., Zhang, L.-j., Gaaloul, W., and Ning, K. (2018b). Scientific Workflow Clustering and Recommendation Leveraging Layer Hierarchical Analysis. *IEEE Transactions on Services Computing*, 11(1):169–183.

Edited by: Anand Nayyar
Received: Feb 28, 2019
Accepted: Mar 16, 2019

DOI 10.12694/scpe.v20i2.1517
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 259–283. http://www.scpe.org © 2019 SCPE

GREEN AND SUSTAINABLE HIGH-PERFORMANCE COMPUTING WITH SMARTPHONE CROWD COMPUTING: BENEFITS, ENABLERS, AND CHALLENGES

PIJUSH KANTI DUTTA PRAMANIK[*], SAURABH PAL[†] AND PRASENJIT CHOUDHURY[‡]

Abstract. The introduction of the Internet of Things (IoT) and Big Data applications have garnered a massive amount of digital data. Processing and analysing these data demand vast computing resources, proportionately. The major downside of producing and using computing resources in such volumes is the deterioration of the Earth's environment. The production process of the electronic devices involves hazardous and toxic substances which not only harm human and other living beings health but also contaminate the water and soil. The production and operations of these computers in largescale also results in massive energy consumption and greenhouse gas generation. Moreover, the low use cycle of these devices produces a huge amount of not-easy-to-decompose e-waste. In this outlook, instead of buying new devices, it is advisable to use the existing resources to their fullest, which will minimize the environmental penalties of production and e-waste. This paper advocates for using smartphones and smartphone crowd computing (SCC) to ease off the use of PCs/laptops and centralized high-performance computers (HPCs) such as data centres and supercomputers. The paper aims to establish SCC as the most feasible computing system solution for sustainable computing. Detailed comparisons, in terms of environmental effects (e.g., energy consumption, greenhouse gas generation, etc.), between SCC and supercomputers and other green computing initiatives such as Grid and Cloud Computing, are presented. The key enablers of SCC are identified and discussed. One of them is today's computationally powerful smartphones. A comprehensive statistical survey of the various commercial CPUs, GPUs, SoCs for smartphones is presented confirming the capability of the SCC as an alternative to HPC. The challenges involved in realizing SCC are also considered. One of the major challenges is handling the issue of limited battery in smartphones. The reasons for battery drain are recognized with probable measures. An exhaustive survey is presented on the present and optimistic future of the continuous improvement and research on different aspects of smartphone battery and other alternative power sources which will allow users to use their smartphones for SCC without worrying about the battery running out.

Key words: Green computing, Sustainable computing, Mobile computing, Grid computing, Smartphone computing, HPC, Cloud computing, Data centre, Energy efficiency, Smartphone battery, Battery research, Future battery, Non-conventional energy, Wireless charging, Alternative power source

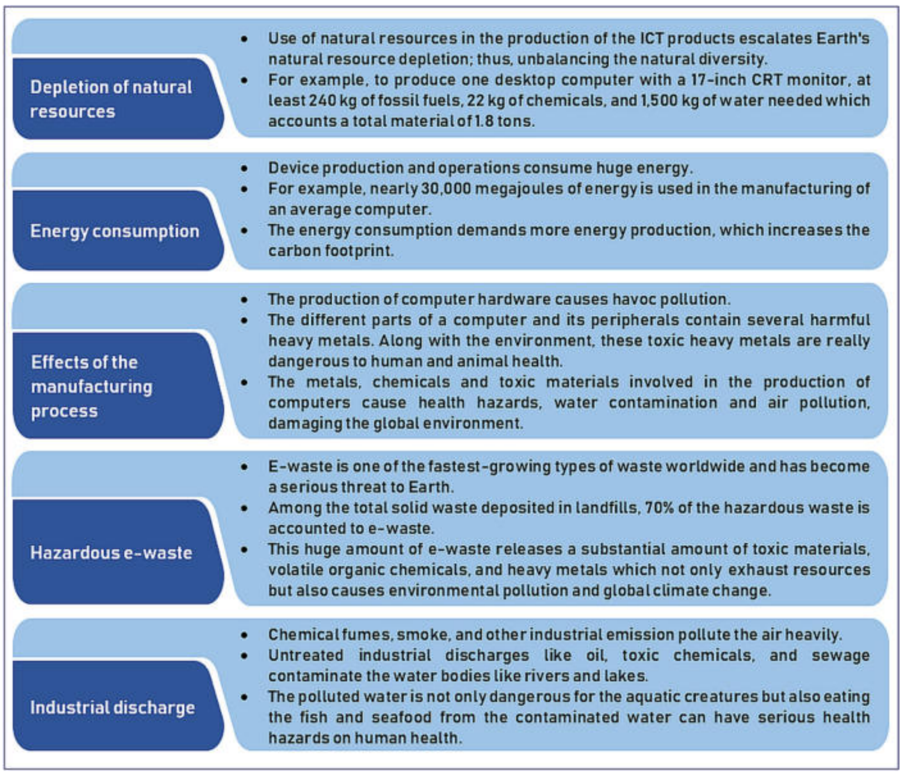
AMS subject classifications. 68M14, 68W10

1. Introduction.

In line with the massive increase in data from innumerable sources, the need for high-performance computing (HPC) has increased enormously. But this has impacted the global environment badly through various forms such as excessive energy consumption, carbon and greenhouse gas emission, heat generation, use of toxic materials in production, harmful industrial discharge, non-degradable waste generation, etc. Device production and operations consume huge energy. For example, nearly 30,000 megajoules of energy is used in the manufacturing of an average computer. The energy consumption demands more energy production, which increases the carbon footprint [64]. More requirement of computers leads to more productions and more uses of computers which means more environmental hazards and pollution. The various environmental hazards caused by the production and operations of computing devices are listed in Fig 1.

To mitigate the environmental hazards due to computing devices, we need to concentrate on green and sustainable computing. Several computational measures, as shown in Fig 1, are adopted that helps in minimising the energy consumption of the computers. But this is not sufficient for realising sustainable computing absolutely. One of the strategies to attain sustainable computing to utilise the existing resources optimally and fully which will minimise the requirement of new devices which ultimately reduce the problems due to the production process as well as the e-waste. Several approaches have been opted in this regard as mentioned in Table 1. Among them, Grid and Cloud computing are the most prominent initiatives which have minimised the requirement of owning personal computer systems considerably. They have also replaced the need for centralised HPC systems such as supercomputers and mainframes to some extent. Though Grid computing intends to fully utilise the existing resources (desktops), Cloud computing is not that successful in this aspect. Cloud computing needs additional resources to provide cloud services. The centralised resources such as data centres, at the cloud service providers end, consumes massive energy, leading to greenhouse gas emission substantially.

[*]National Institute of Technology, Durgapur, India (pijushjld@yahoo.co.in)
[†]Bengal Institute of Technology, Kolkata, India
[‡]National Institute of Technology, Durgapur, India



Depletion of natural resources	• Use of natural resources in the production of the ICT products escalates Earth's natural resource depletion; thus, unbalancing the natural diversity. • For example, to produce one desktop computer with a 17-inch CRT monitor, at least 240 kg of fossil fuels, 22 kg of chemicals, and 1,500 kg of water needed which accounts a total material of 1.8 tons.
Energy consumption	• Device production and operations consume huge energy. • For example, nearly 30,000 megajoules of energy is used in the manufacturing of an average computer. • The energy consumption demands more energy production, which increases the carbon footprint.
Effects of the manufacturing process	• The production of computer hardware causes havoc pollution. • The different parts of a computer and its peripherals contain several harmful heavy metals. Along with the environment, these toxic heavy metals are really dangerous to human and animal health. • The metals, chemicals and toxic materials involved in the production of computers cause health hazards, water contamination and air pollution, damaging the global environment.
Hazardous e-waste	• E-waste is one of the fastest-growing types of waste worldwide and has become a serious threat to Earth. • Among the total solid waste deposited in landfills, 70% of the hazardous waste is accounted to e-waste. • This huge amount of e-waste releases a substantial amount of toxic materials, volatile organic chemicals, and heavy metals which not only exhaust resources but also causes environmental pollution and global climate change.
Industrial discharge	• Chemical fumes, smoke, and other industrial emission pollute the air heavily. • Untreated industrial discharges like oil, toxic chemicals, and sewage contaminate the water bodies like rivers and lakes. • The polluted water is not only dangerous for the aquatic creatures but also eating the fish and seafood from the contaminated water can have serious health hazards on human health.

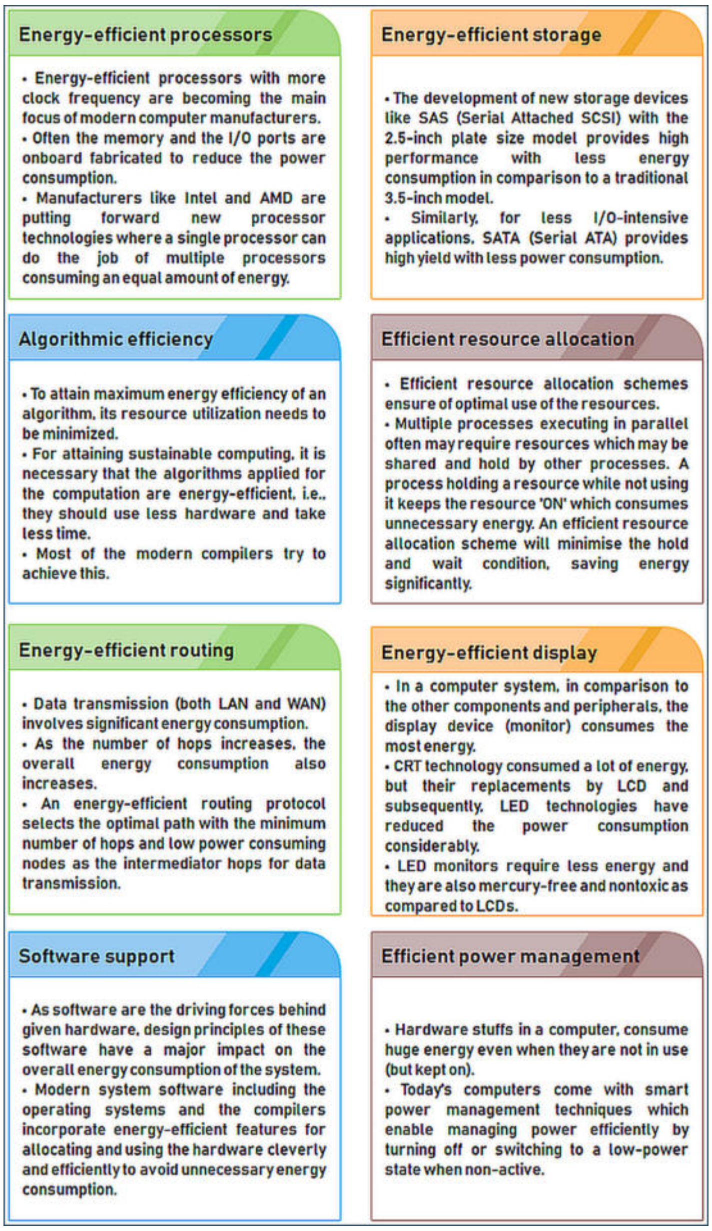
FIG. 1.1. *Environmental hazards of the computing devices.*

In fact, data centres capture the one-third share of the total energy consumption of ICT. A well-known report [45] of the year 2010 stated that the electricity consumption by data centres had risen by 56% in five years; whereas, during the same period, the overall increase in U.S. electricity usage was only 36%. This statistic reflects the seriousness of the energy requirements of the data centres, which has become more severe in recent years. If the right measures are not taken in due course, data centres are sure to become a grave threat to the environment. Many companies and organizations are doing their best to tackle the energy appetite of the data centres by adopting different measures. For example, Apple has the largest solar installation for powering its data centre. Facebooks data centre in Iowa uses wind energy to meet the energy requirements of its data centre. Google has placed its data centre in Hamina, Finland under the sea for cooling, thus curbing the energy requirements significantly. Google is using AI for optimizing the energy requirements and consumptions in its data centres.

The problem with Grid computing is that the desktops are losing popularity; in fact, the same for laptops. Instead, smartphones are gaining huge acceptance as the new computer with the computing power they offer thanks to the power-packed hardware. The technological progress of smartphones such as powerful SoCs with multicore CPUs and GPUs has made them favourable as the primary computing device to many people. Industries are also showing interest in this direction. Initiatives such as Microsofts Continuum[1] and Samsungs DeX[2] are striving to bring the desktop experience on the smartphone. Microsoft has endeavoured to run its full version of Windows 10 on the ARM chipsets, the most popular chipset for smartphones.

And the great thing about smartphones is that they have become indispensable to our lifestyle. It is not feasible to restrain ourselves from using that. So, why dont we use these devices of its optimal potential, i.e., for computing purposes as well? If these abundantly available public-owned powerful smartphones are used as

[1]https://www.microsoft.com/en-in/windows/continuum
[2]https://www.samsung.com/global/galaxy/apps/samsung-dex/



Energy-efficient processors

• Energy-efficient processors with more clock frequency are becoming the main focus of modern computer manufacturers.
• Often the memory and the I/O ports are onboard fabricated to reduce the power consumption.
• Manufacturers like Intel and AMD are putting forward new processor technologies where a single processor can do the job of multiple processors consuming an equal amount of energy.

Energy-efficient storage

• The development of new storage devices like SAS (Serial Attached SCSI) with the 2.5-inch plate size model provides high performance with less energy consumption in comparison to a traditional 3.5-inch model.
• Similarly, for less I/O-intensive applications, SATA (Serial ATA) provides high yield with less power consumption.

Algorithmic efficiency

• To attain maximum energy efficiency of an algorithm, its resource utilization needs to be minimized.
• For attaining sustainable computing, it is necessary that the algorithms applied for the computation are energy-efficient, i.e., they should use less hardware and take less time.
• Most of the modern compilers try to achieve this.

Efficient resource allocation

• Efficient resource allocation schemes ensure of optimal use of the resources.
• Multiple processes executing in parallel often may require resources which may be shared and hold by other processes. A process holding a resource while not using it keeps the resource 'ON' which consumes unnecessary energy. An efficient resource allocation scheme will minimise the hold and wait condition, saving energy significantly.

Energy-efficient routing

• Data transmission (both LAN and WAN) involves significant energy consumption.
• As the number of hops increases, the overall energy consumption also increases.
• An energy-efficient routing protocol selects the optimal path with the minimum number of hops and low power consuming nodes as the intermediator hops for data transmission.

Energy-efficient display

• In a computer system, in comparison to the other components and peripherals, the display device (monitor) consumes the most energy.
• CRT technology consumed a lot of energy, but their replacements by LCD and subsequently, LED technologies have reduced the power consumption considerably.
• LED monitors require less energy and they are also mercury-free and nontoxic as compared to LCDs.

Software support

• As software are the driving forces behind given hardware, design principles of these software have a major impact on the overall energy consumption of the system.
• Modern system software including the operating systems and the compilers incorporate energy-efficient features for allocating and using the hardware cleverly and efficiently to avoid unnecessary energy consumption.

Efficient power management

• Hardware stuffs in a computer, consume huge energy even when they are not in use (but kept on).
• Today's computers come with smart power management techniques which enable managing power efficiently by turning off or switching to a low-power state when non-active.

FIG. 1.2. *Measures adopted to minimise energy consumption of the computing devices.*

computing resources by forming, like desktop grids, a grid of smartphones, the generated computing power can well be compared to other HPC systems. We call this computing paradigm as smartphone crowd computing (SCC). SCC refers to the approach of utilizing the public-owned smartphones (crowd of smartphones) to yield an HPC facility anywhere and ad hoc basis. A number of powerful smartphone devices, collectively, can offer huge computing capability. A satisfactory HPC may be achieved by making a grid of smartphones. The cumulative computing power achieved by such grids of smartphones can tail off the dependency on the data centres and low-end supercomputers as well [63].

In this paper, we advocate for optimized utilization of smartphones as primary computing devices with the aim of reducing the two-thirds share of ICT energy consumption on the count of the use of computers and data centres. The use of smartphones will lessen the environmental hazards because the manufacturing process and device operation will be minimized significantly. Due to smaller in size, the less raw material will be used in production means less exposure to harmful elements and less pollution due to e-waste. If organizations can avoid buying computing resources, they can save a significant IT investment and operational cost. And less device in production and operation means a direct benefit to the environment.

Contribution of the paper: This paper offers a fourfold contribution as mentioned below:

- The environmental hazards of the growing computing devices and centralised HPC systems are impeccably identified.
- SCC has been attempted to be established as a better alternative to the existing HPC systems in terms of sustainable computing.
- A detailed survey is presented on the modern smartphones computing capability, highlighting the capabilities and specifications of the prominent commercial CPUs, GPUs, and SoCs for mobile devices.
- An exclusive and state-of-the-art survey has been presented on the recent developments (research and commercial) on different aspects of smartphone batteries.

Organisation of the paper: Section 2 discusses the environmental hazardous factors especially for centralised HPCs, in addition to the usual hazards of computers as mentioned in Fig. 1. Section 3 asserts the focal essence of this paper. It establishes the role of SCC in sustainable computing. In Section 4, the major factors that are supposed to make the idea of SCC feasible are identified. To endorse that, a comprehensive statistical review of todays smartphones processing capability is presented in this section. The major challenges faced in the successful deployment of SCC are recognized in Section 5. Among others, the foremost challenge is the battery concern. The major reasons for battery drain are identified in Section 6. In this section, a detailed survey is projected that covers the continuous research and improvement on different aspects of smartphone battery which should help users to use their smartphone without worrying about the battery drain. The recent developments, in different segments, towards enhanced energy sources for smartphones are also discussed. And finally, Section 7 concludes the paper.

2. Factors Behind Environmental Hazards in Centralized HPC Systems like Cloud, Mainframe, and Supercomputers. Centralized computing facilities such as Cloud services and data centres, mainframes, and supercomputers have become a real threat to the environment. For instance, the data centres are responsible for 17% of the global carbon footprint [61]. These large computing systems are notorious for energy consumption. It is projected that data centres in the US only would be consuming 73 kWh in 2020. Nearly 30 billion watts of electric power is needed to run the data centres. Among the amount of electricity consumed by data centres, the servers waste 90% of their total power consumption in keeping the systems running for the entire day. Besides the operational energy consumption, there are several other factors that are associated with environmental hazards of the big centralised computing systems, as mentioned below [46]:

Coolant: Computing and allied infrastructures in the data centre produce enormous heat, and this requires heavy cooling. Most of the air conditioning or cooling system uses coolant for quick and deep cooling. The coolants used earlier for air conditioning systems are freon/halocarbon or chlorofluorocarbon. These coolants are mildly toxic in nature and have caused ozone depletion significantly. The new coolants hydrofluorocarbon currently used in all air conditioning system is not risky to the ozone layer, but they significantly cause atmospheric warming. This coolant traps the heat within the atmosphere and causes global warming.

Batteries: Data centres require huge batteries for power backup. These batteries make sure the power flow is constant, and provide an electric power supply for a few minutes to hours based on the power cut or power fluctuation. These batteries have a limited life span of 3 to 4 years. The batteries used more often are the lead-acid battery. These batteries may cause an adverse effect on human health and the environment. The fumes emitted from the battery are dangerous to human health. Whereas the lead used in batteries is highly toxic, and if they are not properly disposed of, they may contaminate the soil and water. Presently, in most power backup systems, dry batteries are used. These batteries use heavy metals like mercury, nickel, cadmium, and silver. The metals, mercury, nickel, and cadmium are toxic and have a fatal effect on human health. These batteries cannot be dumped into the open and need to be properly recycled. It is necessary that the batteries are properly disposed of. There are multiple companies which recycle UPS batteries. 80% of leading manufacturing units in the U.S. undergo battery recycling.

Cleaning materials: Dusting and cleaning are important in data centres in order to operate efficiently.

Table 1.1
Sustainable computing approaches, their advantages and problems in terms of sustainability

	Working principles	**Environmental advantages**	**Problems**
Grid computing	The unused computing resources of the connected computers are used opportunistically providing a massive virtual computing facility [62]	Utilising of existing idle resources facilitates less energy consumption and minimise environmental hazards due to the manufacturing and operation of the computing systems required otherwise.	Needs fast and reliable LAN and WAN connections. In the volunteered grid, it is difficult to motivate the resource owner to lend their resources. In the case of the non-volunteered grid (e.g., commercial grids), the services might be costly. Not only setting up and managing the grid resources but also accessing them often require expertise.
Cloud computing	Facilitates a shared pool of configurable computing resources along with quality services which can be rapidly provided on-demand basis [65]	Eliminates the need for private computers, servers, and data centres which has significantly contributed to energy saving.	The data centres, comprising the large servers and computers, and associated cooling systems consume massive power. Accessing the cloud service depends on the internet connection. The instability/unreliability and unavailability of connection hold back accessing the cloud. Involves security and privacy issues. It is true that the cloud computing eradicates the big upfront investment on computing resources but accessing the right service is not that cheap either. Involves data transfer cost. Since cloud services are typically generic they lack in flexibility. The user/client has minimal control of their own applications as the cloud service infrastructure is entirely owned, managed and monitored by the service provider. Switching or migrating to a different cloud service provider is often complex or infeasible.
Serverless computing	Consumers borrow a third-party server when needed; thus, doing away with purchasing and maintaining own server exclusively [36].	Saves the energy requirement for running privately-owned servers. It also scales down quickly by shutting resources down when they are not in use. This saves a lot of energy consumption. Serverless codes not necessarily be run in any specific server. Hence, the serverless applications can be deployed in the edge of the network, i.e., close to the end users [18]. This will not only reduce the latency but also saves a significant amount of energy by eliminating the need for unnecessary data transmission [65].	Not suitable for real-time applications which require low latencies. Also, not suitable for applications which need long execution times. Everything operates in stateless fashion; hence, handling state using stateless functions is a real issue. After some time of being idle, the function will require to go through a cold start which not only takes up to a few seconds but also consumes energy.
Using terminal servers	The use of terminal server along with the thin clients gives users a feel of computation being taking place in the very same terminal, while the actual computation takesplace in the terminal server.	The processing and storage requirements for client machines are minimal because a terminal server hosts all the application logic which also runs on the server. The thin client uses up to the 1/8th amount of energy in comparison to workstations and thus considerably reduces energy consumption.	Running applications on a remote server always involve performance issues. There are chances for terminal servers getting bottlenecked with overloaded requests. Hence, the terminal server needs to be powerful enough to be able to handle all connections. If the terminal server is not backed up, there is a high risk of downtime due to a single point of failure. If the communication network is not reliable, the system will be affected harshly.
Virtualization	A single physical server is logically partitioned into multiple virtual servers or server instances, sharing the same hardware resources and allows processing multiple applications or jobs on different virtual servers.	Effective resource utilisation leads to fewer production and less e-wastage. A single physical server acting as multiple server instances consumes considerably less energy in comparison to separate dedicated servers. It requires quantitatively less hardware to run similar application than dedicated systems which leads to fewer device production and less e-wastage. The absence of the usage of the local hardware or software cuts the overall energy consumption.	The required hardware specification (e.g., memory, processor, etc.)is much higher for the same task executed in a native computer. Involves complex troubleshooting, in case of failure. Degraded performance than a physical server. Suffers from availability issue which discourages using virtual servers for mission-critical applications. Has major security issues.

Dusting and cleaning require special material that minimizes static discharge and are safe for electronics. Varieties of cleaning solution are available, and most of them are toxic as they contain bleach, ammonia, or chlorine. These toxic cleaning solutions have an adverse effect on human health.

Diesel fuel: Another measure of overcoming power failure is the diesel fuel-based power generator. In a location where there is a recurrent power failure, the data centres heavily depend upon the power generators for a constant supply of power. Diesel fuel-based power backup is the primary solution for powering data centres since battery technology is incapable of producing power for long times. Diesel fuel produces an enormous amount of CO_2 and other chemicals which have an adverse effect on human and does global warming.

Electronic waste: Electronic equipment have a finite lifespan. Even for functional equipment, the performance degrades with time. Conventionally, the IT products in business firms and data centres such as a server rack, UPS, desktop computers, servers, monitors, hard drives, etc., are replaced after every 3 to 5 years. Globally, the scrapped IT equipment for the last 20 years are enormously huge, and it is increasing day-by-day. The e-wastes are harmful to the environment if dumped on the open landfills. The monitor, CPU, UPS, etc., are made up of materials which have negative effects on the environment. It is the responsibility of the business organizations to either reuse or sell/donate the reusable equipment or recycle them properly.

Fire suppression: It is important to have data intact and safe with the service providers. Due to the fact, the data centres are dealing with high voltage, and hence, there is a chance of short-circuiting and fire. All data centres carry extensive and fast fire suppression system which may be environmentally negative. Various chemicals employed for the fire system may be harmful to the environment contributing to effects like ozone layer depletion and global warming. These chemicals are toxic and may find its way to underground water or to rivers, thus, contaminating the water resources. However, as these systems are not used or tested regularly, the chemical harm is minimized.

Packaging: Business organizations like IT firms and data centres import huge IT equipment every year. Customers and data centres ship equipment and supplies to and from data centres frequently. The equipment comes with packaging material like boxes, cardboard, plastic bags, foams, thermocols adds huge waste every year. A large data centre processes literally tons of boxes each year. Packaging materials like cardboard boxes can be recycled and are biodegradable, but other materials like foams, thermocols, plastic bag, and plastic support accessories are nonbiodegradable and need proper recycling. Dumping these on the open area may harm the environment.

Office zones: IT business firm and data centre office area contribute a lot to emissions and wastes. This includes wastes due to the inappropriate use of electricity and water resources. Running air conditioners, heating equipment, and lights throughout the office for the entire day, even if it is not required, causes considerable electricity wastage. Similarly, unmonitored use of water resources in washrooms and kitchen lead to wastage of water. Besides, daily office chores produce lots of paper, plastics, and packaging wastes. Floor cleaning, glass pane, computers, and carpets also produce chemical wastes. Even the fluorescent bulbs used by offices also carry toxic matters like mercury, lead, etc., and may cause environmental and health hazards.

Overall, it can be said that the centralised HPC systems have a negative impact with respect to nature and natural resources. But as said there is always room for improvement, which may range from minimizing the toxic chemicals used to practising the policies of reuse-reduce-recycle across wherever possible. As technology is developing, new products are coming which causes less emission, less environment hazard and lowers the greenhouse gas emissions.

3. Smartphone Crowd Computing as a Solution Towards Sustainable HPC. Whatsoever the negative impacts of computers have on the environment, we cannot head them off from our livings. We need them in every step of our daily life. Actually, we need more and more powerful computers day by day for various purposes. In view of that, we need to consider seriously to minimize the environmental externalities of producing and using computers.

On the other hand, smartphones have become a part of our daily life, and they are getting more resource-packed and hence, computationally more powerful (see Sect. 4.2). So, it is a good idea to use them as computing devices because smartphones have less adverse effects on the environment compared to desktops and laptops. Actually, to many people, smartphones have already become the primary computing device [63]. Because of portability and easy to use, people are fulfilling most of the daily computing needs using smartphones.

Since the smartphones are getting close to the sophisticated computers in terms of resources and computing ability, much like desktop Grid Computing, if a number of such smartphones are connected through a network and make them act as a virtual single unit of computing resources, it can match the power of high-end powerful computing systems such as supercomputers and data centres. Like volunteer computing, instead of buying the required number of smartphones, public-owned smartphones are exploited on a need basis. Hence, this computing paradigm is termed as smartphone crowd computing (SCC) [63, 67].

3.1. Smartphone Crowd Computing. SCC is a distributive computing framework where a large job is divided and distributed to the peoples smartphones (hence, crowd computing) to be executed. The philosophy of SCC is to combine computation power of numerous distributed smartphones to escalate the overall computation power. SCC paradigm shares the CPU cycle of multiple smartphones in a voluntary or involuntary basis to resolve a high computational task. The concept is to process a computationally big task by segmenting the task into smaller microtasks, and each micro-task is processed on the smartphones individually. The processing is typically carried out opportunistically, i.e., whenever the host smartphone is idle, the crowd computing application installed on the smartphone will assign the task to the smartphone processor (CPU or GPU). Even though the computation power of smartphone devices is less, the cumulative processing power of smartphones is quite high enough to execute complex computational jobs. Fig 3.1 lists the highlights of SCC.

In SCC, a designated coordinator, may be fixed (a desktop computer) or mobile (a smartphone), hosts the crowd computing application. The coordinator is responsible for dividing and distributing the job, finding suitable resources (smartphones), allocating and scheduling the jobs to the smartphones, monitoring for fault-tolerance and collecting the results from each resource. The coordinator keeps microtasks in the job pool, ready to be dispatched. For distributing the tasks, available crowd devices (smartphones) are searched, and a set of suitable devices are being selected as crowdworkers (computing resources) for processing the jobs. After the execution of these tasks, each node submits the completed task with results to a centralized master device, where all the results are gathered, checked for error to get the final result. It is the job of the master device (coordinator) to check the error and validity of each task execution.

Since the devices are mobile, the reliability of an available device for a long time is very thin. The coordinator makes a task group among smartphones which either voluntarily or involuntarily joins the group. SCC provides quite a flexibility in terms of job allocation and execution as a small task group could be established, comprising a few available smartphones. Even without an internet connection, the smartphones in the task group can communicate using Hotspots or Bluetooth. This makes SCC as the ideal alternative of centralised HPCs where these HPCs are not available or not accessible. The ubiquity and flexibility of SCC is an ideal candidate for providing the computing facility to applications such as pervasive computing [68] and cognitive IoT [66], where the data need to be processed near the data source as well as sink.

3.2. How Can Smartphone Crowd Computing Minimize Environmental Hazards?. Object-oriented programming brought the revolution in software development by introducing the concepts of reusability and polymorphism, which allow software modules to be used multiple times for multiple purposes. This saves a significant amount of man-hours and cost. The same is true for crowd computing. Already existing devices are used for computational and other purposes. This reduces the requirement of buying IT infrastructure separately because the public will buy phones for their own purposes, anyway. Moreover, the utilization of already existing smartphones to achieve SCC instead of traditional HPC will reduce the requirement for dedicated large computers.

If smartphones and SCC are employed in a wide-ranging computing requirement, due to the substantial reuse, the need for new devices will be much lower. That means unnecessary production is avoided which will, in turn, minimize the environmental externalities of producing computing devices. The reduction in the production, operation, and degradation of unnecessary devices will reduce the amount of global e-waste considerably. In addition to that, being much smaller in size, smartphones produce less e-waste.

Using smartphones for computations will curb energy requirements significantly. The smartphone processor architectures offer far more energy efficiency than those of desktops and laptops. The energy consumption of a standard smartphones ranges from a few to 10 kWh per year. Therefore, total energy consumption is 10 TWh per year considering one billion smartphones are in operation worldwide. This is only 1% of the total energy consumed by ICT which is typically on the order of 1,000 TWh per year [32].

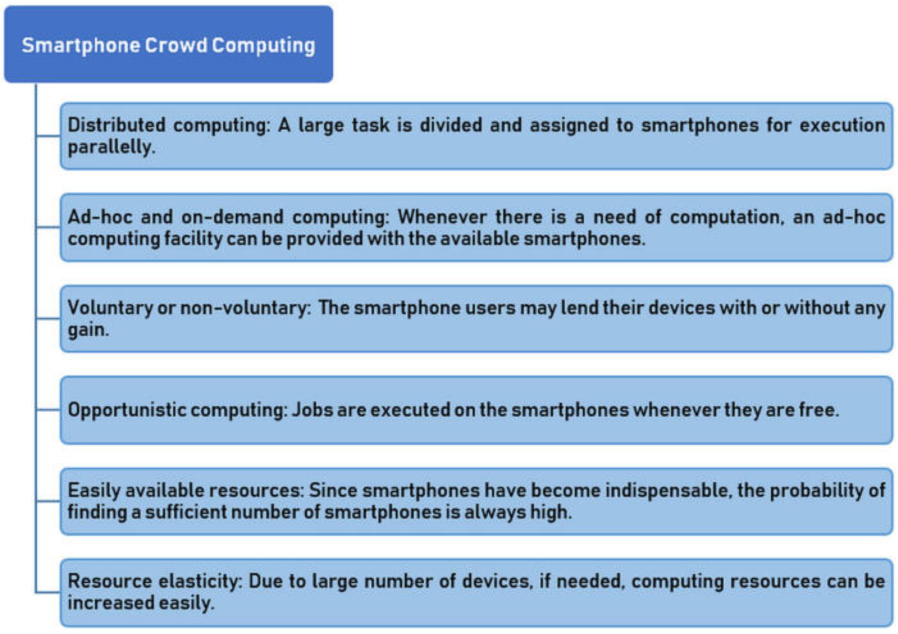

Smartphone Crowd Computing
Distributed computing: A large task is divided and assigned to smartphones for execution parallelly.
Ad-hoc and on-demand computing: Whenever there is a need of computation, an ad-hoc computing facility can be provided with the available smartphones.
Voluntary or non-voluntary: The smartphone users may lend their devices with or without any gain.
Opportunistic computing: Jobs are executed on the smartphones whenever they are free.
Easily available resources: Since smartphones have become indispensable, the probability of finding a sufficient number of smartphones is always high.
Resource elasticity: Due to large number of devices, if needed, computing resources can be increased easily.

Fig. 3.1. *Highlights of Smartphone Crowd Computing*

Hence, it is apparent that smartphones and SCC will significantly help in protecting the environment along with saving resources and minimized organizational IT expenditure.

3.3. Environment-friendliness of Smartphone Crowd Computing in Comparison to Other HPC Systems. In line with our argument that SCC is a viable alternative to the costly HPC systems, in this section, we shall check out how SCC is more environment-friendlier than other HPC systems viz. desktop Grid Computing, data centre, and supercomputers.

Due to their computing capacity and power, the energy requirements of supercomputers are gigantic and might well be equivalent to that of a small city. The correct response relates to electricity; specifically, 17.8 megawatts (MW) of power is required to run Tianhe-2, one of the Top500 ranking supercomputer, boasting 33.9-petaflop through 3.12-million processors. An exaflop (1,000 petaflops) computer needs approximately 500 MW, which is equivalent of the total output of an average-size coal plant, and enough electricity to cater the needs of all the households in a city like San Francisco [35].

In recent years, the computing services offered through Cloud Computing have got tremendous popularity. People can rent computing resources on usage and requirement basis. The cloud service providers maintain big data centres to cater to the computing resource need of the clients. A data centre, abstractly, can be described as an abundant number of computers stacked together. To make the cloud service available 24x7, these computers are kept ON throughout the day which makes them very hot. As a result, a huge amount of power is consumed not only to run these computers but also to keep them cool. About 30 billion watts of electricity is needed to run the data centres (comparable to the electricity generated from 30 nuclear power plants) which is nearly 17% of the total carbon footprint caused by technology [61]. Data centres consumed 416.2 TWh of electricity in the year 2015 which is roughly 3% of the global electricity supply [9]. A single data centre can consume more power than an average town. To provide uninterrupted power supply in case of power failure, the data centres run generators that emit diesel exhaust. Todays data centres cause approximately 0.3% of overall carbon emissions [43], which is equivalent to the carbon footprint generated by the airline industry [9]. Of the total global greenhouse gas emissions, the power-hungry data centres account for nearly 2%. This is putting an immense impact on the environment leading to global warming. The bad news is, every four years, this energy requirement is getting doubled and the total energy requirement of the data centres, globally, will increase threefold in the next decade. By 2025, data centres are expected to use 20% of the worlds energy

[5]. The efficiency of the data centre is measured in terms of Power Usage Effectiveness (PUE). PUE compares the non-computing energy to the amount of energy to power actual machines. Data centres operate at 70% of overhead energy. It means another 0.7 units are used behind the infrastructure of the data centres. So, the total PUE goes up to 1.7 [61]. Typically, the PUE of the common data centres is about 2.0 [43].

As mentioned in Sect. 1, the desktop Grid is an affordable option for sustainable computing. Here, the existing devices are used as computing resources instead of buying dedicated HPC. This helps the environment as this will minimize the requirement for producing extra dedicated HPC systems. Grid Computing can lower the environmental externalities of supercomputers and data centres considerably. SCC promises to minimize it further.

Due to the much smaller size, smartphones have much less negative environmental impact than desktops and laptops and obviously than supercomputers and data centres. So, a grid of smartphones (SCC) will be more environment-friendly. As people will buy smartphones anyway, SCC can use the existing devices without requiring additional device production. The environmental advantages of SCC can be summarized as follows:

- No need for explicit production of computing devices as people will use smartphones anyway.
- Production of smartphones is much environment-friendly compared to large computers.
- Due to the small size, the e-waste will be lesser and can be handled more efficiently.
- No dedicated cooling systems are required.
- No power backups, such as large batteries and generators are required.
- Smartphone processors are typically energy-efficient. As a result, they consume much less energy than other computing systems to perform the same operation.

Table 3.1 statistically demonstrates the advantages of smartphones, in terms of environment-friendliness, compared to desktops and laptops (Grid Computing), data centres, and supercomputers.

The excessive number of smartphones may be advantageous to SCC, but it is really worrisome considering the amount of e-waste generated. Therefore, it is extremely crucial to opt for the proper disposal of discarded devices and try to recycle as much as possible. Fig. 3.2 lists the responsibilities of the smartphone user suggesting what to be done when they discard their smartphones.

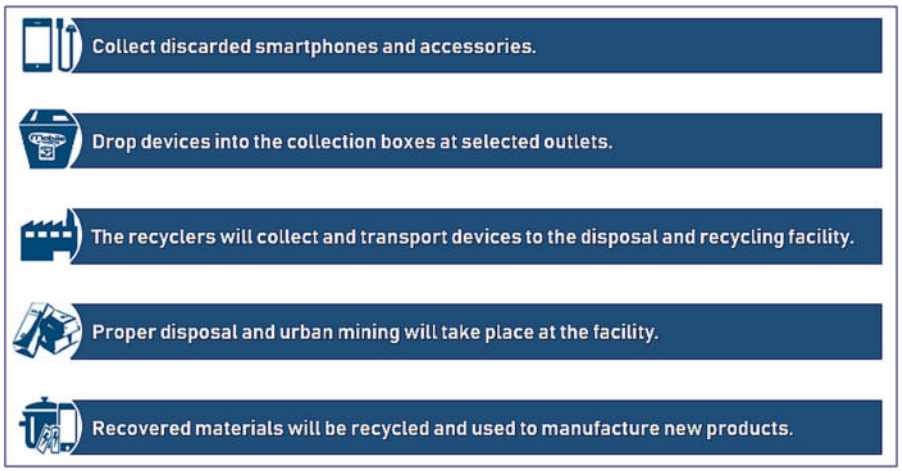

Collect discarded smartphones and accessories.
Drop devices into the collection boxes at selected outlets.
The recyclers will collect and transport devices to the disposal and recycling facility.
Proper disposal and urban mining will take place at the facility.
Recovered materials will be recycled and used to manufacture new products.

FIG. 3.2. *Smartphone users' responsibility towards e-waste management [51].*

4. The Enablers of Smartphone Crowd Computing. In this section, we shall look into the key factors that have helped SCC to be a feasible HPC solution.

4.1. The Mass Adoption of Smartphones. According to the recent research market statistics, globally smartphone shipments had reached 1.55 billion [75]. It is expected that by 2020, the shipment would reach 6 billion [76, 34]. Fig 4.1 shows the estimation of the number of smartphones that are going to be shipped globally from 2018 to 2022. Alone in India, in 2017, the number of mobile users had crossed 300 million [38].

Table 3.1
Environmental externalities comparison of smartphones with data centres, supercomputers, and Grid Computing (desktops and laptops)

			Grid Computing		
	Data centre	**Super-computer**	**Desktop**	**Laptop**	**Smartphone**
Energy consumption	200 TWh/year [43].	17.8 MW for Tianhe-2, the 33.9-petaflop supercomputer with 3.12-million processors [35].	100-150 Watts/hour, 600 kWh/year [25].	60 Watts/hour, 300-150 kWh/year [25].	1.5-3 Wh. An average phone needs 2kWh/year.
CO_2 emission in the manufacturing process	171,630 kg CO_2 [39]. Around 0.3% of overall carbon emissions [43].	0.175 million kg/year per supercomputer (equivalent capacity of 1000 PCs).	380 kg/desktop [31].	227 to 270 kg/laptop [1].	16 kg/year [10]. An average mobile emits 35 kgs of carbon while manufacturing [2].
Other environmental hazards [40, 55, 28]	Along with the common hazardous materials such as Fe, Cu, Al, Ag, Au, Pt, Pd, Pb, Hg, As, Cd, Se and hexavalent Cr and BFRs, other harmful elements such as ethylene/propylene glycol for cooling systems, diesel fuel for backup generators, lead-acid batteries for UPSs, and compressed gases for fire suppression makes data centre real peril to the environment.	Same as data centres.	The metals contained in PCs commonly include Al, Ag, As, Au, Ba, Be, BFR, Cd, Co, Cr, Cu, Fe, Ga, Hg, Mn, Pb, Pd, Pt, PVC, Sb, Se, and Zn. Most of them are really hazardous and contaminate soil, water, and air, if not properly disposed of.	Almost all the hazardous elements of desktops are also found in a laptop, but the quantity is less as laptops are typically smaller than desktops.	The hazardous metals such as Al, Ag, Au, Cu, Fe, Pb, Hg,Cd, etc. are needed in smartphone manufacturing also, but in much less quantity than desktops and laptops.
E-waste generated	32360 metric tons of e-waste in 2018.	9.3 million tons/year [19].	41698.8 metric tons [79].	3230 metric tons [79].	In India, out of 650 million mobile users, 40% have changed their phones in 2017, generating huge e-waste [52]. In the USA, yearly, nearly 150 million mobile phones are discarded.
Weight fraction of materials (%) [49]	N.A.^{3}	N.A.$^{??}$	47.2 Fe, 0.9 Cu, 2.8 plastic, 9.4 PCB.	19.5 Fe, 2.4 Al, 1.0 Cu, 25.8 plastic, 13.7 PCB, 14.4 battery.	0.8 Fe, 0.3 Cu, 37.6 plastic, 30.3 PCB, 20.4 battery.
E-waste decomposition	Decomposing is challenging as a huge volume of e-waste generated due to a large scale of components. Require large dumping ground; risk of toxic metals and chemicals; and contamination risk. But, since the data centres are generally owned by big companies/institutes, they are expected to follow the systematic decomposing and recycling process.	Same as data centres.	As the number of users is very large scale, the proper and systematic decomposition of e-waste is really difficult. Most of the computers are public-owned or owned by small organizations. Most of them do not follow the proper decomposition and recycling processes.	The same problem, but moderate due to less equipment as compared to desktop.	The continuous growth in smartphone users with very brief use-cycles is a great challenge in terms of decomposing and recycling as the lack of awareness and eagerness among the public. But if the civic authorities take active roles and are able to convince people the necessity of proper disposal of discarded devices, the problem can be tackled.

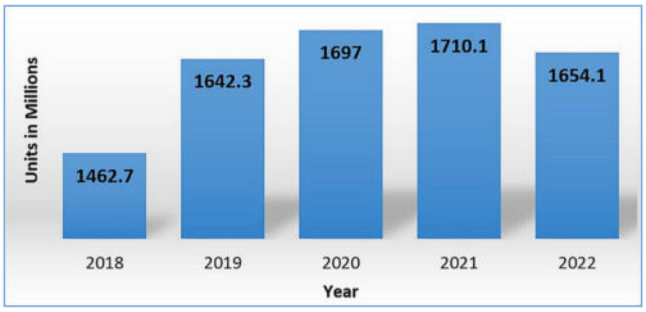


FIG. 4.1. *Worldwide smartphone shipments forecast from 2018 to 2022 [76].*

The number of active smartphone connections, worldwide, will reach 6 billion by 2020 [34]. The huge adoption of smart mobile devices across the world has put a big platform for crowd computing [63, 67].

Factors driving the smartphone market: There is a number of factors influencing the growth of smartphone use in the global market as mentioned below [34]:

- Rapidly falling price of smartphones has accelerated the customers to move from basic and standard feature phones to smartphones.
- Developing smartphone technology has a reason for the increase in the sale of low-end smartphones.
- The increasing availability of the highspeed 3G/4G spectrum with increased mobile broadband connections all around the world.
- The availability of highspeed data-centric services and low tariffs have increased the adoption of smartphones in both developed and developing societies. Further, the availability of cheap data tariffs tailored as per the customers needs has also reasoned for smartphone adoption in developing countries.
- The concept of tailoring the data tariffs for cost-conscious prepaid consumers can be linked with the selling rate of smartphones.
- Efficient retail channels and supply chain have helped manufacturers to reach customers from every corner across the globe.
- The government policies in support of the growth of the smartphone and subsidiary industries have a significant role in price slicing and the growth of mobile networks.

4.2. Powerful Smartphones. Over the last couple of years, the smartphone industry has seen an unprecedented focus on its hardware. Be it CPUs or GPUs or even DSPs, the processing capability of smartphones, to meet various purposes, has been increased exceptionally. The CPU and memory architectures are designed and tuned to boost heterogeneous computing. The GPUs are also engineered to enhance general purpose GPU (GPGPU) computing performance. Advancement on each module, though separately, makes the smartphones, as a whole unit, a great possibility to become a powerful computing platform. Let us have a run up on the latest developments on the smartphone hardware.

4.2.1. Powerful CPUs. The first multicore smartphone, announced at the end of 2010, was the LG Optimus 2X, loaded with the Tegra 2 processor from NVIDIA, with a maximum frequency up to 1.2 GHz [80, 57]. Since then all smartphone manufacturers have been in a war to load their products with an increasing number of cores. To buttress this exigence, SoC makers are coming out with more and more powerful smartphone processors quite regularly.

ARM Processors: ARM is the most popular processor architecture used in smartphones and supported by most of the major smartphone operating systems. More than 95% of smartphones employ ARM processors either as main application processors (Cortex processors) or other auxiliary processors such as ARM Embedded Processors, ARM Embedded Real-time Processors, ARM Secure Processors, etc. [6]. These processors are particularly preferred in smartphones due to the verity that they yield impressive performance in spite of consuming significantly less power. High scalability, power efficiency and superior computing performance (nearly 30 times better than the smartphones of that period) of ARMv7-A[4] based smartphones have resulted in an increase in smartphone sales colossally since its introduction in 2009 [6]. ARMs latest architecture, ARMv8-A[5], having state-of-the-art 64-bit instruction set with superior 64-bit programming model, is the pertinent extension of its predecessor. It supports the 64-bit general purpose and floating-point registers, each of 32 numbers, promising enhanced performance [6]. ARMs Cortex-A53[6] and Cortex-A57[7] are the first ARMv8-A based big.LITTLE[8] pair. As the big processor, Cortex-A57 produces an exceptional degree of performance by harvesting techniques like out-of-order execution, wide multi-issue capacities with large memories, whereas the Cortex-A53 focuses on power-efficient operations by means of a simpler pipeline in a smaller configuration while still delivering handsome performance. In the context of energy efficiency, it is worth mentioning of the Cortex-A35[9] which is designed for the next-generation smartwatches and probably has set a new standard in energy efficiency for mobile processors [36]. In 2015, the ARM came with Cortex-A72[10], based on 64-bit ARMv8-A, to thrust the performance up for the next generation of phones. The Cortex-A73[11], the smallest ARMv8-A based processor, has been 30% faster and 20% energy efficient than A72. A75[12] came with significantly improved integer and floating-point performance, and better memory workload management. The latest Cortex-A76[13] is designed to deliver laptop-class performance with mobile efficiency. In comparison to the A75, it offers 35% better performance with 40% more power efficiency. The development of ARM processors, in terms of competence in floating point performance, has made them a serious contender in consideration for a range of scientific applications. In fact, due to their energy efficiency, the ARM processors are going to be used in the futures supercomputers also.

4.2.2. Powerful GPUs. In this section, we shall check out some of the latest and most powerful GPUS for smartphones. **Imagination PowerVR:** The PowerVR[14] Series7XT[15] family drives GPU performance of mobile devices to new heights which can deliver up to 1.6 TFLOPS by employing multiple clusters (2 to 16). All the GPUs in this family are offered in two modes FP32 and FP16. The significant 8 clustered GT7800 has 256 FP32 ALU cores or 512 FP16 ALU cores. At the clock rate of 800 MHz, the FP32 cores deliver 820 GFLOPS while the FP16 cores deliver 410 GFLOPS. The highest rung in the series, the GT7900[16] has 16 clusters either of 512 FP32 cores that deliver 800 GFLOPS at a clock rate of 800 MHz (for 16nm FinFET+) or of 1024 FP16 cores, capable of delivering an imposing 1.6 TFLOPS at the same clock rate [77]. The company has provided an optional FP64 ALU for every pipeline that can be added to designs to deal with high-end supercomputing like applications.

Qualcomm Adreno: Snapdragon 810's Adreno 430[17] GPU has a whopping 288 numbers of shader cores. It has a clock rate of 600 630 MHz, and it delivers 388.8 408 GFLOPS. The Adreno 530[18] packed in Snapdragon 820 and 820A works in the range of 510 - 624 MHz offering 407.4 - 498.5 GFLOPS. It can achieve 588 GFLOPS working at 736 MHz.

NVIDIA Tegra: Tegra K1[19], has 192 Kepler cores running at 850 MHz and delivering 326.4 GFLOPS. The Tegra X1[20] features impressive 256 Maxwell cores. Though it has fewer shader cores than Adreno 430, it outsmarts the latter in ground performance. The clock speed of Tegra X1's GPU touches 1 GHz, and it delivers 512 GFLOPS.

[4]https://developer.arm.com/docs/ddi0406/latest
[5]https://developer.arm.com/docs/den0024/latest/armv8-a-architecture-and-processors/armv8-a
[6]https://developer.arm.com/products/processors/cortex-a/cortex-a53
[7]https://developer.arm.com/products/processors/cortex-a/cortex-a57
[8]https://developer.arm.com/technologies/big-little
[9]https://developer.arm.com/products/processors/cortex-a/cortex-a35
[10]https://developer.arm.com/products/processors/cortex-a/cortex-a72
[11]https://developer.arm.com/products/processors/cortex-a/cortex-a73
[12]https://developer.arm.com/products/processors/cortex-a/cortex-a75
[13]https://developer.arm.com/products/processors/cortex-a/cortex-a76
[14]https://www.imgtec.com/graphics-processors/
[15]https://www.imgtec.com/graphics-processors/powervr-series7xt/
[16]https://www.extremetech.com/gaming/199933-powervr-goes-4k-with-gt7900-for-game-\\consoles
[17]https://www.notebookcheck.net/Qualcomm-Adreno-430.146784.0.html
[18]https://www.notebookcheck.net/Qualcomm-Adreno-530.156189.0.html
[19]https://www.nvidia.com/object/tegra-k1-processor.html
[20]https://www.nvidia.com/object/tegra-x1-processor.html

ARM Mali: The Mali GPUs from ARM are not really well-known for their performance. Though they work at the higher clock rate, the GFLOPS output from them, comparatively, on the lower side. The Mali-T880 MP4 used in Helio X25 runs at 850 MHz but offers merely 115.6 GFLOPS while the same GPU used in Kirin 950/955 runs at 900 MHz and offers 122.4 GFLOPS [3]. The Mali-T880 MP12 used by Samsung Exynos 8890 clocks at 650 MHz offering 265.2 GFLOPS. The latest of the series, the Mali-G71[21] is 50% faster than the Mali-T880, attributed to 32 shader cores, twice the amount used on the Mali-T880 while at the same time it is 20% more power efficient [4, 37].

4.2.3. Powerful SoCs. In this section, we shall have a glimpse of some of the latest and most powerful smartphone processors and SoCs.

Qualcomm Snapdragon: Snapdragon[22] is a family of mobile SoC processor, architected and marketed by Qualcomm Technologies. It leapt a significant step in mobile computing by introducing the first 1 GHz mobile processor ever which featured in Toshiba TG01 in the year 2009 [42]. Since then there is no stopping from this SoC leader. Qualcomm Snapdragon 800 processor, launched in the year 2013, packed with four Krait 400 CPU cores and providing up to 2.45 GHz clock speed, outperformed all other processors in the mobile segment [72]. Next in the line, the octa-core Snapdragon 810 follows the big.LITTLE concept from the ARM and arranged in two clusters of different clock frequencies, providing an overall performance of 2.7 GHz [50]. The performance-oriented cluster of four 64-bit ARM Cortex-A57 runs at 1.958 GHz, and an energy-efficient cluster of four ARM Cortex-A53 runs at 1.555 GHz [72]. It also has one of the highly powerful GPUs, the Adreno 430. The Snapdragon 820, a 64-bit CPU that uses the ARMv8-A instruction set has four custom Kryo cores arranged as two dual-core setups - one for highly demanding tasks and the other for relatively low-profile tasks [54]. The performance-oriented cluster clocks at up to 2.15 GHz and the energy-saving cluster runs at 1.6 GHz. The Snapdragon 820 also packs Adreno 530 GPU, the new Hexagon 680 DSP and the 14-bit Qualcomm Spectra ISP [70]. The integration of these powerful chips is expected to provide high-performance mobile computing. To reinforce effective heterogeneous computing, the Adreno GPUs are devised to perform seamlessly with the Snapdragon CPUs, DSPs, and ISPs to accomplish processing-intensive GPGPU compute tasks. The new Snapdragon 821[23], loaded with the Qualcomm Kryo quad-core CPU, delivers 10% better performance than the 820, topping speeds up to 2.4 GHz. The Snapdragon 835[24] is 35% smaller and uses 25% less power than its predecessors. The Snapdragon 845[25] having an Octa-core CPU (Qualcomm Kryo 385) provides a clock speed of 2.8 GHz. It also incorporates the Qualcomm Adreno 630 GPU. The latest Snapdragon 855[26], built on a 7nm[27] technology from Taiwan Semiconductor (TSMC), have an eight-core (one of 2.84 GHz, three of 2.42 GHz, and four of 1.80 GHz) Qualcomm Kryo 485 CPU and an Adreno 640 GPU.

NVIDIA Tegra: NVIDIA has been the most prominent SoC makers in the market with their powerful Tegra series. The company has redefined mobile processing with their incredibly advanced SoC, the Tegra K1, launched in 2014. It encompasses 192 supercomputer-class GPU cores, based on the Kepler architecture that has accelerated some of the worlds most powerful supercomputers [15]. These prodigious powerful GPUs generate unbelievable extent of computing potential within an incredibly tiny package with remarkable power efficiency. As mentioned earlier, NVIDIA compares their latest product the Tegra X1 as yesteryears first teraflop supercomputer. As part of the CPU, it boasts eight 64-bit ARM cores, divided into two different clusters of four each to exploit ARM's HMP scheme. The first cluster is of ARM Cortex-A57 (for enhanced performance) and the second one of ARM Cortex-A53 (for energy efficiency) cores. At 16-bit floating point operations, the Tegra X1 nails a peak 1024 GFLOPS. Undoubtedly, the NVIDIA Tegra X1 is the most advanced SoC for any mobile device by far and considered well ahead of time.

Samsung Exynos: The Exynos series of SoCs is developed by Samsung, indigenously, for their sophisticated high-end smartphones. Samsung Exynos 7420[28], the octa-core SoC, features four Cortex-A57 cores at

[21]`https://developer.arm.com/products/graphics-and-multimedia/mali-gpus/mali-g71-gpu`
[22]`https://www.qualcomm.com/products/mobile-processors`
[23]`https://www.qualcomm.com/products/snapdragon-821-mobile-platform`
[24]`https://www.qualcomm.com/products/snapdragon-835-mobile-platform`
[25]`https://www.qualcomm.com/products/snapdragon-845-mobile-platform`
[26]`https://www.qualcomm.com/products/snapdragon-855-mobile-platform`
[27]`https://www.tsmc.com/english/dedicatedFoundry/technology/7nm.htm`
[28]`https://www.samsung.com/semiconductor/minisite/exynos/products/mobileprocessor/exynos-7-octa-7420/`

2.1 GHz and four Cortex-A53 cores at 1.5 GHz along with the ARM Mali-T760 MP8, at 772 MHz, as a 12 core GPU which delivers 210 GFLOPS [3]. The latest Exynos 8890[29], also an octa-core processor, has four power-efficient A53 running at 1.586 GHz and four Exynos M1 running at 2.29 GHz, the highest in the segment. If only 1-2 cores are used at a time, the Exynos M1 can peak up to 2.60 GHz [30], which is more than an Intel Core i5 chip (Intel i5 4302Y[30]) that has a clock speed of 2.3 GHz. As GPU, Exynos 8890 boasts the Mali-T880 MP12[31] which has a clock speed of 650 MHz and delivers astonishing 265.2 GFLOPS.

Apple A9x: Apple Inc. also did not want to be behind in the race as they came up with their indigenous SoCs, the Apple A series. They have showcased their sophisticated product Apple A9X[32], a 64-bit ARMv-8A based SoC, into their iPad Pro, which was released on November 11, 2015. The A9X features a 2.26 GHz dual-core CPU, called Twister. It boasts PVR 12 cluster Series7 GPU (GT7800 model has 8 cores while GT7900 has 16). The A9X also has the M9 motion coprocessor embedded in it. The Apple A10 is claimed to have better CPU and GPU performance compared to the Apple A9 by 40% and 50% respectively. The successor of A10, A11 Bionic[33] incorporates six cores; out of which two have 25% better performance than A10 while the performance of the rest is up to 70% better than A10. Additionally, the three-core GPU offers 30% faster graphics performance than the A10. In the A12 Bionic[34], out of six cores, the two performance cores are 15%, and four energy-efficient cores are 50% better than their counterparts in A11. The four-core GPU of A12 is up to 50% faster than A11s three-core GPU.

MediaTek's Helio: High performance in the low-budget is the hallmark of the SoCs from this Chinese company. It has produced the first-ever deca-core SoC, the Helio X20[35] which runs at 2.3 GHz, also featuring Mali-T880 MP4 GPU of 780 MHz that delivers 106 GFLOPS. Helio X25[36] has a Mali-T880 MP4[37] GPU of 8500 MHz, delivering 115.6 GFLOPS. The latest X30[38], also a ten-core chip, is made up of four Cortex-A72 cores at 2.5 GHz, two Cortex-A72 cores at 2.0 GHz, two Cortex-A53 cores at 1.5 GHz and two Cortex-A35 cores with 1.0 GHz.

Huawei HiSilicons Kirin: HiSilicons Kirin 950[39], the first smartphone SoC equipped with ARMs new Cortex A72 processor is an octa-core 64-bit chip. Following ARM's big.LITTLE architecture, it boasts four Cortex-A72, clocking at 2.3 GHz for high performance and four 1.8 GHz Cortex-A53 for power efficiency. The SoC also bundles a four-core Mali T880MP6 GPU, which is purposefully architectured to process large blocks of data in parallel. This prodigious GPU runs at 900 MHz, offering 122.4 GFlops. In Kirin 955[40], the clock rate of A72 processors is increased to 2.5 GHz, though the rest of the architecture in this SoC remains the same as its predecessor [21]. The latest in the league, Kirin 980[41], claimed as the first commercial SoC built on the 7nm[42] (TSMC) and also the first SoC to adopt Cortex-A76, has an eight-core CPU (two A76 of 2.6 GHz, two A76 of 1.92 GHz, and four A55 of 1.8 GHz). Kirin 980 offers a 75% better performance while being 58% more energy efficient than its predecessor Kirin 970. Its Mali G76 GPU also performs 76% better than Kirin 970. Moreover, Huawei claims that the performance of its innovative dual Neural Processing Unit (NPU) which is capable of processing 4500 pictures per minute, is almost two and three times greater than Snapdragon 845 and Apple A11 respectively.

4.2.4. Final Verdict. In the foreseeable future, more powerful processors with more cores are anticipated. With 7nm like technologies, more muscle can be put up in a single core which will enable shoving more computing capacity without compromising the chip size. The future smartphones loaded with these powerful SoCs will be,

[29]https://www.samsung.com/semiconductor/minisite/exynos/products/mobileprocessor/exynos-8-octa-8890/
[30]http://cpuboss.com/cpu/Intel-Core-i5-4302Y
[31]https://www.notebookcheck.net/ARM-Mali-T880-MP12-Benchmarks.160645.0.html
[32]https://www.anandtech.com/show/9766/the-apple-ipad-pro-review/2
[33]https://www.apple.com/newsroom/2017/09/iphone-8-and-iphone-8-plus-a-new-generation-of-iphone/
[34]https://www.apple.com/iphone-xs/a12-bionic/
[35]https://www.mediatek.com/products/smartphones/mt6797-helio-x20
[36]https://www.mediatek.com/products/smartphones/mt6797t-helio-x25
[37]https://www.notebookcheck.net/ARM-Mali-T880-MP4.159590.0.html
[38]https://www.mediatek.com/products/smartphones/mediatek-helio-x30
[39]http://www.hisilicon.com/en/Products/ProductList/Kirin
[40]https://www.notebookcheck.net/HiSilicon-Kirin-955-Octa-Core-SoC-Benchmarks-and-Specs.163438.0.html
[41]https://consumer.huawei.com/en/campaign/kirin980/
[42]https://www.tsmc.com/english/dedicatedFoundry/technology/7nm.htm

in the true sense, the little computing giants.

5. Challenges for Smartphone Crowd Computing. This section describes several challenges that are faced, in general, in using a smartphone as a computing device. Below, some of the rudimentary issues are mentioned.

Motivating the crowd: The challenge to motivate the crowd is always the primary concern of the vendors. As more people are motivated and participate in SCC, the greater will be the benefit. What one can do, one must do. Contributing to a social cause gives a person to achieve a feeling of self-actualization. One can perceive a self-satisfaction by doing what he/she can do within his/her capability. In this process, others may be benefitted. In other words, the implication of this realization of potential philanthropic results in the greater good.

Adopting a balanced and effective incentive policy: To attract people to actively participate in SCC, suitable and attractive incentive policies need to be formulated. The incentive policies should be according to the particular SCC application rather than generalised.

Security and privacy concerns: Mobile devices hold numerous personal information like financial data, email, location updates, social media information, and other user id and passwords. For using the mobile as a crowd computing device, it is utmost important that mobile data are secured from tracking, phishing or destroyed. The user must be given a sense of data security and privacy. It is this concern for which people are going to volunteer their devices for SCC. The SCC applications for the mobile devices should have a secured architecture which ensures confidence that the application is in no way sharing or have access to memory location and services of other apps.

Variable wireless network: With the majority of wireless services having a variety of different rate of transmission, both the average rate and rate of oscillation are vital metrics to measure the performance. It is quite challenging to determine the apt rate of oscillation a service can tolerate obtaining a high average rate. Here the service satisfaction for a smartphone user may be quantified with an increase in the concave utility function of the instantaneous transmission rate.

Designing middleware: As per consumer/market demand, smartphone companies are launching new products very regularly. These devices are widely heterogeneous in terms of hardware platform (CPU, GPU, SoC, no. of cores, signalling module, etc.), system and application software, manufacturer specifications, user interface, etc. This diverseness poses a great challenge in designing a generalised middleware and client application. The enhancement of interoperability among both the SCC server and the participant smartphones by incorporating the appropriate middleware that can handle smartphones irrespective of their hardware and software specification and support efficiently, to be developed and managed.

Maintaining QoS: Alike other distributed and networked computing systems, maintaining the quality of service (QoS) is challenging in SCC too. It is more challenging in SCC considering the factors such as device mobility, variable communication quality, diverseness in resources, etc. The criteria of satisfactory QoS have a vital contribution to the overall success of the SCC applications.

Heat: For smartphones, extreme hot or cold conditions are potentially damaging. The cell phone battery can be damaged due to excessive heat dissipated. If the CPU is busy for a long time, the phone will be heated. This may deter users from participating in SCC.

Limited memory: Smartphones have a limited internal memory which cannot be increased or reduced by the user. They have no additional expansion slots. The external memory option is not present in all the available smartphones. Therefore, it is vital to use the limited phone memory space intelligently so that it would not affect the other apps running in memory while importing jobs through SCC.

Battery power: One of the major obstacles in smartphone computing is the limited power of smartphone batteries. The current batteries are struggling to keep up with users active and ever-increasing smartphone demands. A whole lot of multiple features, faster processing, high resolution, etc. consume a huge amount of battery power. In addition to that, SCC will keep busy the smartphone processor which will drain the battery quickly. This heavily influences the participants willingness to lend their device. To expand the battery lifespan, the manufacturers are bringing batteries with huge sizes, causing inconvenience to the users. Fortunately, active research is going on towards solving the battery menace. In the following section, we shall explore some of the major initiatives.

6. Future Battery Technologies will Encourage Smartphone Crowd Computing: An Optimistic Study.

The most concerning facet regarding smartphone computing is the limitation and fast draining of battery power. While portable computers, smartphones, wearable computers, and other mobile devices are growing ever more advanced, both technologically and architecturally, they are still limited by power. The battery or energy technology has not moved at par in decades, which pulls back the pervasive & ubiquitous computing revolution. And it is only going to get worse as next-generation 4G networks come online, giving the phones access to high-speed, always-on connections and torrents of data. The battery is unarguably the biggest obstacle to smartphone computing. In a recent survey, nearly 70% of respondents stated that battery life is the single biggest limitation of their mobile phone and most of them are willing to pay more for the phones which offer extended power [27]. Present smartphone batteries are struggling to keep up with users active and ever-increasing smartphone usage demands. In spite of several limitations like slow charging, volatility, and degradation the current Lithium-ion batteries have served us well so far, since its first commercial release by Sony and Asahi Kasei 25 years ago. But this technology is approaching its limits after which no enhancement can be made out. According to researchers, potentially it can reach a maximum of 30% above current levels [23]. Though other classes of battery technologies, especially the lithium-sulphur (Li-S) and the lithium-air (Li-air) have potential to surpass the lithium-ion, they are not in the mainstream production and uses due to some fundamental challenges in these technologies.

6.1. Major Battery Drainers.

The main reasons for the majority of battery drainage are:

Screen: The major culprit in consuming power in a smartphone is its vibrant screen. The phones are coming with higher resolutions resulting in more power consumption. Lower-power display technology like OLED (Organic LED) is expected to take over LCD displays in the coming years, resulting in better battery life. And as people are more inclined to larger screens (most of the Indians prefer phablets); it indirectly paves to better battery performance. A larger screen would consume more energy, certainly, but it will allow fitting a larger battery also. If a phone of the same architecture has a 20% larger screen size, will run 40% longer period because of the bigger battery.

Power savvy processor: Akin to PCs, the processing capacity of smartphone processors also has increased consistently by Moors law. But it is not achieved, as it may seem, at the expense of additional battery power. Actually, improvement in processor technologies has rendered more processing power per watt. For example, as compared to mid-1980s processors which would consume 100-watt power, todays high-end processors can carry out more than 100,000 times MIPS, consuming the same amount of watt [56]. So, modern processors have become not only more powerful but more power-efficient as well. For example, thanks to a proprietary technology called Asynchronous Multiprocessing (ASMP), Snapdragon processors can dynamically adjust the voltage and speed of each CPU core independently that leads to decrease in wasted power and heat [6]. Another major factor which influences power efficiency, especially in mobile devices, is the SoC. SoCs offer smartphones a better power management capability by making able to put into sleep that particular power-consuming hardware which is not required at that time; thus, saving a considerable amount of battery power [56].

Signalling module: Similar to processor technology radio/wireless technologies are also improving significantly. Todays 4G network allows users to talk much longer than earlier older mobile networks by disbursing equal watt. Another basis of reduced battery drain is lesser transmitting distance. As the number of mobile phone users has grown enormously, the number of cellular network base stations has been greater than before. That means transmitting signals must travel a shorter distance from phone to base stations; which results in less power consumption [56]. Alongside mobile devices and mobile networks, the number of Wi-Fi routers also has escalated, and it should be noted that Wi-Fi connections consume significantly less energy compared to 3G mobile networks [78] and of course, even more, less than 4G. By eating up the same amount of power, Wi-Fi can transfer more than twice the volume of data as 3G [44]. This fact should go in favour of our view of that the smartphones who participate in volunteer computing should use the local wireless network for necessary message passing (no one would voluntarily offer their data pack balance for the purpose which least bothers him, at least immediately).

6.2. Other Causes.

The various reasons responsible for the battery drain are [11, 17]:

The GPS: The global positioning system (GPS) in a mobile device helps to identify the geographic location of the mobile user with the help of orbiting GPS satellites. The mobile device picks the GPS satellite signal to determine the mobile position on Earth. Since the GPS satellite signal is a week, the processor needs to perform a hard task to eliminate noise and decoding the signal; this makes the mobile device to consume energy faster. The energy consumption increases as mobile user travel and could deplete the battery in one or two hours.

Wi-Fi variations: A major source of battery power consumption comes from the Wi-Fi interface. The power consumption of the mobile device depends upon how signal strength it receives, weaker the signal more energy it consumes. The receiving Wi-Fi signal strength varies with the distance from Wi-Fi base station and also gets affected by obstructions like a wall, door, etc., in between the path of the signal.

The constant search for Wi-Fi networks: Constantly searching for a Wi-Fi station and connecting to the chosen router through the jungle of signals needs more power; this drains the battery significantly. The energy consumption increase as the person moves and the mobile device constantly searches for Wi-Fi base station for networking. Switching off the Wi-Fi interface while not in use would significantly increase the battery life.

LTEs high data rates: Huge battery is consumed when the phone is used for calling or e-mailing using the GSM, 3G, or LTE (4G). The smartphone hooks on to a single base station at a time, picking the one that offers the strongest signal and communicates even when the phone is also not being used. Mobile devices change the communicating base station for an incoming or outgoing call and for data communication while on the move. The handing over of communication services from one base station to another takes more energy. The network handover may also happen even in indoor locations. While moving from one room to another, there may be a loss of signal strength or network communication due to signal obstructions by walls or other things; this enforces the devices to search and connect to other base station available for communication causing more energy consumption. In another case, while the mobile device is not communicating directly, but apparently it is busy in the background for data communication. Many apps loaded in device get updated in the background by pulling and pushing data from the web. As this data updation happens in the background it does access data communication lines without the knowledge of the user. This consumes a lot of network data and thus enforces mobile devices to continually search and connect to various network connections and hence consumes a lot of energy. One way to restrict this power drain is to disable LTE, 3G, and Wi-Fi communication when not used and turning off automatic notifications of apps which are not used much.

The screen brightness is set too high: High screen brightness is a major source of battery consumption. The brightness can be managed easily by changing the screen brightness level as per the surroundings. For indoor where surrounding light is very less a low screen brightness is sufficient to view. Whereas at outdoors, where the surrounding luminosity is high, the screen brightness can be changed as per suitability. The other way of reducing the amount of battery power consumption is shortening the display time and turning off the display when not used.

Apps are staying busy in the background: One of the major reasons for profuse battery drain in a smartphone is the apps that run in the background. Mobile applications like memory booster, file cleaning, Facebook, email, etc., run in the background continually. This keeps draining the battery. The apps such as email, news and the apps for social media keep refreshing themselves in the background, discharging the battery automatically. Even the battery saver applications do not help; most of them actually consume more power than they save. While an application is installed, the background refresh change setting should be changed from automatic to manual. This will skip the automatic data update or feed every time the application starts and will improve the mobile battery life.

Using resource-intensive apps: One of the major sources of battery drain is continuously using resource-intensive applications for a long time. Gaming applications which are resource intensive in terms of processor cycle, display, sound, memory, and network keep continuously consuming energy. Frequent use of heavy apps such as weather apps and the Google Map which constantly locate users locations causes power leakage. The same happens in the case of overusing the apps which require GPS. Also, changing the application setting for refresh and updates frequently, does not help in battery conservation. To save battery life, the resource intensive application must be constraints for longer use.

Software updating: Updating the OS or application may add updates and fixes or bugs. Updating software causes connecting to the internet and continuously downloading data. This may contribute to battery usage.

The phone has a hardware issue: A phone suffering from hardware glitches is prone to quick battery drain.

Battery-draining activities: Phones that are over-engaged with battery-drainer activities are bound to drop battery power quickly. For example, the constant use of hotspot, Wi-Fi, and Bluetooth or continuous streaming of music and videos are the major reasons for faster batter loss. Also, making phone calls while travelling needs frequent hands-off, which requires energy. Other battery-losing activities are watching videos and playing graphics-heavy games for a long duration, using the phone camera frequently, etc.

Automatic syncing of the apps and data: The auto backup of the data (e.g., photos, videos, or documents) and synchronization of apps account for huge battery drain.

Notifications are going off all day: Constant lighting, vibration, or ringing notification sounds may be a reason why the battery dies quickly. The constant stream of notification sound is annoying; hence users often prefer to substitute it by vibration resulting in rapid battery drain.

Charging using a slower charger or a faulty cable: Charging a mobile device using a faulty or cheap cable or slower charger or a replica charger for a long time could harm the battery. This reduces the charge retention capability of the battery and could lead to excessive power draining. Therefore, to increase the battery life, it is required to charge the phone with the original charger or the charger which is shipped with the mobile device.

Killing recent apps or using task managers to free memory or kill apps: The recent Android version dynamically allocates memory to all apps running in the background. In these versions, the memory of apps is managed dynamically. For if the system requires memory, it automatically takes the memory allocated to apps running in the background without taking too much of the resources. Closing an app which is running in the background may close some of the system processes with the app. This causes the application to be restarted again from scratch next time when the application is started. This would cause unnecessary wastage of CPU resources and thus consumes battery. Hence, it is better to keep the application running in the background than to restart it again.

Using a ton of widgets, third-party customizations and health trackers: Using a huge number of widgets on an Android home screen may experience a drain of battery in smartphones. For example, widgets like step counters, calorie burn calculators, etc. are quite notorious as a battery burner.

Live wallpapers or bright wallpapers on AMOLED displays: Live wallpapers or bright wallpapers on AMOLED display look amazing, but they consume a great deal of battery power. Using sharp contrast and vivid coloured wallpapers on the AMOLED screen definitely juice more battery power. LCD, in contrast to AMOLED, consumes less battery power, no matter what wallpaper is displayed.

Unnecessary push notifications: The push notifications from different corners of the digital world are being hurled every day, which causes the phone to light up multiple times and make the transceiver antenna busy which consumes battery.

Replace the original old battery with a nonstandard one: Smartphones older by 2 or 3 years many often have issues with the battery. The battery either dies out or retains a charge for very less time. Replacing the old battery with a replica or nonstandard one is often cost-effective but may have serious problems. The nonstandard battery goes faulty very quickly and does not retain charge much longer; this needs frequent recharging, causing a lot of power consumption. Replacing the original battery with nonstandard ones is not an effective solution for the long run and may also cause harm to the mobile device also.

6.3. Advancement in Battery Technologies. It is untoward that the lack of advancement in the field of battery technologies has prevented smartphones being used as per their potential. But the good news is that we are on the verge of a power revolution. Research groups in universities and organizations are coming up with innovative ideas either to extend battery life or to minimize the recharging time and may be to accomplish both. Some are experimenting with alternate power sources like solar, body heat or air. In this section, we shall explore some of the promising new developments (present and future) which may encourage users to use their mobile devices more comprehensively without worrying about power limitation.

6.3.1. Extended Durability. Majority of the researches in battery technology are focusing on to maximise the duration between two charges, i.e., to minimise the charging frequency. The notable endeavours towards this direction are mentioned below:

- Intelligent Energy, a British energy technology company, aspires to revolutionize smartphones by replacing traditional lithium-ion batteries with hydrogen fuel cells. They are developing cells based on embedded fuel cell technology which will keep smartphones powered for more than a week after just a single charge [26].
- Rechargeable lithium metal batteries offer nearly 10 times of energy capacity than lithium-ion batteries. A Li-metal anode batterys capacity could be as high as 3,860 mAh/g, whereas a typical Li-ion battery with graphite on the anode can store around 380 mAh/g [41]. But these batteries could not be used in mobile devices so far due to safety hazards [29]. Scientists at Cornell University have developed a new safe way to use rechargeable lithium metal batteries at room temperature, an achievement which may lead to significantly extended power backup in mobile devices [69].
- The solid-state batteries found by MIT scientists along with Samsung offer more power and last longer. In these batteries, power density is improved by 20 to 30% which means more power to the device. The founders claim that the life of these batteries is thousands of cycles before degrading [24].
- Fuji Pigment has come out with their exciting aluminium-air Alfa batteries which have 40 times more capacity than lithium-ion and can be recharged by simply being topped up with water (with or without salt) [24]. According to the company, these batteries should last up to 14 days.
- Samsung also has claimed that they may have figured out a way to nearly double the battery life of smartphones by adding graphene on top of silicon anodes. When paired with a commercial lithium cobalt oxide cathode, the silicon carbide-free graphene coating allows the full cell to reach volumetric energy densities of 972 and 700 Wh at first and 200th cycle, respectively, 1.8 and 1.5 times higher than those of current commercial lithium-ion batteries [73].
- To tackle with the battery consumption by the smartphone screen, the main culprit as battery killer, a British start-up has come up with smart glass, a unique and innovative technology where the display will no longer rely on the battery to illuminate the screen, as it will use electrical pulses instead, allowing a device's battery life to last for a week [48]. Dr. Peiman Hosseini, founder of Bodle Technologies, enlightened that the smart glass technology, based on the technology that is used for rewritable DVDs, uses electrical pulses to create vivid, hi-tech displays that require no power and can be viewed clearly, even in direct sunlight [14].

6.3.2. Larger Capacity and Longer Battery Life. Larger battery capacity is one way to ensure the mobile devices run a long time without frequent recharging. Manufacturers are now going for batteries with a larger capacity which can hold huge power and thus less recharging. Companies are coming up with batteries with huge capacities. A couple of such examples is given below:

- Oukitel has introduced battery with a capacity of 10000mAh and which is increasingly on the standard battery capacity of 2000-3000mAh. The Oukitel battery of 10000mAH once charged can go 15 days without recharging.
- A group of physicists from the University of Missouri has developed a material with a unique structure (a honeycomb lattice) that are claimed to enable increasing the life of batteries by more than a hundred times [74].

Batteries with larger capacity are bigger in size and thus making the mobile device bigger in size. Companies are struggling to keep a balance between battery size and battery power. New battery technology is needed which seriously increases the battery power without compromising the battery size and thus the mobile design [7].

6.3.3. Faster Charging. Battery capacities are increasing, but the time it takes for charging makes the manufacturer and the users frustrating alike. Everybody hates waiting incessantly to charge ones mobile phone. Who has the time for this pesky task! Contentedly for us, we're starting to get smartphone batteries that not only last longer but also recharge awfully fast. Fortunately, the researchers are working and exploring a new way to charge batteries in faster ways. In the near future, we are going to have a breakthrough in charging technology.

Some of the efforts being explored by the researcher around the globe are:

- Quick Charge[43], a new charging technology, introduced by Qualcomm for its next generation of Snapdragon processors, has facilitated users to pull off a considerable amount of battery life with a short initial burst of charging. The introductory version (Quick Charge 1.0) could charge up to 40% faster than conventional charging while Quick Charge 2.0 augmented the charge time gain up to 75% [33]. The company claims the latest version, i.e., Quick Charge 3.0 is four times faster than conventional charging and 38% more efficient than Quick Charge 2.0. By their laboratory tests, it enables smartphones to attain from 0 to 80% charge in 35 minutes [22].
- Super VOOC Flash Charge[44], used in latest Oppo smartphones, thanks to the all-new 5V low-voltage pulse-charge algorithm, can charge an average smartphone battery (2500mAh) up to 100% in just 15 minutes [47].
- Lyte Systems has come up with a portable battery called Lumopack[45] that can be loaded in just 6 minutes with enough power to charge an iPhone 6 fully, thanks to its charge rate at 140W.
- A team at Stanford University has developed an ultrafast rechargeable high-performance, a long-lasting aluminium battery that can be fully charged in one minute and able to withstand more than 7,500 cycles without any loss of capacity in comparison of 1000 cycles of a typical lithium-ion battery [71].
- A team of scientists at MIT has created nanoparticles based ”yolk and shell” battery with a titanium dioxide ”shell” and aluminium ”yolk” (anode) [16]. They claim that this battery can be charged from zero to full in just six minutes. Moreover, according to them, these batteries can hold three times the capacity of the current lithium-ion batteries and also degrade slower, giving the longer life of the battery [58].
- Scientists at Rice University have made a breakthrough in micro-supercapacitors using which in batteries will allow them to charge 50 times faster than current batteries and discharge even slower than current supercapacitors [60].
- StoreDot[46], a start-up, born from the nanotechnology department at Tel Aviv University developed a biological semiconductor based superfast charger called StoreDot charger that can charge a mobile phone battery from empty to full in only thirty seconds.
- Harvard student Eesha Khare developed an award-winning tiny battery. The battery can store plenty of charge despite its small dimensions. The battery is a supercapacitor energy storage device made up of carbon fibre and metal oxide. The nanotechnology used in fabrication has made charging faster than previously possible. It not only charges faster but holds the charge for much longer time. Eesha is working further to reduce the supercapacitor battery charging time less than a minute [7].
- Kansas based engineer Shawn West claimed to develop a prototype rechargeable AA cell which recharges fully in 26 seconds, unlike other rechargeable battery which takes 34 hours for full charging. The battery is a lithium-ion capacitor with a voltage regulator and monitor circuitry attached to it. The circuitry ensures the correct voltage is being applied for charging and the right voltage is given away as output supply [7].

6.3.4. Wireless Chargers for Hassle-free Charging. In the near future, wireless chargers will replace traditional inconvenient wired chargers which require mobile devices are to be connected to a power socket or USB port. Wireless chargers can make us loose of this binding. The idea can be compared to how mobile devices access the Internet through Wi-Fi. The user can walk around the room with his phone or can work on it while it charges over the air, provided it should be in the range of the transmitter. Below some examples are cited of this kind of initiative:

- A start-up company uBeam[47] has discovered a wireless charging technology that allows a phone to be charged in the air from a transmitter, may be attached to the wall, using ultrasound for transmitting electricity. The uBeam technology[48] is based on the pioneering attempt of turning power into the sound

[43] `https://www.qualcomm.com/products/snapdragon/quick-charge`
[44] `http://www.oppo.com/en/technology/vooc`
[45] `lytesystems.com`
[46] `https://www.store-dot.com/`
[47] `https://ubeam.com/`
[48] `https://ubeam.com/technology/`

waves to be transmitted and then converted back to power on reaching the device.

- Ukrainian company XE has come up with a wireless charging prototype that can charge mobiles up to 16 feet away from the charging point [13]. One XE charger can charge four mobile devices simultaneously.
- Researchers from KAIST in South Korea have developed a new Dipole Coil Resonant System (DCRS) that can charge up to 40 smartphones from 5 meters away [8].
- The Cota project from Ossia Inc. enables multiple mobile devices to be charged wirelessly over a distance of 30 feet [7]. The charging can be done across walls, doors, and clothes.
- Many of todays high-end phones (e.g., Apple iPhone 8 and X series, Samsung Galaxy S7, S8, and S9 series, LG G7 Thinq and V30, Sony Xperia XZ series, Nokia 8 Sirocco and Lumia 735 and above, etc., to name a few) incorporate wireless charging facility. Most of them follow the Qi wireless charging standard which is capable of providing 5-15 watts of power to small personal electronic devices [20].

The technology is still in its early stage. Many issues are to be addressed. For instance, most of the energy is lost on the way from the charging base to the device. Also, the wirelessly transmitted power is very limited, unable to charge multiple devices continuously. Nevertheless, wireless chargers are getting popularity among users. Modern furniture, especially the work desks and coffee tables are incorporating wireless charging pads inbuilt. This will eventually allow users to get rid of mobile chargers.

6.3.5. Alternative Power Sources for Ease of Charging. Researchers have tried to explore different non-conventional sources of energy to charge batteries of mobile devices as described below:

Solar charged batteries: Batteries, rechargeable by solar rays are also being experimented by the smartphone makers like Alcatel [53] and Tag Heuer [12] where a transparent solar panel over the screen would let users charge their phone by simply placing it in the sun or even some artificial light. X-Play5 from Vivo, a Chinese mobile company, features the solar charging facility.

Batteries charged by human body heat: The idea of charging batteries from human body heat has been fascinated by the researchers for a long time. Recently a research team at Korea Advanced Institute of Science and Technology (KAIST) has developed a thin and light thermoelectric generator which collects body heat and converts it into energy, supplying the gadget (mainly wearables) it is attached to a never-ending stream of power [59].

Sound energy for charging phone: Nokia in collaboration with British engineers is creating a prototype phone which gets automatically recharged by background noise [7]. The phone will literally never go out of power. Further Nokia also has launched a first of its kind a wireless charging trouser. The trouser has an inductive charging plate which charges the Nokia phone on the move.

6.3.6. Other Innovative Way to Maximise the Battery Life. Other innovative approaches are being tried by researchers in preserving the battery as long as possible by developing technology for mobile devices which consumes less battery power. One such example is mentioned below [7]:

The innovative e-ink phone: Smartphone manufacturer Yota Devices has developed dual screen display for YotaPhone 2[49] mobiles to extend the mobiles battery life. An extra screen is added to the back side of the mobile device which works using e-ink technology causing very minimal battery usage. The e-ink screen like Amazons Kindle notebook is used for reading a text message, SMS, notifications, mail, news, etc.; thus, avoiding the LED screen usage and saving a huge amount of energy. For viewing video, web page, and managing a desktop application, the colour LED screen is used. However, it is found that separating the view into two displays would not much of the difference in overall battery life. A saving of two and a half days is gained in between charging. It might not be considered enough, but a good initiative to start.

6.4. Final Verdict. It is very unlikely that all the above-mentioned endeavours, towards better power backup for mobile devices, should go to the production. Many of them are not commercially viable. Some are not practicable. But considering all the research works going on, with different approaches, we are very much optimistic that the days are not far when the users will no further be haunted by the horror of low battery.

7. Conclusion. Smartphone crowd computing (SCC) has been seen as an ideal alternative to the existing high-performance computing (HPC) systems in attaining green and sustainable computing. There will be

[49]`http://yotaphone.com/gb-en/`

no additional productions because people will be using smartphones anyhow. Due to the smaller size, the environmental hazards and amount of e-waste will be substantially lesser. For the successful employment of SCC, several challenges need to be addressed. The most concerning facet regarding using smartphones is the limitation and fast draining of battery power. Battery technologies have not developed as per with the processor, memory, and storage technologies. But hopefully, some recent researches promise to solve the battery woos in smartphone uses. Various technologies such as long-lasting batteries, quick-charging batteries, wireless-charged batteries, batteries charged through unconventional means such as light or body temperature, etc. are on the verge of commercial launch. Some of them are already successfully introduced. Provided the challenges are resolved and with sufficient public awareness, SCC is expected to be as successful as Grid and Cloud computing as a cost and energy-efficient sustainable HPC paradigm.

REFERENCES

[1] Factory is where our computers eat up most energy. https://phys.org/news/2011-04-factory-energy.html, April 2011. Accessed 22 February 2019.

[2] The global footprint of mobiles. https://therestartproject.org/the-global-footprint-of-mobiles/, 2018. Accessed 24 February 2019.

[3] Gpu gflops. https://gflops.surge.sh/, March 2018. Accessed 27 February 2019.

[4] F. Alan. Arm introduces cortex a-73 chip and mali-g71 graphics chip; higher performance, energy efficient. http://www.phonearena.com/news/ARM-introduces-Cortex-A-73-chip-and-Mali-G71-graphics-chip-higher-performance-energy-efficient_id81620, May 2016. Accessed 8 June 2016.

[5] A. S. Andrae. Total consumer power consumption forecast. In *Nordic Digital Business Summit*, Helsinki, Finland, 2017.

[6] ARM. Arm and qualcomm: Enabling the next mobile computing revolution with highly integrated armv8-a based socs. Technical report, ARM/Qualcomm, 2014.

[7] C. Barraclough. How mobile phone battery life is going to seriously improve in 2016. https://recombu.com/mobile/article/smartphone-mobile-phone-battery-life-better-longer#, December 2015. Accessed 30 September 2016.

[8] C. B. Bautista. Wireless charging is coming: Researchers juice up 40 smartphones from 5 meters away. http://www.digitaltrends.com/mobile/farewell-cords-new-tech-can-wirelessly-charge-40-smartphones-5-meters-away/, April 2014. Accessed 25 June 2016.

[9] T. Bawden. Global warming: Data centres to consume three times as much energy in next decade, experts warn. https://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html, January 2016. Accessed 23 February 2019.

[10] Lovefone Blog. How much co2 does it take to make a smartphone? https://www.lovefone.co.uk/blogs/news/how-much-co2-does-it-take-to-make-a-smartphone, March 2018. Accessed 22 February 2019.

[11] J. Bolluyt. 9 reasons your phone battery dies so quickly. http://www.cheatsheet.com/gear-style/reasons-phone-battery-dies-quickly.html/?a=viewall, February 2017. Accessed 17 July 2017.

[12] A. Boxall. This luxury phone's screen has an amazing built-in solar charging panel. http://www.digitaltrends.com/mobile/tag-heuers-merediist-infinite-phone-has-a-solar-charging-screen/#!Elqeb, April 2014. Accessed 12 July 2016.

[13] J.-P. Buntinx. Long-distance wireless charging could boost mobile and bitcoin payments. https://news.bitcoin.com/long-distance-wireless-charging-boost-mobile-bitcoin-payments/, December 2015. Accessed 30 July 2016.

[14] R. Burn-Callander. Is this the end of constant smartphone recharging? http://www.telegraph.co.uk/finance/businessclub/technology/12008390/Is-this-the-end-of-constant-smartphone-recharging.html, November 2015. Accessed 12 July 2016.

[15] B. Caulfield. The story behind project 192: How a salinas, calif., crop circle became a worldwide puzzle. https://blogs.nvidia.com/blog/2014/01/05/salinas-crop-circle-and-project-192/, January 2014. Accessed 24 June 2016.

[16] D. L. Chandler. yolks and shells improve rechargeable batteries. http://news.mit.edu/2015/yolks-and-shells-improve-rechargeable-batteries-0805, August 2015. Accessed 21 February 2019.

[17] Deccan Chronicle. 8 reasons why your smartphone battery could die faster. http://www.deccanchronicle.com/technology/mobiles-and-tabs/260516/8-reasons-why-your-smartphone-battery-could-die-faster.html, May 2016. Accessed 17 July 2017.

[18] Cloudflare. How are serverless computing and platform-as-a-service different? — paas vs. serverless. https://www.cloudflare.com/learning/serverless/glossary/serverless-vs-paas/, 2019. Accessed 28 March 2019.

[19] C. Conger. Can my computer poison me? https://computer.howstuffworks.com/computer-poison1.htm., August 2009. Accessed 22 February 2019.

[20] Wireless Power Consortium. Qi - mobile computing. https://www.wirelesspowerconsortium.com/qi/, 2018. Accessed 17 February 2019.

[21] I. Cutress and A. Frumusanu. Huawei launches the p9 and p9 plus. http://www.anandtech.com/show/10231/huawei-launches-the-p9-and-p9-plus, April 2016. Accessed 7 June 2016.

[22] C. d. Looper. Qualcomm's quick charge 3 takes smartphones from 0 to 80 percent battery in 35 minutes. http://www.

techtimes.com/articles/84999/20150915/qualcomms-quick-charge-3-takes-smartphones-0-80-percent-battery.htm, September 2015. Accessed 11 March 2016.

[23] P. Daniel. 5 future battery technologies to make your phone last longer and charge faster. http://www.phonearena.com/news/5-future-battery-technologies-to-make-your-phone-last-longer-and-charge-faster_id67357, March 2015. Accessed 30 September 2016.

[24] L. Edwards. Future batteries, coming soon: charge in seconds, last months and power over the air. http://www.pocket-lint.com/news/130380-future-batteries-coming-soon-charge-in-seconds-last-months-and-power-over-the-air, December 2015. Accessed 4 March 2016.

[25] Energuide. How much power does a computer use? and how much co2 does that represent? https://www.energuide.be/en/questions-answers/how-much-power-does-a-computer-use-and-how-much-co2-does-that-represent/54/, 2019. Accessed 22 February 2019.

[26] Intelligent Energy. Intelligent energy launches a customer funded č5.25m project to deliver embedded week-long mobile phone power. http://www.intelligent-energy.com/news-and-events/company-news/2016/02/08/intelligent-energy-launches-a-customer-funded-525m-project-to-deliver-embedded-week-long-mobile-phone-power/, February 2016. Accessed 25 June 2016.

[27] Intelligent Energy. Survey: 70devices. http://www.intelligent-energy.com/news-and-events/company-news/2016/03/10/survey-70-of-consumers-drained-by-battery-limitations-of-mobile-devices/, March 2016. Accessed 25 June 2016.

[28] ERIdirect. How long does it take electronic waste to decompose? https://eridirect.com/blog/2015/11/how-long-does-it-take-electronic-waste-to-decompose/, November 2015. Accessed 22 February 2019.

[29] T. Fleischman. Room-temperature lithium metal battery closer to reality. http://news.cornell.edu/stories/2016/02/room-temperature-lithium-metal-battery-closer-reality, February 2016. Accessed 25 June 2016.

[30] A. Frumusanu. Early exynos 8890 impressions and full specifications. http://www.anandtech.com/show/10075/early-exynos-8890-impressions, February 2016. Accessed 14 September 2016.

[31] Fujitsu. Life cycle assessment and product carbon footprint. Technical report, 2010.

[32] E. Gelenbe and Y. Caseau. The impact of information technology on energy consumption and carbon emissions. *Ubiquity*, 2015(June):1–15, 2015.

[33] G. Gordon. Introducing quick charge 3.0: next-generation fast charging technology. https://www.qualcomm.com/news/snapdragon/2015/09/14/introducing-quick-charge-30-next-generation-fast-charging-technology, September 2015. Accessed 10 March 2016.

[34] GSMA. Smartphones to account for two thirds of world's mobile market by 2020 says new gsma intelligence study. https://www.gsma.com/newsroom/press-release/smartphones-account-two-thirds-worlds-mobile-market-2020/, September 2014. Accessed 17 August 2018.

[35] M. Halper. Supercomputing's super energy needs, and what to do about them. https://cacm.acm.org/news/192296-supercomputings-super-energy-needs-and-what-to-do-about-them/fulltext, September 2015. Accessed 22 February 2019.

[36] E. Herrmann. The fastest mobile processors compared. https://www.androidpit.de/die-schnellsten-handy-prozessoren, February 2016. Accessed 11 March 2016.

[37] J. Hruska. Arm announces new cortex-a73 cpu, mali g71 gpu. http://www.extremetech.com/computing/229328-arm-announces-new-cortex-a73-cpu-mali-g71-gpu, May 2016. Accessed 13 September 2016.

[38] IANS. Number of smartphone users crosses 300 million in india as shipments grew 18 percent. https://www.firstpost.com/tech/news-analysis/number-of-smartphone-users-crosses-300-million-in-india-as-shipments-grew-18-percent-3696457.html, January 2017. Accessed 17 August 2018.

[39] IEA. Co2 emissions from fuel combustion highlights. Technical report, 2013.

[40] Research Triangle Institute. Hazard assessment of the electronic component manufacturing industry. Technical report, U.S. Department of Health and Human Services, Ohio, 1985.

[41] D. Johnson. Nanomembrane may bring rechargeable lithium-metal batteries back. http://spectrum.ieee.org/nanoclast/semiconductors/materials/nanomembrane-may-bring-rechargeable-lithiummetal-batteries-back, February 2016. Accessed 25 June 2016.

[42] N. Jones. Battle of the 1ghz snapdragon super-phones. http://www.knowyourmobile.com/products/8324/battle-1ghz-snapdragon-super-phones, March 2010. Accessed 23 June 2016.

[43] N. Jones. How to stop data centres from gobbling up the worlds electricity. https://www.nature.com/articles/d41586-018-06610-y, September 2018. Accessed 24 February 2019.

[44] G. Kalic, I. Bojic, and M. Kusek. Energy consumption in android phones when using wireless communication technologies. In *35th International Convention MIPRO*, Opatija, Croatia, 2012.

[45] J. Koomey. Growth in data center electricity use 2005 to 2010. https://www.missioncriticalmagazine.com/ext/resources/MC/Home/Files/PDFs/Koomey_Data_Center.pdf, August 2011. [Accessed 1 September 2018].

[46] J. Kozlowicz. 8 ways data center environmental impact goes beyond emissions. https://www.greenhousedata.com/blog/data-center-environmental-impact-goes-beyond-emissions, November 2015. Accessed 18 August 2018.

[47] A. Kumar. Oppo wows with fast-charging super vooc, image stabilization tech. http://www.pcmag.com/article2/0,2817,2499860,00.asp, February 2016. Accessed 26 February 2016.

[48] V. Lanaria. Seven-day battery life on your smartphone? it could be coming soon. www.techtimes.com/articles/110852/20151126/seven-day-battery-life-on-your-smartphone-it-could-be-coming-soon.htm, November 2015. Accessed 12 July 2016.

[49] S. Liu. Analysis of electronic waste recycling in the united states and potential application in china. Technical report,

Columbia University, New York City, 2014.

[50] D. Luis. Tech war: Nvidia tegra x1 takes on snapdragon 810 with raw gpu power. http://www.phonearena.com/news/Tech-war-Nvidia-Tegra-X1-takes-on-Snapdragon-810-with-raw-GPU-power_id64748, January 2015. Accessed 04 March 2016.

[51] MCMC. Mobile e-waste: Old phone, new life. https://mobileewaste.mcmc.gov.my/en-my/about-mobile-e-waste, 2015. Accessed 25 February 2019.

[52] R. Mihindukulasuriya. Your mobile phone is a major contributor to toxic e-waste in the country. https://theprint.in/science/your-mobile-phone-is-a-major-contributor-to-toxic-e-waste-in-the-country/141430/, October 2018. Accessed 25 February 2019.

[53] S. Miles and I. Morris. Transparent solar panel display charges your phone through the screen. http://www.pocket-lint.com/news/126245-transparent-solar-panel-display-charges-your-phone-through-the-screen, January 2014. Accessed 12 July 2016.

[54] J. Mundy. Snapdragon 820 vs exynos 8890: Which galaxy s7 cpu is best? http://www.trustedreviews.com/opinions/snapdragon-vs-exynos, March 2016. Accessed 11 March 2016.

[55] S. Needhidasan, M. Samuel, and R. Chidambaram. Electronic waste an emerging threat to the environment of urban india. *Journal of Environmental Health Science & Engineering*, 12(36), 2014.

[56] Deloitte Network. Smartphone batteries : Better but no breakthrough. Technical report, Deloitte Touche Tohmatsu Limited, London, 2015.

[57] NVIDIA. The benefits of multiple cpu cores in mobile devices. Technical report, NVIDIA Corporation, 2010.

[58] J. Parsons. A battery that charges in 6 minutes? mit breakthrough could power smartphones of the future. http://www.mirror.co.uk/news/technology-science/science/battery-charges-6-minutes-mit-6231611, August 2015. Accessed 25 June 2016.

[59] Phys.org. Thermoelectric generator on glass fabric for wearable electronic devices. http://phys.org/news/2014-04-thermoelectric-glass-fabric-wearable-electronic.html, April 2014. Accessed 25 June 2016.

[60] Phys.org. Scientists see the light on microsupercapacitors: Laser-induced graphene makes simple, powerful energy storage possible. http://phys.org/news/2015-12-scientists-microsupercapacitors.html, December 2015. Accessed 25 June 2016.

[61] P. Prakash. Environmental impact of internet searches and data centers. https://www.linkedin.com/pulse/environmental-impact-internet-searches-data-centers-pranav-prakash/, July 2017. Accessed 18 August 2018.

[62] P. K. D. Pramanik and P. Choudhury. Iot data processing: The different archetypes and their security & privacy assessments. In S. K. Shandilya, S. A. Chun, S. Shandilya, and E. Weippl, editors, *Internet of Things (IoT) Security: Fundamentals, Techniques and Applications*, pages 37–54. River Publishers, 2018.

[63] P. K. D. Pramanik, P. Choudhury, and A. Saha. Economical supercomputing thru smartphone crowd computing: An assessment of opportunities, benefits, deterrents, and applications from indias perspective. In *4th International Conference on Advanced Computing and Communication Systems (ICACCS - 2017)*, Coimbatore, India, 2017. IEEE.

[64] P. K. D. Pramanik, B. Mukherjee, S. Pal, T. Pal, and S. P. Singh. Green smart building: Requisites, architecture, challenges, and use cases. In A. Solanki and A. Nayyar, editors, *Green Building Management and Smart Automation*. IGI Global, 2019.

[65] P. K. D. Pramanik, S. Pal, A. Brahmachari, and P. Choudhury. Processing iot data: From cloud to fog. its time to be down-to-earth. In P. Karthikeyan and M. Thangavel, editors, *Applications of Security, Mobile, Analytic and Cloud (SMAC) Technologies for Effective Information Processing and Management*, pages 124–148. IGI Global, 2018.

[66] P. K. D. Pramanik, S. Pal, and P. Choudhury. Beyond automation: The cognitive iot. artificial intelligence brings sense to the internet of things. In A. K. Sangaiah, A. Thangavelu, and V. M. Sundaram, editors, *Cognitive Computing for Big Data Systems Over IoT: Frameworks, Tools and Application*, pages 1–37. Springer, 2018.

[67] P. K. D. Pramanik, S. Pal, G. Pareek, S. Dutta, and P. Choudhury. Crowd computing: The computing revolution. In R. Lenart-Gansiniec, editor, *Crowdsourcing and Knowledge Management in Contemporary Business Environments*, pages 166–198. IGI Global, 2018.

[68] P. K. D. Pramanik, B. Upadhyaya, S. Pal, and T. Pal. Internet of things, smart sensors, and pervasive systems: Enabling the connected and pervasive health care. In N. Dey, A. Ashour, S. J. Fong, and C. Bhatt, editors, *Healthcare Data Analytics and Management*, pages 1–58. Academic Press, 2018.

[69] PTI. Superior batteries for laptops, smartphones in the offing. http://www.business-standard.com/article/pti-stories/superior-batteries-for-laptops-smartphones-in-the-offing-116020800581_1.html, February 2016. Accessed 25 June 2016.

[70] M. Shedd. Snapdragon 820 and kryo cpu: heterogeneous computing and the role of custom compute. https://www.qualcomm.com/news/snapdragon/2015/09/02/snapdragon-820-and-kryo-cpu-heterogeneous-computing-and-role-custom, September 2015. Accessed 11 March 2016.

[71] M. Shwartz. Aluminum battery from stanford offers safe alternative to conventional batteries. http://news.stanford.edu/2015/04/06/aluminum-ion-battery-033115/, April 2015. Accessed 25 June 2016.

[72] R. Smith and A. Frumusanu. The qualcomm snapdragon 820 performance preview: Meet kryo. http://www.anandtech.com/show/9837/snapdragon-820-preview, December 2015. Accessed 08 June 2016.

[73] In Hyuk Son, Jong Hwan Park, Soonchul Kwon, Seongyong Park, Mark H Rümmeli, Alicja Bachmatiuk, Hyun Jae Song, Junhwan Ku, Jang Wook Choi, Jae-man Choi, et al. Silicon carbide-free graphene growth on silicon for lithium-ion battery with high volumetric energy density. *Nature communications*, 6:7393, 2015.

[74] J. Sossamon. New device could increase battery life of electronic devices by more than a hundred-fold. https://munews.missouri.edu/news-releases/2018/0516-new-device-could-increase-battery-

life-of-electronic-devices-by-more-than-a-hundred-fold/, May 2018. Accessed 17 February 2019.

[75] S. Srivastava. Global smartphone shipments reached record 1.55 billion units in cy 2017. https://www.counterpointresearch.com/global-smartphone-shipments-reached-record-1-55-billion-units-cy-2017/, February 2018. Accessed 17 August 2018.

[76] Statista. Global smartphone shipments forecast from 2010 to 2022 (in million units). https://www.statista.com/statistics/263441/global-smartphone-shipments-forecast/, May 2018. Accessed 17 August 2018.

[77] A. Voica. Powervr gt7900: redefining performance efficiency in graphics. https://imgtec.com/blog/powervr-gt7900-redefining-performance-efficiency/, February 2015. Accessed 13 September 2016.

[78] Y. Xiao, R. S. Kalyanaraman, and A. Yla-Jaaski. Energy consumption of mobile youtube: Quantitative measurement and analysis. In *Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, Cardiff, UK, 2008.

[79] A. Zeenat. *A Study and Development of an Efficient E-waste Management System for Minimizing the Risks of Environmental Pollution.* PhD thesis, Savitribai Phule Pune University, Pune, 2016.

[80] C. Ziegler. Lg optimus 2x: first dual-core smartphone launches with android, 4-inch display, 1080p video recording. https://www.engadget.com/2010/12/15/lg-optimus-2x-first-dual-core-smartphone-launches-with-android/, December 2010. Accessed 23 June 2016.

Edited by: Anand Nayyar
Received: Mar 5, 2019
Accepted: Mar 25, 2019

DOI 10.12694/scpe.v20i2.1501
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 285–297. http://www.scpe.org © 2019 SCPE

THE SLOW HTTP DISTRIBUTED DENIAL OF SERVICE ATTACK DETECTION IN CLOUD

A. DHANAPAL*AND P. NITHYANANDAM†

Abstract. Cloud computing became popular due to nature as it provides the flexibility to add or remove the resources on-demand basis. This also reduces the cost of investments for the enterprises significantly. The adoption of cloud computing is very high for enterprises running their online applications. The availability of online services is critical for businesses like financial services, e-commerce applications, etc. Though cloud provides availability, still these applications are having potential threats of going down due to the slow HTTP Distributed Denial of Service (DDoS) attack in the cloud. The slow HTTP attacks intention is to consume all the available server resources and make it unavailable to the real users. The slow HTTP DDoS attack comes with different formats such as slow HTTP headers attacks, slow HTTP body attacks and slow HTTP read attacks. Detecting the slow HTTP DDoS attacks in the cloud is very crucial to safeguard online cloud applications. This is a very interesting and challenging topic in DDoS as it mimics the slow network. This paper proposed a novel method to detect slow HTTP DDoS attacks in the cloud. The solution is implemented using the OpenStack cloud platform. The experiments conducted exhibits the accurate results on detecting the attacks at the early stages. The slowHTTPTest open source tool is used in this experiment to originate slow HTTP DDoS attacks.

Key words: DDoS, slowloris, RUDY attacks, slow HTTP attack, OpenStack, Cloud Security, Layer 7 Attack

AMS subject classifications. 68M14, 68M12

1. Introduction.

1.1. Cloud computing and classifications. Cloud computing helps enterprises to minimize the initial investments and operational cost [1] on the data centres. The enterprises running online applications are moving to the cloud. National Institute of Standards and Technology (NIST) defines the cloud computing [2] is a model for enabling convenient on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications and services that can be rapidly provisioned and released with minimal management effort or service provider interaction. The benefit of making the services available anywhere and anytime, adding or removing the resource based on the demands, pay per usage model are the most important factors for promoting cloud across the enterprises.

Cloud computing broadly classified as shown in Fig. 1.1.

The Service Delivery Model is based on the type of cloud service provided. This model sub-divided [3] as Software as a Service, Platform as a Service and Infrastructure as a Service. The Deployment Model is based on cloud deployment. This is further classified as [4] Public cloud, Private cloud and Hybrid cloud.

Cloud computing threats: Cloud computing has many potential threats and generally categorized as shown in Fig. 1.2.

1.2. Distributed Denial of Service (DDoS) attacks . The DDoS attacks are designed to disturb the service provided and make it unavailable to the real end-users. The DDoS attacks divided into [5] [6] following types:

- Volumetric-based attacks: The network bandwidth is targeted to bring down the services. Example: UDP flooding attacks.
- Protocol-based attacks: The resources of the server is consumed to bring the service down. Example: Ping of death.
- Application Layer attacks: The goal of this attack is to make application un-available to real users. Example: slow HTTP DDoS attack.

This paper focused on one of the application layer specific HTTP DDoS attack known as the slow HTTP DDoS attacks. This work proposed a solution to effectively detect such slow HTTP DDoS attacks in cloud computing.

*School of Computing Science and Engineering, VIT University, Chennai, India (Dhanapal.a2013@vit.ac.in, dhanapal.ang@gmail.com).

†School of Computing Science and Engineering, VIT University, Chennai, India (Nithyanandam.P@vit.ac.in)

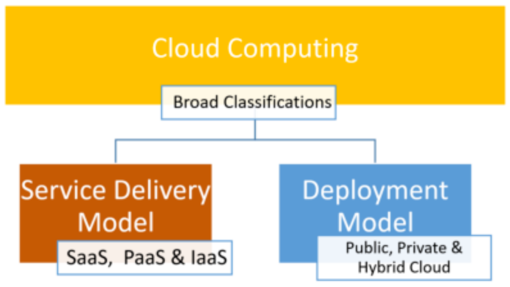


Fig. 1.1. *The cloud computing classifications*

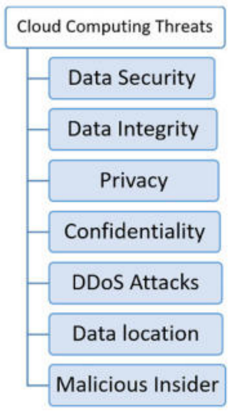


Fig. 1.2. *Cloud computing threats*

1.3. The HTTP DDoS Attacks and Types. The HTTP DDoS Attacks are application layer attacks in the system. These attacks aimed to make the online web services unavailable to the legitimate end users. The HTTP DDoS attacks type is visualized as in Fig. 1.3.

- HTTP flooding DDoS Attacks: The HTTP flooding attacks is application layer attacks and focused on flooding with an excessive request to the web server so that to overload and make the web server unable to process incoming requests. The service will be brought down eventually.
- Slow HTTP DDoS Attacks: This is another form of HTTP DDoS attacks that exploit the HTTP protocol behaviour and implementation of the application in a legitimate way. This attack consumes all the available resources in a slow manner. This slow attack aimed to make the web service unavailable to the real end-users.

2. The slow HTTP DDoS attacks . The slow HTTP attack to the web server can take places in three different formats. The types of slow HTTP attacks is given in Fig. 2.1.

The detailed explanation of each type is given below. following characteristics create the special needs and complexities associated with the HTTP flooding detection in the cloud environment.

- Slowloris Attacks (or) Slow HTTP Header Attacks [7]: The HTTP Get requests are usually send with valid HTTP Header. The web server processes the HTTP Get request upon receiving the complete header. The attackers make use of this HTTP protocol behaviour to carry out the DDoS attack on the web server. In this method, the attacker sends incomplete HTTP header to the web server and make the server to wait for the complete message. The attacker creates many such requests to the server until the server is unable to process any requests. The valid HTTP Header always ends with consecutive CR LF CR LF (ASCII Value 0d 0a 0d 0a) as shown in Fig. 2.2. During the attack, the HTTP header has only one CR LF (ASCII value 0d 0a) to indicate incomplete header to the web server as depicted in

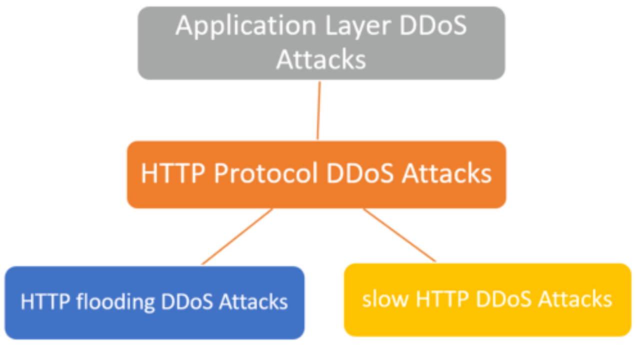

Application Layer DDoS Attacks
HTTP Protocol DDoS Attacks
HTTP flooding DDoS Attacks
slow HTTP DDoS Attacks

Fig. 1.3. *The HTTP DDoS attacks and types*

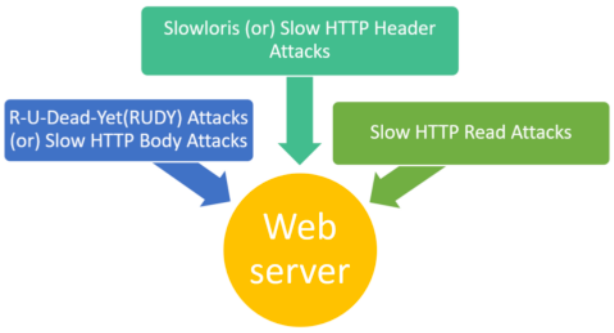
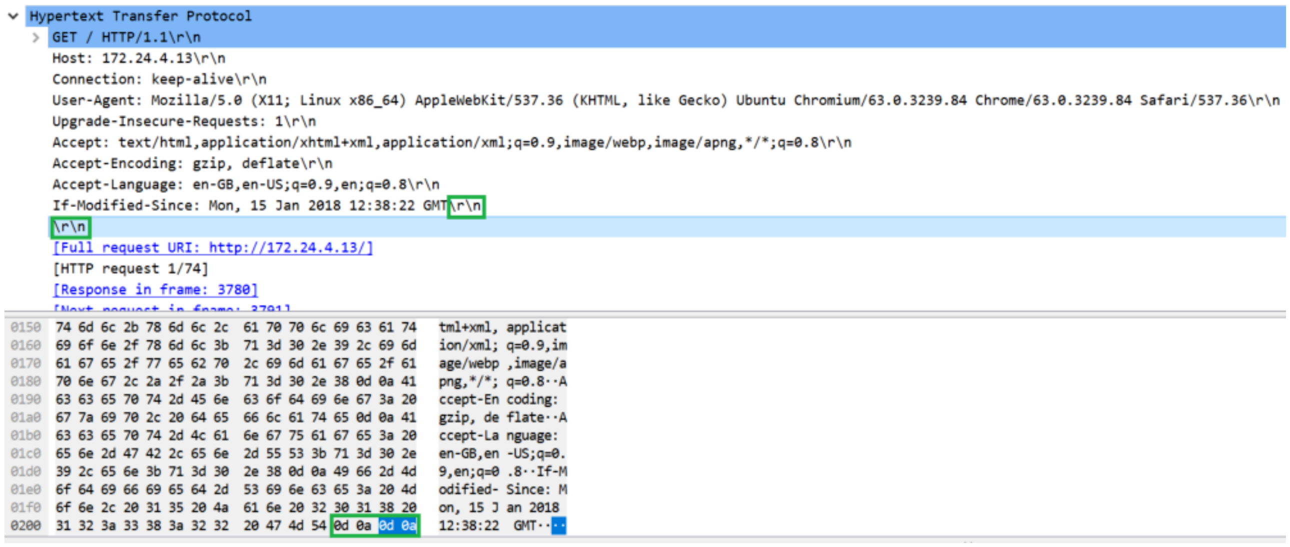

Slowloris (or) Slow HTTP Header Attacks
R-U-Dead-Yet(RUDY) Attacks (or) Slow HTTP Body Attacks
Slow HTTP Read Attacks
Web server

Fig. 2.1. *The slow HTTP DDoS attack classifications*

Fig. 2.3.

- RUDY Attacks (or) Slow HTTP Body Attacks [8]: RUDY or slow HTTP body attack is carried out using the HTTP Post requests. Unlike the slowloris attacks, this attack comes with a valid complete header in place. The length of the message has a bigger value. The attacker sends HTTP Post request with very small content size varies from one to few bytes with slow intervals of time. This makes the web server to wait for a long time to get the complete Post request. The attacker opens many connections with the server to consume all its resource so that the server will not be available to legitimate users. During the attack, the HTTP Post request contains an abnormal content-length field when compared to the normal request. The normal HTTP Post request and abnormal request are covered in Fig 2.4 and Fig. 2.5 respectively.
- Slow HTTP Read Attacks [9]: This is one another method of DDoS attack to the web server. In this model, the attacker sends the valid header and Get request to the web server, but it delays the read response from the server. The web client/browser at the attacker side frequently sends a message to the web server saying that it does not have enough space to receive the message so that to delay the server response. The TCP window size field is used during the 3-way handshaking method to exchange the window size between web client and server for agreeing upon the number of messages that they can handle. After establishing the connection web browser asks for more data, but it says not enough space to receive it. This makes the server to reserve the allocated resources for that transaction. The multiple transactions of this type being opened from attacker end to the victim web server to make web server unable to process any incoming requests. During the attack scenario, the web browser sends multiple TCP ZeroWindow messages to the web server as depicted in Fig. 2.6.

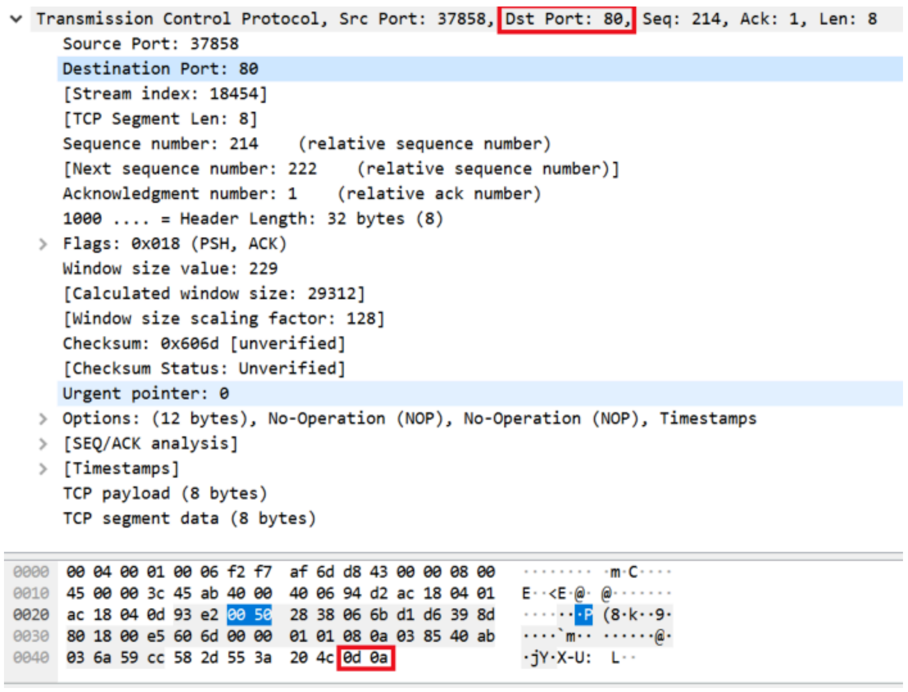
The rest of the paper is organized as follows: Section 3 discussed the related works. Section 4 explained



```
v Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: 172.24.4.13\r\n
    Connection: keep-alive\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/63.0.3239.84 Chrome/63.0.3239.84 Safari/537.36\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-GB,en-US;q=0.9,en;q=0.8\r\n
    If-Modified-Since: Mon, 15 Jan 2018 12:38:22 GMT\r\n
    \r\n
    [Full request URI: http://172.24.4.13/]
    [HTTP request 1/74]
    [Response in frame: 3780]
    [Next request in frame: 3791]

0150  74 6d 6c 2b 78 6d 6c 2c  61 70 70 6c 69 63 61 74   tml+xml, applicat
0160  69 6f 6e 2f 78 6d 6c 3b  71 3d 30 2e 39 2c 69 6d   ion/xml; q=0.9,im
0170  61 67 65 2f 77 65 62 70  2c 69 6d 61 67 65 2f 61   age/webp ,image/a
0180  70 6e 67 2c 2a 2f 2a 3b  71 3d 30 2e 38 0d 0a 41   png,*/*; q=0.8··A
0190  63 63 65 70 74 2d 45 6e  63 6f 64 69 6e 67 3a 20   ccept-En coding: 
01a0  67 7a 69 70 2c 20 64 65  66 6c 61 74 65 0d 0a 41   gzip, de flate··A
01b0  63 63 65 70 74 2d 4c 61  6e 67 75 61 67 65 3a 20   ccept-La nguage: 
01c0  65 6e 2d 47 42 2c 65 6e  2d 55 53 3b 71 3d 30 2e   en-GB,en -US;q=0.
01d0  39 2c 65 6e 3b 71 3d 30  2e 38 0d 0a 49 66 2d 4d   9,en;q=0 .8··If-M
01e0  6f 64 69 66 69 65 64 2d  53 69 6e 63 65 3a 20 4d   odified- Since: M
01f0  6f 6e 2c 20 31 35 20 4a  61 6e 20 32 30 31 38 20   on, 15 J an 2018 
0200  31 32 3a 33 38 3a 32 32  20 47 4d 54 0d 0a 0d 0a   12:38:22  GMT····
```

FIG. 2.2. *Normal HTTP Get Request*



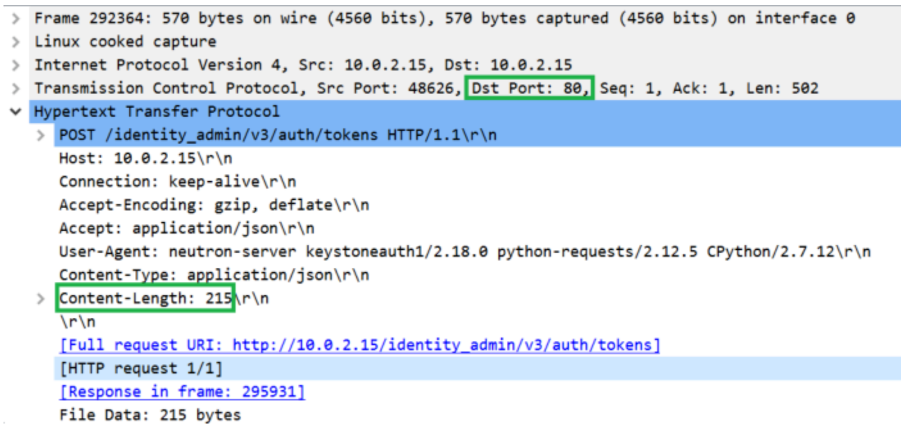
```
v Transmission Control Protocol, Src Port: 37858, Dst Port: 80, Seq: 214, Ack: 1, Len: 8
    Source Port: 37858
    Destination Port: 80
    [Stream index: 18454]
    [TCP Segment Len: 8]
    Sequence number: 214    (relative sequence number)
    [Next sequence number: 222    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x018 (PSH, ACK)
    Window size value: 229
    [Calculated window size: 29312]
    [Window size scaling factor: 128]
    Checksum: 0x606d [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (8 bytes)
    TCP segment data (8 bytes)

0000  00 04 00 01 00 06 f2 f7  af 6d d8 43 00 00 08 00   ········ ·m·C····
0010  45 00 00 3c 45 ab 40 00  40 06 94 d2 ac 18 04 01   E··<E·@· @·······
0020  ac 18 04 0d 93 e2 00 50  28 38 06 6b d1 d6 39 8d   ·······P (8·k··9·
0030  80 18 00 e5 60 6d 00 00  01 01 08 0a 03 85 40 ab   ····`m·· ······@·
0040  03 6a 59 cc 58 2d 55 3a  20 4c 0d 0a               ·jY·X-U:  L··
```

FIG. 2.3. *The HTTP header during slowloris attack*

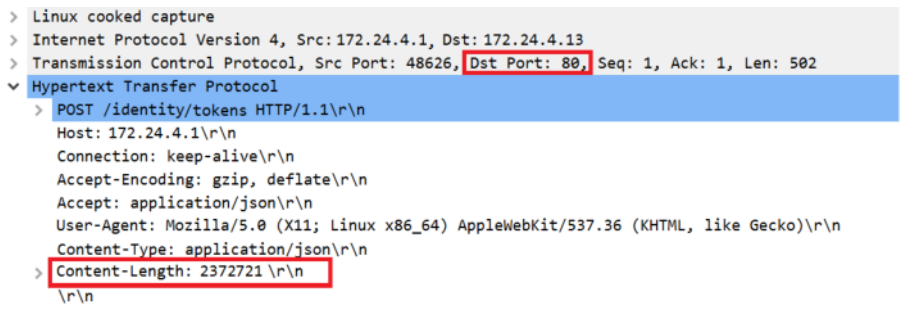
the details of the proposed solution architecture. Section 5 captured the details on experimentation details and the results. The conclusion and future direction of research are covered in Sec. 6.

3. Literature Review of the related works. The authors carried out a literature review of a large number of works related to DDoS detection and this section limits the number of papers very relevant to the



```
> Frame 292364: 570 bytes on wire (4560 bits), 570 bytes captured (4560 bits) on interface 0
> Linux cooked capture
> Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.15
> Transmission Control Protocol, Src Port: 48626, Dst Port: 80, Seq: 1, Ack: 1, Len: 502
v Hypertext Transfer Protocol
  > POST /identity_admin/v3/auth/tokens HTTP/1.1\r\n
    Host: 10.0.2.15\r\n
    Connection: keep-alive\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept: application/json\r\n
    User-Agent: neutron-server keystoneauth1/2.18.0 python-requests/2.12.5 CPython/2.7.12\r\n
    Content-Type: application/json\r\n
  > Content-Length: 215\r\n
    \r\n
    [Full request URI: http://10.0.2.15/identity_admin/v3/auth/tokens]
    [HTTP request 1/1]
    [Response in frame: 295931]
    File Data: 215 bytes
```

FIG. 2.4. *Normal HTTP POST Request*



```
> Linux cooked capture
> Internet Protocol Version 4, Src: 172.24.4.1, Dst: 172.24.4.13
> Transmission Control Protocol, Src Port: 48626, Dst Port: 80, Seq: 1, Ack: 1, Len: 502
v Hypertext Transfer Protocol
  > POST /identity/tokens HTTP/1.1\r\n
    Host: 172.24.4.1\r\n
    Connection: keep-alive\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept: application/json\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)\r\n
    Content-Type: application/json\r\n
  > Content-Length: 2372721 \r\n
    \r\n
```

FIG. 2.5. *The HTTP Post request during RUDY Attack*

topic of interest slow HTTP DDoS attack detection and mitigation.

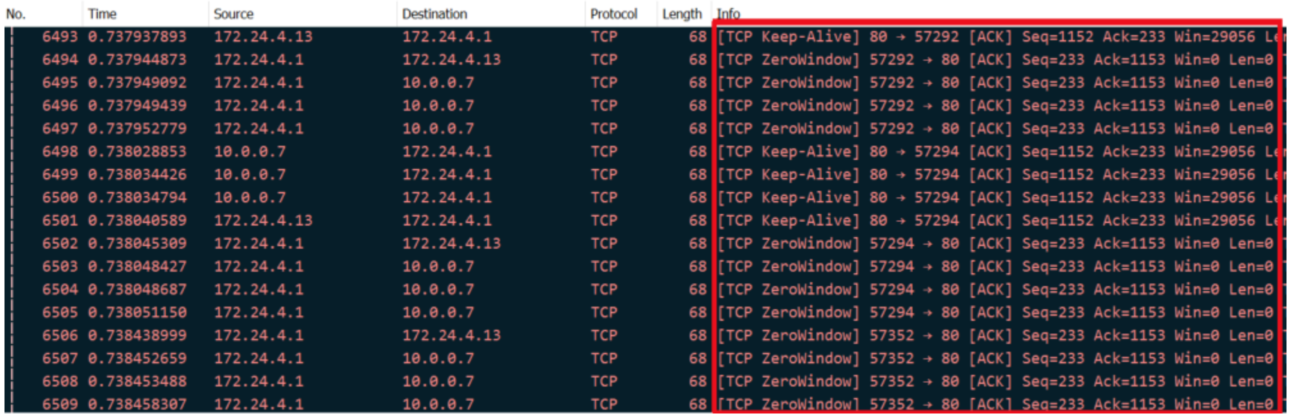
Tanishka Shorey et al. [10], This work studied and compared the DDoS attack tools such as slowloris, GoldenEye and Xerxes. The performance comparison is done based on the three parameters time to launch attacks successfully, Rate of Traffic and Size of packet. This gives insight into the potential tools used for HTTP DDoS attacks.

Clifford Kemp et al. [11], proposed a solution to detect slow HTTP Read DDoS attack detection using the machine learning techniques. The slowHTTPTest tool is used for generating the attack. This proposed technique works for the standalone environment. It has a scope for improvement to the cloud environment. Also, the solution has to be extended to address other slow HTTP DDoS attack types such as RUDY and slow HTTP Read attacks.

Kiwon Hong et al. [12], used the software-defined network method to implement their solution against the slow HTTP DDoS attacks. The traffic flow between the web server and the web browser via the switch is used by the defence application for analysis. This work classified the request as attacks, real slow connections and legitimate requests. This is a general solution proposed and need to be enhanced for cloud computing.

Shunsuke Tayama et al. [13], worked on the slow HTTP Read attack. The important parameter considered to detect the attack is the bandwidth rate. This work concluded that the browser having the connection limit equal to the processing capability of the web server and bandwidth greater than 500Kbps is good enough to carry out successful slow HTTP Read attacks. This work has gaps in analyzing the other types of slow HTTP DDoS attacks.

Aanshi Bhardwaj et al. [14], used the OpenStack cloud to analyze and evaluate the DDoS attacks. This work carried out to study DDoS attacks in the cloud using multiple DDoS attack tools. This paper requires an enhancement of providing the solution to detect the slow HTTP DDoS attacks in the cloud environment.



```
No.   Time         Source        Destination   Protocol Length Info
 6493 0.737937893  172.24.4.13   172.24.4.1    TCP      68 [TCP Keep-Alive] 80 → 57292 [ACK] Seq=1152 Ack=233 Win=29056 Le
 6494 0.737944873  172.24.4.1    172.24.4.13   TCP      68 [TCP ZeroWindow] 57292 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6495 0.737949092  172.24.4.1    10.0.0.7      TCP      68 [TCP ZeroWindow] 57292 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6496 0.737949439  172.24.4.1    10.0.0.7      TCP      68 [TCP ZeroWindow] 57292 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6497 0.737952779  172.24.4.1    10.0.0.7      TCP      68 [TCP ZeroWindow] 57292 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6498 0.738028853  10.0.0.7      172.24.4.1    TCP      68 [TCP Keep-Alive] 80 → 57294 [ACK] Seq=1152 Ack=233 Win=29056 Le
 6499 0.738034426  10.0.0.7      172.24.4.1    TCP      68 [TCP Keep-Alive] 80 → 57294 [ACK] Seq=1152 Ack=233 Win=29056 Le
 6500 0.738034794  10.0.0.7      172.24.4.1    TCP      68 [TCP Keep-Alive] 80 → 57294 [ACK] Seq=1152 Ack=233 Win=29056 Le
 6501 0.738040589  172.24.4.13   172.24.4.1    TCP      68 [TCP Keep-Alive] 80 → 57294 [ACK] Seq=1152 Ack=233 Win=29056 Le
 6502 0.738045309  172.24.4.1    172.24.4.13   TCP      68 [TCP ZeroWindow] 57294 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6503 0.738048427  172.24.4.1    10.0.0.7      TCP      68 [TCP ZeroWindow] 57294 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6504 0.738048687  172.24.4.1    10.0.0.7      TCP      68 [TCP ZeroWindow] 57294 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6505 0.738051150  172.24.4.1    10.0.0.7      TCP      68 [TCP ZeroWindow] 57294 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6506 0.738438999  172.24.4.1    172.24.4.13   TCP      68 [TCP ZeroWindow] 57352 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6507 0.738452659  172.24.4.1    10.0.0.7      TCP      68 [TCP ZeroWindow] 57352 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6508 0.738453488  172.24.4.1    10.0.0.7      TCP      68 [TCP ZeroWindow] 57352 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
 6509 0.738458307  172.24.4.1    10.0.0.7      TCP      68 [TCP ZeroWindow] 57352 → 80 [ACK] Seq=233 Ack=1153 Win=0 Len=0
```

FIG. 2.6. *The slow HTTP Read attack*

A. Dhanapal et al. [15], proposed a cloud testbed model using the OpenStack Infrastructure as a Service (IaaS) model. The different possible DDoS attack scenarios are explained in the work. This paper discussed only the framework details of the cloud environment.

Ghafar A. Jaafar et al. [16], reviewed the various work done to detect HTTP DDoS attacks in the time frame Jan 2014 to Dec 2018 and summarised the different parameters used in the solutions, attack detection levels, evalution methods, dataset used and performance matrix. This is a review paper helps to understand currently available solutions.

Trung V. Phan et al. [17], offered a hybrid machine learning model to improve the classifications and history-base filtering method combined togather to detect the DDoS attacks in SDN-based cloud environment. This is mainly used for the cloud environment build based on the SDN.

The proposed solution enhanced the OpenStack framework discussed in [15] and implemented the solution to detect the different type of slow HTTP DDoS attacks in the cloud environment. The slowHTTPTest tool is used to generate the variants of slow HTTP DDoS attack to the NGINX web server running in the OpenStack cloud.

The literature review of the numerous work studied and discussed shows that still we have the gaps to define the effective solution to detect the slow HTTP DDoS attack in the cloud.

4. The Proposed Solution. The proposed solution helps to detect the slow HTTP DDoS attack in the cloud environment and notifies to the administrator in case of an attack. The system architecture of the proposed defence model is depicted in Fig. 4.1.

The offered solution captures the multi-tenancy of cloud computing. The multiple tenants are located across the cloud service provider environment. Each tenant runs their own virtual machines (VM). The cloud has the following components

- Compute Controller: The cloud controller component contains the web interface to access and control the cloud environment, metering services used for calculating pay per usages, cloud monitoring services to look for the health of the systems, and most importantly the cloud scheduler to allocate the resources to the various tasks. This component is controlled and managed by the cloud service provider.
- Networking Node: This part provides networking services across the different components of the cloud environment. The important work of the networking node is to provide internal and external IP services and connectivity to the multiple tenants of the cloud services.
- Cloud Compute Cluster: The compute cluster is the place where each of the cloud users work is carried out. This component runs the various virtual machines of the tenants. Multiple tenants are kept in parallel in the cloud compute clusters. The cloud compute cluster has an entry point to access the virtual machines of the tenants. The proposed DDoS detection solution is implemented in this cloud

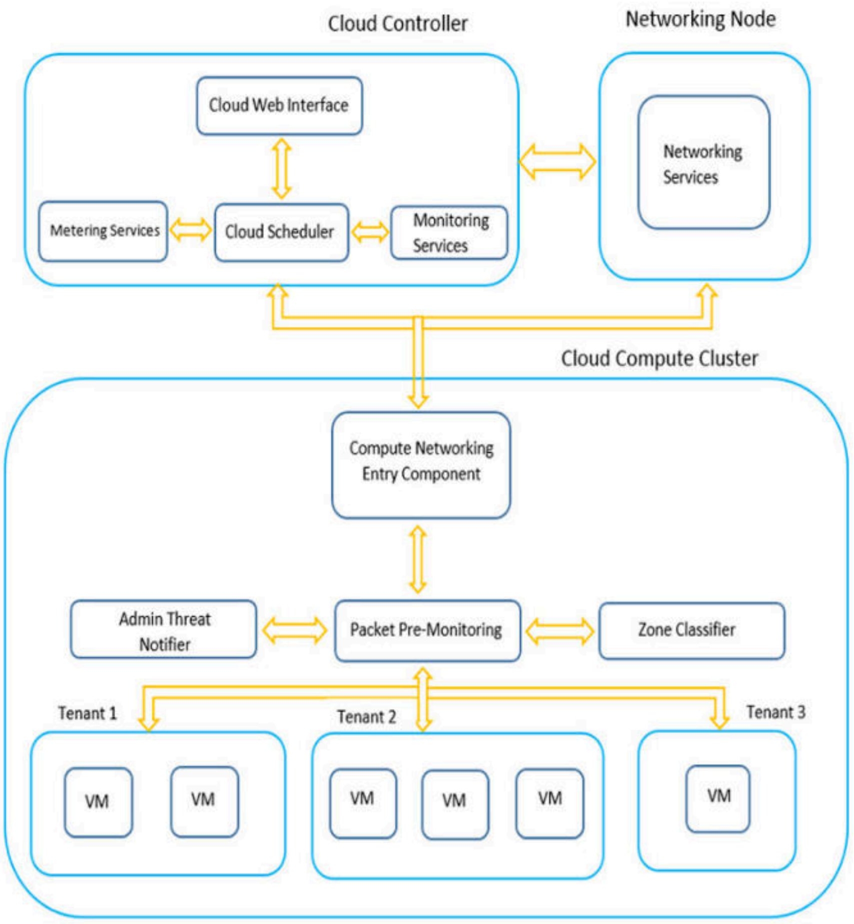


FIG. 4.1. *Proposed Solution System Architecture*

compute cluster. Every HTTP requests entering the cloud environment is monitored using packet pre-monitor module to identify the behaviour of the client and the same information is passed to the zone classifier module. The zone classifier module has two stages known as ALLOWED LIST zone and BLOCKED LIST zone. The clients are classified into either of the zones in any given point in time based on their behaviour.

Initially, all the clients requesting to the web server are kept in the ALLOWED LIST zone, and requests are forwarded to the web server for further processing. The behaviour of the client is continuously monitored by the classifier module. The client exhibits the suspecting activity are moved into BLOCKED LIST zone.

The client with the following behaviour is put into BLOCKED LIST zone.

- The average network delay to reach the client is calculated by means of sending 5 ping requests to the client. The average reply response time of those ping request is calculated and considered as network delay. This helps to identify the client mimicking the slow network.
- The client will be exhibiting the slow network. Whenever the classifier module gets the slow request continuously, five times the average network delay is compared with the client request interval. If the time between each of the HTTP request of the client exceeds more than five times of the calculated network delay, the client is treated as a bot machine and moved into BLOCKED LIST zone. The five times of the network delay selected by considering the processing required for the application.
- The client sends the HTTP Post request with the abnormal size of data more than three thousand of

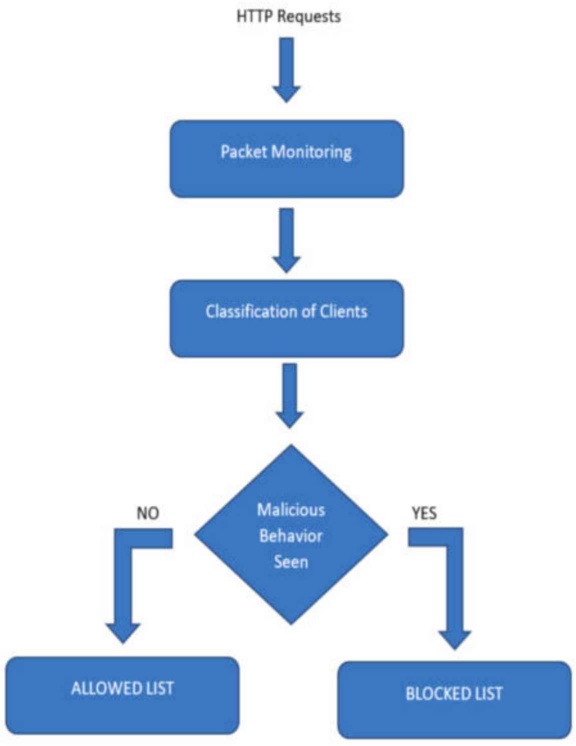


FIG. 4.2. *Flowchart of the proposed architecture*

the bytes and one or few bytes in the request with slow network behaviour is closely monitored. The same above calculated network delay is compared with the request interval time of the client. In case the request interval exceeds more than 5 times of average network delay, it is considered as a bot and moved into BLOCKED LIST zone.

- The client with a frequent request to the server saying that zero TCP Window size is monitored and moved into BLOCKED LIST if this behaviour is seen persistently.
- The client sends out an HTTP Post request with one or few bytes or Get request with an incomplete header in the time interval very close (above 80%) to the connection keep-alive time is treated as an attack and moved to BLOCKER LIST zone. For example, if the connection keep-alive time is 10 seconds and the client sends a request in the interval only after 8 seconds are suspected.
  Whenever the classifier detects the abnormal activity of the client, it moves the client into BLOCKED LIST zone, and the appropriate information is sent to the admin threat notifier module. The admin threat notifier is responsible for indicating the cloud administrator about the client's suspicious activity and corresponding classifier zone movement. This helps the cloud administrator to take necessary action against the slow HTTP DDoS attacks.

The flowchart of the proposed architecture is captured in figure Fig. 4.2.

4.1. Implementation Details. The solution discussed is implemented using the OpenStack IaaS cloud software. The OpenStack is freely available on the internet for anyone. The OpenStack Network Topology look as shown in Fig. 4.3.

The multiple customers are placed on the OpenStack cloud and each of the customers is associated with separate IP networks. The OpenStack network topology consists of three private networks namely Orange-Private-Network, Green-Private-Network, Private-Network and one Public-Network. Each private network associated with one customer. Every virtual instance or nodes of the customer is assigned with an internal IP address in the given address range. The OpenStack cloud provides two IP addresses to each virtual node. The IP address

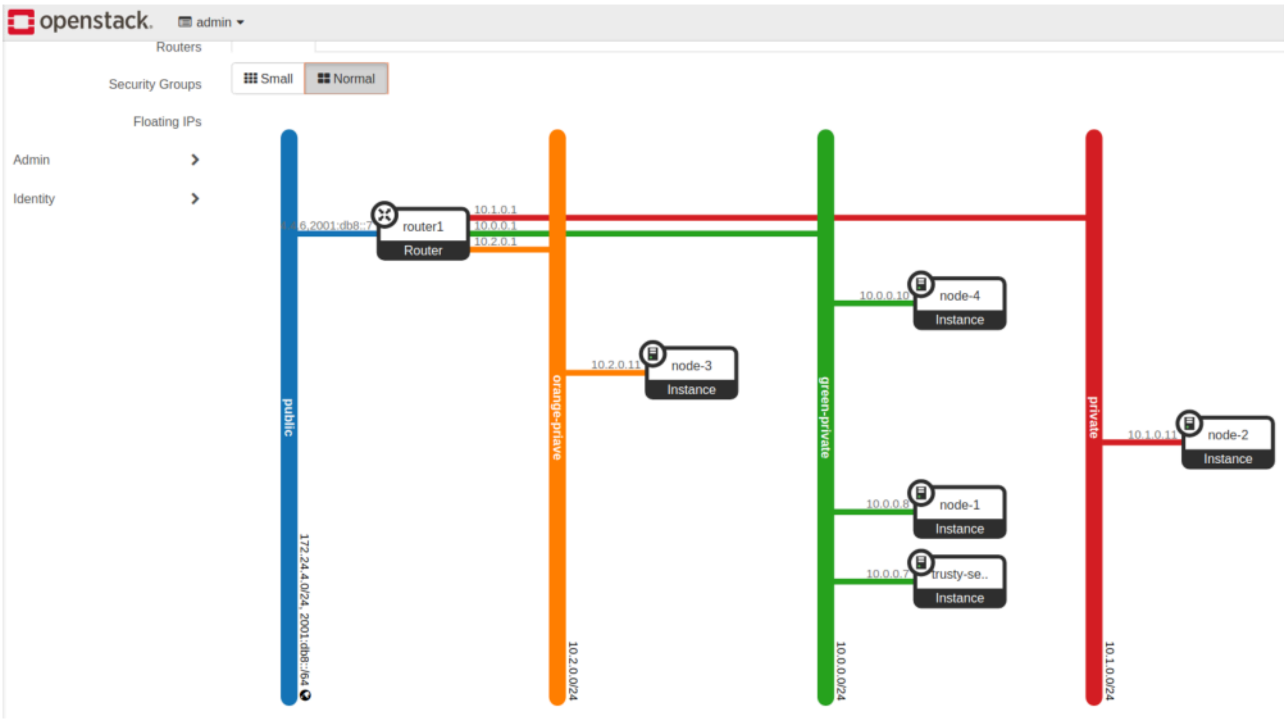


Fig. 4.3. *The OpenStack network topology*

Table 4.1
The OpenStack Virtual Instance Details

Instance Name	Internal/Private IP address	External/Public IP address
Trusty-server	10.0.0.7	172.24.4.13
Node-1	10.0.0.8	172.24.4.4
Node-2	10.1.0.11	172.24.4.11
Node-3	10.2.0.11	172.24.4.14
Node-4	10.0.0.10	172.24.4.12

starting with 10.x.x.x are known as private IP address and used for communication across the virtual nodes within the cloud environment. Another IP address is known as public or floating IP address. This floating IP address starts with 172.24.x.x and used for communicating with the outside world through the internet. The Public-Network has an IP address in this range to communicate to the external world.

The customer-1 on OpenStack runs only one virtual node known as node-3, and the customer is placed on the Orange-Private-Network with IP address range of 10.2.0.0/24.

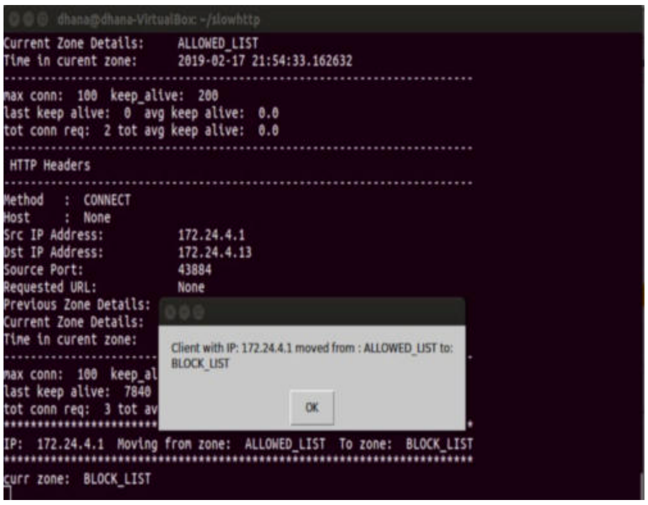
The customer-2 is placed on the Green-Private-Network with IP address range of 10.0.0.0/24. This customer runs three virtual nodes namely node-1, node-4 and trusty-server. The trusty-server node runs the NGINX web server in the OpenStack cloud.

Similarly, the customer-3 placed on the Private-Network has IP network 10.1.0.0/24. The node-2 is virtual instance launched for customer-3.

Each of the virtual nodes internal, as well as external IP address details are captured in Table 4.1.

The NGINX web server is running over the trusty-server of the customer-2. The web pages are accessed using the URL http://172.24.4.13/WebPage.html. The opensource tool named slowHTTPTest is used to generate three different types of slow HTTP DDoS attacks against the NGINX web server running in the cloud.

5. Experimentations Results and Discussions. Slow HTTP DDoS test carried out using slowHTTP-Test tool with following options:



```
dhana@dhana-VirtualBox: ~/slowhttp
Current Zone Details:     ALLOWED_LIST
Time in curent zone:      2019-02-17 21:54:33.162632
----------------------------------------------------------------------
max conn:  100  keep_alive:  200
last keep alive:  0  avg keep alive:  0.0
tot conn req:  2 tot avg keep alive:  0.0
----------------------------------------------------------------------
 HTTP Headers
----------------------------------------------------------------------
Method   :  CONNECT
Host     :  None
Src IP Address:           172.24.4.1
Dst IP Address:           172.24.4.13
Source Port:              43884
Requested URL:            None
Previous Zone Details:
Current Zone Details:
Time in curent zone:
-----------------------
max conn:  100  keep_al
last keep alive:  7840
tot conn req:  3 tot av
***********************
IP:  172.24.4.1  Moving from zone:  ALLOWED_LIST  To zone:  BLOCK_LIST
**********************************************************************
curr zone:  BLOCK_LIST
```

```
Client with IP: 172.24.4.1 moved from : ALLOWED_LIST to:
BLOCK_LIST
```

Fig. 5.1. *The detection of slow HTTP DDoS attack*

c is number of connections.
H indicates the slowloris Header attack mode, -B for slow HTTP Body attack mode, -X slow HTTP read attack mode.
g says generate statistics and output with file -o file name.
i time interval between requests.
r number of connections per seconds.
t type of request (GET/POST)
u target URL
l test length in seconds.

The slow HTTP Header DDoS attacks, as well as slow HTTP Read attacks are launched against the web server URL http://172.24.4.13/products.html

The command used for slow HTTP Header attack:

```
slowhttptest -c 10000 -H -g -o slowhttpheader -i 10 -r 500 -t GET -u http://172.24.4.13/products.html -l 600
```

The slow HTTP read attack done using:

```
slowhttptest -c 10000 -X -g -o slowhttpread -i 10 -r 500 -t GET -u http://172.24.4.13/products.html -l 600
```

The slow HTTP Body attacks are carried out for the URL http://172.24.4.13/register.html

The command used for slow Body attack is:

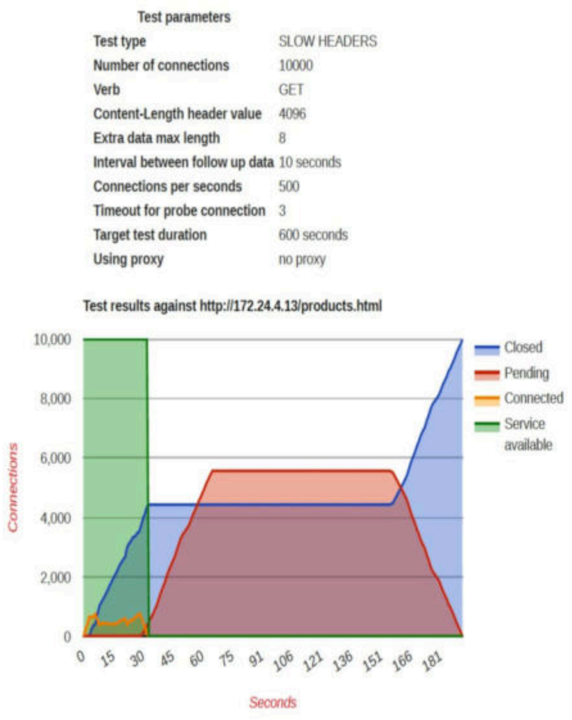
```
slowhttptest -c 10000 -B -g -o slowhttpbody -i 10 -r 50 -t POST -u http://172.24.4.13/register.html -l 600
```

The proposed DDoS detection model efficiently identify all possible slow HTTP DDoS attacks and classify the attacking client to BLOCKED LIST zone. Fig. 5.1 shows that notification of the slow HTTP DDoS attack launched from client 172.24.4.1 is detected and moved to BLOCKED LIST zone from ALLOWED LIST zone.

The common parameters used for generating different slow HTTP DDoS attacks by the slowHTTPTest tool are:
The total number of connections: 10000;
The content-length of the header: 4096 bytes;
Timeout for probe connection: 5 seconds.

The performance results of the slow HTTP Header attack captured in Fig. 5.2.

The service available line Fig. 5.2 indicates the availability of the web services to the slowHTTPTest tool. The implemented system detected the attack when the connection request crossed above 4000, and the client is



Test parameters

Test type	SLOW HEADERS
Number of connections	10000
Verb	GET
Content-Length header value	4096
Extra data max length	8
Interval between follow up data	10 seconds
Connections per seconds	500
Timeout for probe connection	3
Target test duration	600 seconds
Using proxy	no proxy

Test results against http://172.24.4.13/products.html

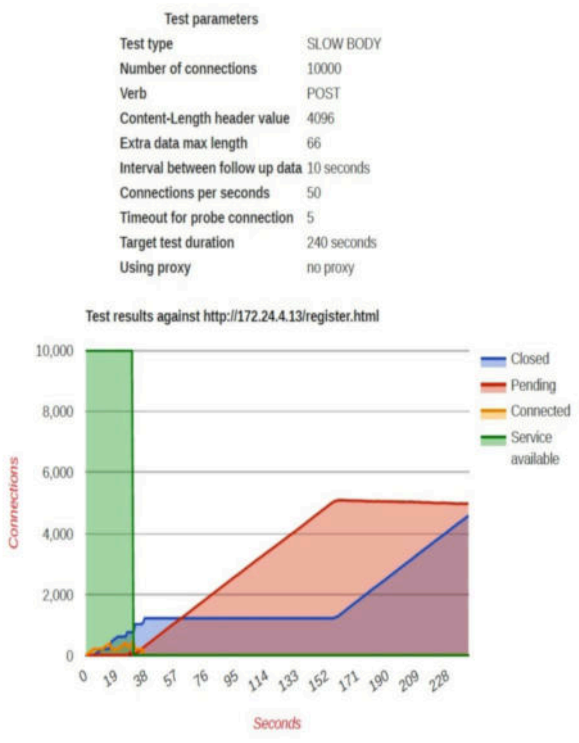
FIG. 5.2. *The slow HTTP Header attacks result*

moved into BLOCKED ZONE list. Any further request from the client is blocked, so the attacker thinks that the web server is down, but the requests from the attacker are not processed. The pending request from the attacker is growing in the very high rate as shown in Fig. 5.2.

The slow HTTP Body attack results are shown in Fig. 5.3, and the system detected the attack during when connection request reached 1̃700. After this point of time, any request from the attacker is not serviced by the web server. The pending requests are increasing very sharply as displayed in Fig. 5.3.

The performance result of the slow HTTP Read attack is depicted in Fig. 5.4. Figure 5.4 explains that read attacks are generating frequent zero window size to receive the response from the web server. The same behaviour has been detected by the system and moving such clients into the BLOCKED LIST zone. The incoming requests from those clients are not given to the web server for any services. In Fig. 5.4, the slow HTTP Read DDoS attacks are detected when the number of connection reached 1̃000. The detection happened sooner because of the clients continuous malfunctioning of reading the response from the server. The pending requests moved to peak value suddenly 8̃100, due to not serving any request from the client. The total number of connection accepted throughout the experiment is 2̃000. The efficient and early detection of slow HTTP Read attacks help the availability of the web server to the legitimate requests.

6. The conclusion and future directions. The authors discussed cloud computing and classification of cloud computing in details. The DDoS attacks and its types, the important application layer attack of the HTTP protocol is explained. The variation of HTTP DDoS attacks also captured. The slow HTTP DDoS attacks and various forms of slow HTTP attacks are described appropriately. The literature survey of related work and their gaps in addressing the slow HTTP DDoS attacks in the cloud is discussed. The proposed solution to detect such attacks in the cloud environment defined. The offered solution has been implemented and integrated into the OpenStack IaaS software. The details of the OpenStack cloud environment have been explained. The different types of slow HTTP DDoS attacks are carried out to the web server in the OpenStack using the slowHTTPTest tool. The performance and the result of the implemented solution are captured and



Test parameters

Test type	SLOW BODY
Number of connections	10000
Verb	POST
Content-Length header value	4096
Extra data max length	66
Interval between follow up data	10 seconds
Connections per seconds	50
Timeout for probe connection	5
Target test duration	240 seconds
Using proxy	no proxy

Test results against http://172.24.4.13/register.html

FIG. 5.3. *The slow HTTP Body attacks result*

discussed in detail.

The future enhancement of this work is to implement a model to detect, mitigate and prevent any form of the slow HTTP DDoS attack in the cloud environment. Evaluate the performance of the same against the multiple tools available on the internet for generating such slow HTTP DDoS attacks.

REFERENCES

[1] SALESFORCE, *https://www.salesforce.com/uk/blog/2015/11/why-move-to-the-cloud-10-benefits-of-cloud-computing.html*, 09-Jan-2019.

[2] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST), *https://www.nist.gov/programs-projects/nist-cloud-computing-program-nccp*, 09-Jan-2019.

[3] WHATISCLOUD.COM, *http://whatiscloud.com/cloud_delivery_models/index*, 09-Jan-2019.

[4] WHATISCLOUD.COM, *http://whatiscloud.com/cloud_deployment_models/index*, 09-Jan-2019.

[5] RADWARE, *https://security.radware.com/ddos-knowledge-center/ddospedia/dos-attack/*, 09-Jan-2019.

[6] IMPERVA INC., *https://www.incapsula.com/blog/top-10-cloud-security-concerns.html*, 09-Jan-2019.

[7] CLOUDFARE, LU-*https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/*, 09-Jan-2019.

[8] CLOUDFARE, *https://www.cloudflare.com/learning/ddos/ddos-attack-tools/r-u-dead-yet-rudy/*, 09-Jan-2019.

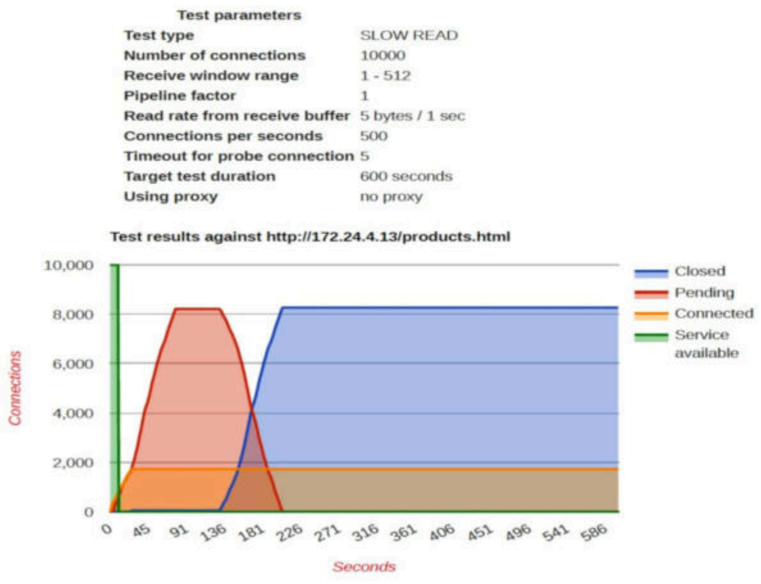
[9] CLOUDFARE, *https://www.cloudflare.com/learning/ddos/ddos-low-and-slow-attack/* , 09-Jan-2019.

[10] TANISHKA SHOREY, DEEPTHI SUBBAIAH, ASHWIN GOYAL, ANURAAG SAKXENA, ALEKHA KUMAR MISHRA, *Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools*, 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), DOI: 10.1109/ICACCI.2018.8554590, (2018).

[11] CLIFFORD KEMP, CHAD CALVERT, TAGHI M. KHOSHGOFTAAR, *Utilizing Netflow Data to Detect Slow Read Attacks*, 2018 IEEE International Conference on Information Reuse and Integration (IRI), DOI: http://doi.ieeecomputersociety.org/10.1109/IRI.2018.00023, (2018), pp. 108–116.

[12] KIWON HONG, YOUNJUN KIM, HYUNGOO CHOI, AND JINWOO PARK , *SDN-Assisted Slow HTTP DDoS Attack Defense Method* , IEEE Communications Letters, Vol. 22, Iss. 4 , DOI: 10.1109/LCOMM.2017.2766636, (2018), pp. 688–691.

[13] SHUNSUKE TAYAMA, AND HIDEMA TANAKA , *Analysis of Slow Read DoS Attack and Communication Environment*, Mobile and Wireless Technologies 2017, Lecture Notes in Electrical Engineering 425, DOI 10.1007/978-981-10-5281-1_38, (2018), pp. 1718–1724.



Test parameters

Test type	SLOW READ
Number of connections	10000
Receive window range	1 - 512
Pipeline factor	1
Read rate from receive buffer	5 bytes / 1 sec
Connections per seconds	500
Timeout for probe connection	5
Target test duration	600 seconds
Using proxy	no proxy

Fig. 5.4. *The slow HTTP Read attacks result*

[14] Aanshi Bhardwaj, Atul Sharma, Veenu Mangat, Krishan Kumar and Renu Vig , *Experimental Analysis of DDoS Attacks on OpenStack Cloud Platform* , Proceedings of 2nd International Conference on Communication, Computing and Networking, Lecture Notes in Networks and Systems 46, (2019)

[15] A.Dhanapal, and P. Nithyanandam , *An OpenStack based cloud testbed framework for evaluating HTTP flooding attacks*, Wireless Networks - Springer, DOI: 10.1007/s11276-019-01937-4, (2019), pp. 570–575

[16] Ghafar A. Jaafar, Shahidan M. Abdullah, and Saifuladli Ismail, *Review of Recent Detection Methods for HTTP DDoS Attack*, Journal of Computer Networks and Communications - Hindawi, DOI: https://doi.org/10.1155/2019/1283472, (2019), Volume 2019, Article ID 1283472, 10 pages

[17] Trung V. Phan, and Minho Park, *Efficient Distributed Denial-of-Service Attack Defense in SDN-Based Cloud*, IEEE Access, DOI: 10.1109/ACCESS.2019.2896783, (2019)

Edited by: Anand Nayyar
Received: Mar 7, 2019
Accepted: Apr 4, 2019

DOI 10.12694/scpe.v20i2.1520
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 299–315. http://www.scpe.org © 2019 SCPE

STATIC LOAD BALANCING TECHNIQUE FOR GEOGRAPHICALLY PARTITIONED PUBLIC CLOUD

MANDEEP KAUR* AND RAJNI MOHANA†

Abstract. Large number of users are shifting to the cloud system for their different kind of needs. Hence the number of applications on public cloud are increasing day by day. Handling public cloud is becoming unmanageable in comparison to other counterparts. Though fog technology has reduced the load on centralized cloud resources to a remarkable extent, still load handled at cloud end is significantly high. Geographic partitioning of public cloud can resolve these issues by adding manageability and efficiency in this situation. Dividing public cloud in smaller partitions opens ways to manage resources and requests in a better way. But, partitioned clouds introduce different ends for submission and operations of tasks and virtual machines. We have tried to handle all these complexities in this paper. Proposed work is focused upon load balancing in the partitioned public cloud by combining centralized and decentralized approaches, assuming the presence of fog layer. A load balancer entity is used for decentralized load balancing at partitions and a controller entity is used for centralized level to balance the overall load at various partitions. In the proposed approach, it is assumed that jobs are segregated first. All the jobs which can be handled locally by fog resources are not forwarded to the cloud layer directly. Those are processed locally by decentralized fog resources. Selection of an appropriate Virtual Machine (VM) for filtered set of job, which are forwarded to cloud environment, is done in three steps. Initially, selecting the partition with a maximum available capacity of resources. Then finding the appropriate node with maximum available resources, within a selected partition. And finally, the VM with minimum execution time for a task is chosen. Results are compared with the results produced in this work with First Come First Serve (FCFS) and Shortest Job First (SJF) algorithms, implemented in same setup i.e. partitioned cloud. This paper compares the Waiting Time, Finish Time and Actual Run Time of tasks using these algorithms. After initial experimentation, it is found that in most of the cases, while comparing above parameters, the proposed approach is producing better results than FCFS algorithm. But results produced by SJF algorithm produce better results. To reduce the number of unhandled jobs, a new load state is introduced which checks load beyond conventional load states. Major objective of this approach is to reduce the need of runtime virtual machine migration and to reduce the wastage of resources, which may be occurring due to predefined values of threshold. The implementation is done using CloudSim.

Key words: Load Balancing, Public Cloud, Cloud Partitioning, Geographically Partitioned Cloud, Static Load Balancing, Overloaded Node Management, Fog Computing, IoT Devices, Fog stratum.

AMS subject classifications. 68M14, 68M20

1. Introduction.

Cloud Computing has the potential to affect a large part of IT industry. Nowadays, developers need not to concern about the over-provisioning or under-provisioning of resources. Elasticity of resources without spending a large amount of funds is a unique concept of its kind [2]. Basic characteristics of Cloud Computing as user friendliness, virtualization, automatic adaptation, scalability, resource optimization, pay-per-use, service SLAs, infrastructure SLAs etc. are attracting users in masses [26]. Cloud Computing is proving itself so beneficial, but there is a requirement of efforts to maintain performance and efficiency of services. Handling concurrent jobs, users and processes through such a large set of machines is a very difficult task. Load balancing is one of the serious concerns in success of cloud computing. Emerging trends like edge or fog computing are being integrated with cloud computing. These share the load on public cloud by shifting significant amount of work from centralized public resources to local fog processing units. Fog/Edge computing is the future of cloud technology. Apart from heavy load challenge, another big concern is of imbalanced load among servers. Many kinds of troubles occur due to overloaded as well as under loaded servers. Under loaded servers cause inefficient energy consumption, inefficient use of resources and add-on to the management overheads. On the other hand, overloaded servers can cause delay in response, low speed of processing, decreased throughput, increased makespan etc. To handle both these situations it is very important to adopt a load balancing mechanism which can distribute tasks evenly among all available servers [28]. Recent growth in the field of IoT has further increased the challenges. As per CISCO cloud traffic is expected to rise upto 14.1 ZB per year by [27]. Data associated IoT devices are the main cause of this growth. Similarly, Internet of Vehicles (IoV) is also an emerging concept which is a subclass of IoT. It refers to the network of heterogenous sensors

*Computer Science Engineering and Information Technology, Jaypee University of Information Technology and School of Computer Applications and Lovely Professional University, India (mandeep.13695@lpu.co.in)

†Computer Science Engineering and Information Technology, Jaypee University of Information Technology, India (rajni.mohana@juit.ac.in)

installed on vehicles [5]. In all such applications major issue is of latency. To handle these latency issues a new term fog computing has been introduced. Fog computing is capable to reduce the operational cost of cloud data centres and helps to avoid revenue loss due to WAN propagation delays [22]. Need of optimal allocation of jobs to available fog resources has led the researchers towards load balancing in fog networks [21]. In paper [12] authors have presented an in-depth analysis of principles of Green-IoT. This chapter also highlights different kind of barriers challenging implementation of Green-IoT in real life. Load balancing can either be dynamic or static. In dynamic load balancing, load status of nodes is checked while jobs are running. During the execution process if any node gets overloaded then few of the jobs are shifted to some other node, with lesser load. This process is known as virtual migration. Static load balancing, efforts are made to distribute tasks equally among VMs. That means while initially putting tasks in VMs, possible efforts are made to keep task distribution even. This method of load balancing is very efficient as it will reduce the need of VM migrations. Run time VM migrations are a kind of interrupt or overhead during the execution period of tasks [17]. In this paper, we have focused on static load balancing part. Load management becomes even more challenging when we are considering a large public cloud. A public cloud has numerous nodes, scattered around in various geographic locations. Huge number of users and large amount of outsourced cloud data makes it very difficult to achieve the performance requirements and system usability [30]. Cloud Partitioning is one of the popular techniques used for load management in clouds. In this technique, clouds are divided into small groups of nodes. Small partitions are more manageable as compared to large, public set of nodes. So handling load will become simple and efficient by using this technique. This paper is an attempt to implement decentralized load balancing through cloud partitioning at an initial state of task submission.

Contribution The main contribution of this work is to design and evaluate a decentralized load management model for public clouds and fog layer in a partitioned environment. It can fully utilize the resources available with Datacentres. Our objective is to introduce a technique for load balancing in a partitioned cloud. The proposed approach suggests that in a public cloud, presence of a substantial number of nodes can be used as a privilege. Static load balancing is done depending upon a threshold value. This value denotes the allowable and optimal amount of load in a partition at a time. Once the load in a partition has reached at that threshold value, that node will no more be considered for job allocation until a few of its jobs have been finished. Considering that we are talking about a public cloud, there must be a scope of remaining resources to handle a few of the jobs, maybe with minor resource requirements. Hence, while calculating the load state of a partition, keeping records of load at all nodes under a partition, its available resources and required resources of jobs can be very helpful. Proposed work implements static load balancing for handling load mismatches during the initial allocation of virtual machines. Static load balancing gives an overall better performance.

Allocation done with a node with a current maximum capacity of resources will be more reliable and need of VM migrations during run time can be reduced. Decentralized load calculation and management through is tremendously capable to reduce execution time to compare, search and allocation process.

Few extra efforts are involved in calculating load status of all the nodes in all partitions. Use of decentralized approach reduces this overhead. Finally, central controller will be getting details from partition balancers hence there is very less communication required across the partitions.

Paper Structure In following sections of this paper, a decentralized load management technique is discussed. Section 2 contains the literature review and existing work done in this area. In Section 3, the proposed approach is discussed in detail. An experimental, simulation-based evaluation and comparative study of our approach along with the retrieved measurements are presented and discussed in Section 4. Finally, Section 5 concludes this paper and identifies paths for future work.

2. Related Work.

Many researchers have addressed the load balancing challenges in their work. In [7] a load balancing technique the cloud public cloud based upon cloud partitioning. They proposed to switch load balancing technology based upon the situation of load at a time. This algorithm applies game theory to handle the partitions with high load status. Though this algorithm needed further experiment to determine the efficient refresh rate and load degree calculation mechanism. The tradeoff was that much testing is required to guarantee system availability and efficiency.

Authors in [24] have proposed a system to perform load balancing as well as dynamic cloud partitioning. Major components of this approach are Partition Manager: responsible for assigning a job to a partition and Job Distributor: which decides the node to which this task can be allocated. To ensure effective distribution of load, partitions are created dynamically. This is done with the help of the honey bee algorithm. This paper left scope to improve the transparency and cloud division beyond geographic limits. They have used game theory to improve the efficiency in a cloud environment. According to the researcher the network is divided into cluster using a clustering algorithm. Every node is a part of a cluster. Every cluster has an Inter Cluster Communication (ICC) node. Clustering is done while initialization of the network.

In work presented by [8], authors have addressed the issues of Service availability and reliability, Lack of Service Level Agreements, Customer data security and privacy and Government compliance regulation requirements. They have proposed intelligent workload factoring service for proactive workload management with a fast and frequent data detection mechanism at the core. This mechanism helps not only in factoring requests on the volume of data, but also on contents of data. Major design goals of this workload refactoring include avoiding overloading scenario through load redirection, smoothening the workload dynamics in base zone application platform, and making flash cloud zone application platform agile through load decomposition. There are few improvements required, such as data security management for hybrid platform, handling data replication and consistency in the flash crowd zone, and load balancing schemes are implemented in two zones.

Authors in [14] have proposed a solution to calculate load degree based upon the turnaround time. This approach helps load balancer to improve load balancing strategies of a load balancing model in a public cloud. An idea of preliminary evaluation of a cloud partitioning approach is expressed in [23], which distributes task execution requests in a volunteer cloud. Validation is done through a simulation-based statistical analysis, using the Google workload data trace. Evaluations of this model are based on comparison between the results of purposed approach and the results of an unpartitioned cloud, using same random data sets. This approach has improvement scope, such as using the bio-inspired solution to add more sophistication. Adding lightweight performance monitors can help in improving performance. A workload classification mechanism can make workload management better. User Module submits the VM allocation request to Front-End Module, which is endowed with best-fit heuristic and random load balancing heuristic. It submits the VM allocation requests to the Server Manager Module. The Server Manager Module is responsible for balancing the load across hosts and performs energy-efficient server consolidation. This Module uses two threshold values, load-related migration threshold and load-related activation threshold to accomplish its job. Authors tried to remove the scheduling decision from Job's Critical Path to improving scheduling decision accuracy in [1]. For this, they have used Schedule First and Manage later approach and job replication. In their proposed model, they suggest replication of jobs and distributing these replicas to multiple servers. Whichever server will pick this job first will notify other servers having a copy of the same job. The main goal of this approach is to make task scheduling simple by removing load balancing tasks from the process of VM allocation. Though, this approach adds a signal propagation delay in the processing time.

In work expressed in [11], authors have tried to highlight multiple issues such as elasticity, energy efficiency, and high operational costs etc. For this, they have developed a combination of algorithms for initial VM placement, partial VM migration, and full VM migration. The initial VM placement algorithm can create or destroy VMs at any time, based on the current load at that VM. A Partial VM Migration algorithm is used by over utilized compute nodes to shifts few of the jobs to one or more other compute nodes. Full migration algorithm is used to shift full load of a compute node if it is underutilized. The purpose of this migration is to reduce the overall energy consumption of a Datacentre. A star-based partitioning and index algorithm for Resource Description Framework (RDF) data of robotic systems in [29] has devised a 3-step process. In first step a start structure is created by using MapReduce and HDFS to construct a coarsened weighted graph. In second step a balance partitioning algorithm is used to divide this graph. In third step a Compressed and Linked S-tree index is proposed to improve the query efficiency. In their paper [18] have discussed the need of fog computing by exploring its taxonomy, applications and various technologies involved in it. This paper has very well described the difference between cloud computing and fog computing. A mobility and heterogeneity-aware partitioning algorithm to support cross-domain resources partitioning, is presented in [9]. In this paper a service popularity-based smart resources partitioning (SPSRP) scheme is proposed. The basic architecture of

SPSRP scheme decouples the computing control layer from data processing layer.

In [6] authors have discussed the tradeoff between power consumption and transmission delay in fog cloud computing. They have proposed an optimal workload allocation between fog and cloud resources. Objective achieved is to get job done with minimal power consumption and restricted transmission delay. For improving efficiency and security [15] have proposed a model to authenticate and compare the load status of various Edge Data Centres (EDCs). SDN-based modified constrained optimization particle swarm optimization (MPSO-CO) algorithm proposed by [25] uses the reverse of mutation particles and linear decrease inertia weight to enhance the performance of constrained optimization particle swarm optimization (PSO-CO). This technique results in reduced latency and improved QoS in Software Defined Cloud/Fog Networking (SDCFN). Technique proposed by [10] decouples computing control layer from data procession layer through a service popularity-based smart resource partitioning (SPSRP) controller. Table 2.1 presents the summarized view of research work done on static load balancing until now. In [16] authors have proposed a model based upon time-based data driven approach to predict load predictions in various in different utilization sectors. This model is capable to accurately predict the energy demand in residential and commercial sector of smart device users.

Authors of [4] have discussed about the variants of Ant Colony Optimization (ACO) and its role in solving discrete problems in various areas of science and engineering discipline, including load balancing. Finding an appropriate node with sufficient resources is a crucial part of load balancing. One such method to find efficient sources is a logarithmic spiral based local search strategy, namely logarithmic spiral local search (LSLS), suggested by [20]. In [19] authors have presented a deep understanding about taxonomy of fog computing, its differences from cloud computing and edge computing technologies, applications, emerging key technologies and various challenges involved in fog technology. Authors of [3] have presented an Artificial Bee Colony (ABC) Optimization algorithm.

As compared to the above approaches, our proposal suggests a combination of centralized and decentralized load management. It ensures the benefits of unlimited resources of public cloud while removing the complexities of handling the substantial number of nodes. Proposed model ensures the better utilization of resources in the partitioned public cloud environment. It also helps in retaining the individual identity of nodes in a partition to ensure the accessibility to load status information and other details of a node. Initially, the node with maximum available resource instances is chosen for job allocation. So, chances of dynamic VM migration can be reduced. The objective is to increase the efficiency of load balancing mechanism lowering the usage and management costs.

3. Proposed Approach. This section contains the presentation model, task distribution between various modules and algorithm of our proposed approach for load balancing in public clouds and fog layer. The cloud is initially partitioned statically based on geographic location. In proposing a model, the primary objective is to reduce the size of public cloud by dividing it into multiple partitions and to apply a decentralized load balancing mechanism to ensure optimal utilization of resources. It is assumed that a significant part of total load is processed at fog layer. These jobs, restricted to reach cloud layer include the latency sensitive and security sensitive jobs. Fog computing environment is required to be implemented in time critical applications. In current scenario data is generated by IoT devices and processed in cloud environment. Most operations of these IoT devices depend upon data transmission to-and-from cloud layer. This is not considered feasible while keeping in view the time and distance constraints. Similarly, the applications involving sensitive data which user is not intended to share on public cloud can be shifted from cloud to fog environment.
So, cloud environment is responsible for jobs left unprocessed by fog environment.

As shown in Fig. 3.1, major entities in this model are clients, partitions, nodes, load balancers and controllers. A node is a provider which holds physical cloud resources which a client requires to process his job. Within that partition, a load balancer is deployed to keep track of load status of the partition. This balancer collects load-related data of individual nodes, compares it with threshold value to determine in which load category this partition falls currently. Same information is conveyed to the controller for making a final decision related to task allocation to a partition and anode. Balancers are local to partitions whereas a centralized controller is an entity which can keep track about all the partitions through information received from load balancers.

The proposed model is a composition of three software modules which are Client (CM), Balancer (BM) and

Table 2.1
Summarized view of research work done on static load balancing

Title	**Proposed Model**	**Limitation**
A Load Balancing Model Based on Cloud Partitioning for the Public Cloud [7]	Load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations like high, low, normal load status. This model applies game theory to load balancing strategy to improve the efficiency in the public cloud environment.	1. Lacks detailed cloud division methodology, 2. Effectively determining the refresh period, 3. Devising a good algorithm to set Load degree, 4. Testing is required to compare different load balancing strategies, 5. Many tests are to be performed to guarantee system availability and efficiency.
A novel approach for Dynamic Cloud Partitioning and Load Balancing in Cloud Computing Environment [24]	The strategic model that performs load balancing as well as dynamic partition of the nodes of different cloud. Game theory is used to load balancing strategy to improve the efficiency in the cloud environment	1. Need to increase levels of transparency 2. Requires effective technique in updating the status report. 3. The time intervals are not very well managed, 4. Dynamic balancing technique could be made dynamic, 5. Finding alternatives to geographical cloud division methodology.
A Cluster-Based Load Balancing Algorithm in Cloud Computing (Surbhi Kapoor, 2015)	A distributed algorithm for load balancing in the master-slave architecture that outperforms the Closest Datacentre algorithm in terms of task distribution across the system and optimal system performance	To evaluate effectiveness of the proposed model in scenarios where a node belongs to more than one cluster and we believe that effective load balancing could be achieved in this case as well
Resource Allocation Issues and Challenges in Cloud Computing (S.Thamarai Selvi, 2014)	Addressed issues are: 1. Resource Provisioning, 2. Job Scheduling, 3. Resource Overbooking, 4. Scalability, 5. Pricing, 6. Load Balancing, 7. Multitier applications, 8. Availability, 9. Overheads in Network I/O Workloads, 10. QoS constraints	1. Lacks elasticity, 2. Need to minimize the costs and maximize resource utilization, 3. Need to assure high availability for long running jobs, 4. Better parallel task scheduling
Proactive Workload Management in Hybrid Cloud Computing+B17 [8]	Addressed issues are: 1. Service availability and reliability, 2. Lack of Service Level Agreements, 3. Customer data security and privacy, 4. Government compliance regulation requirements	1. Load balancing schemes are implemented in two zones, 2. Efficient data replication and consistency management in the flash crowd zone, 3. Better security management for a hybrid platform
Load Degree Calculation for the Public Cloud based on the Cloud Partitioning Model using Turnaround Time [14]	Solution to calculate Load degree of a node in the public cloud based on the Turn Around Time	Lacks efficiency of algorithms
A Workload-Based Approach to Partition the Volunteer Cloud [23]	A preliminary evaluation of a cloud partitioning approach to distribute task execution requests in volunteer cloud. Comparison between the results of the proposed model, i.e. partitioned cloud and an unpartitioned cloud which uses the same random tasks.	1. Requires more sophisticated algorithms such as bio-inspired solutions, 2. Adding lightweight performance monitoring, 3. Better workload classification mechanisms
Replication-based Load Balancing [1]	Removing the Scheduling Decision from Jobs Critical Path, Improving Scheduling Decision Accuracy. Schedule First and Manage later approach is used by implementing Job replication.	1. Presence of signal propagation delay, 2. Devise single parameter configuration that is optimal for all systems.
Decentralized and Energy-Efficient Workload Management in Enterprise Clouds [11]	An issue addressed: 1. Elasticity, 2. Scalability, 3. High operational costs, 4. Efficient energy consumption. Three algorithms are introduced which are: 1. Initial VM Placement, 2. Partial VM Migration, 3. Full VM Migration,	1. Needs decentralized workload manager in an open-source cloud operating system, such as e.g., OpenStack, 2. Can use additional parameters into load-balancing algorithms,
A Partitioning and Index Algorithm for RDF Data of Cloud-based Robotic Systems [29]	Considers the 2-hop star structure as the basis object and proposes a partitioning and index algorithm	More orientation towards data structure operations. Output is dependent upon datasets, query type and strategies for graph operations.

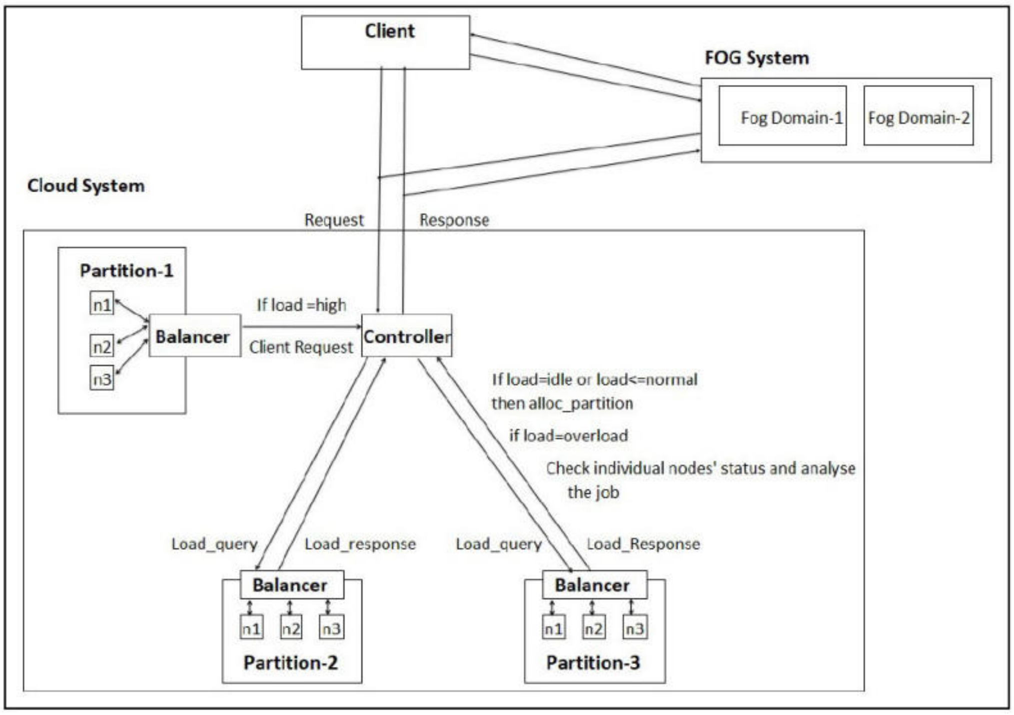


Fig. 3.1. *Architecture of Proposed Model for Load Balancing in Partitioned Cloud*

Centralized Controller (CCM). These three modules have pre-assigned roles to play during initial VM allocation to a task. These modules are capable to interact with each other wherever required. Following three modules are involved in the overall processing of the proposed approach. These are:

1. Client Module (CM) : Main responsibility of this Module is to filter the jobs to be processed by cloud environment and to submit those requests to a server node, through a broker. While submitting a request, the client is responsible to provide details about the resource requirement of that task. These details are very much important while choosing a node for final job allocation.
2. Balancer Module (BM) : Balancer module is local to partitions. They receive a notification whenever a client submits a task to a node belonging to that partition. Balancer has overall responsibility to manage load in a partition. It must keep record of load status of each node in the partition. The balancer is the accessing point of load status information of a partition, hence a controller always interacts with this module.
3. Centralized Controller Module (CCM) : This Module works like a centralized entity. It collects information from Balancer of all the partitions and makes decisions based on that information. Controller module is invoked by a Balancer whenever it finds all its local nodes are fully occupied. The controller asks all other partitions Balancer for their load status and finally directs job to the partition which fulfills following criteria.
   (a) Maximum amount of available resources.
   (b) Enough resources to cater the needs of the job.

In the proposed model, a Client Module (CM) captures the details of the service request being submitted to a node by a user. These modules prepare an estimate of resources required by the current job. Once the resource requirement details are ready, a notification is sent to the Balancer (BM) of the partition to which this node is associated with.

On receiving this notification Balancer (BM) will search resource availability status of that specific node and

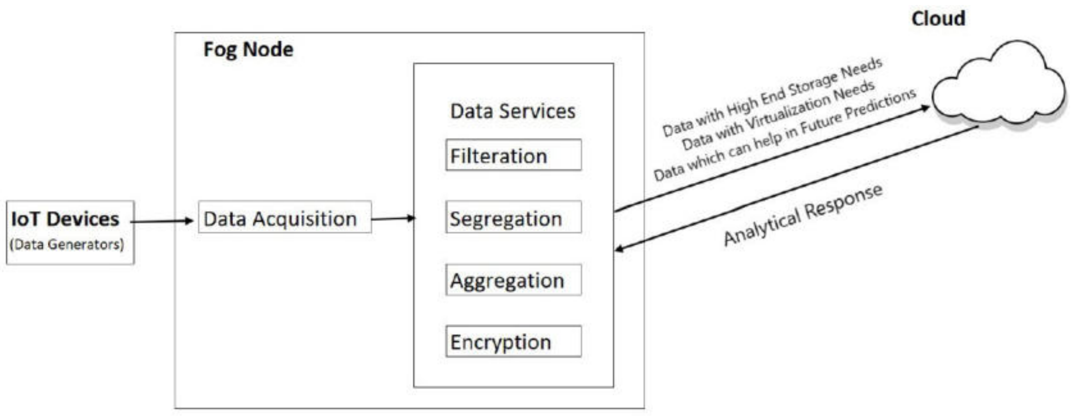


FIG. 3.2. *Distribution of Tasks between Fog and Cloud.*

will compare it with the resource requirement of the newly submitted job. If BM finds that sufficient number of resources are available, then it will allocate the job to the same node. Else it will compare jobs resource requirement with resource availability of other nodes in the same partition. If there exists any node(s) which are having sufficient resources to serve current job, then job is allocate to that node. If there is more than one such node, then the best possible allocation is done by allocating jobs to the node which has maximum availability of resources so that there are least chances of a need of VM migration.

If BM finds no node within partition which can serve job, then it will notify the Central Controller Module (CCM). CCM has the centralized access to the status of all partitions via their load balancers. Upon receiving notifications from BM, CCM will ask for fresh status information from all other BMs. On receiving this, first it will check the load status of all partitions. If it finds a single partition with normal or low load status, then the next step is to search for a node in that partition which has maximum available resources with it and the job is assigned to that node. Load status of nodes and the partition is updated in terms of resource availability. If there are multiple partitions then the one with max availability status is chosen and following same process a node is chosen in it.

There is a possibility that the load status of all the partitions is high, i.e. all the partitions have number of nodes with high load status than predefined threshold value. In such situation, resource availability of individual nodes is checked. Reason behind this step is that may be the percentage of nodes with high work load is exceeding threshold. But still, in remaining percentage there can be few nodes with low or normal load. The available resources of these nodes are usually ignored, but can be utilized in a better manner.

In rare situations, CCM would have not find any individual node with low or normal load status. In that case client module is notified about the non-availability of resources and job is kept unallocated till some of the running jobs are not complete and resources allocated to them are released. Summarized working of all the 3 modules is described in Table 3.1.

Job Distribution among cloud and fog Layer

It is a crucial decision-making point to distribute the jobs among fog and cloud, which highly depends upon their contribution in this fabricated system. Cloud services are assumed to provide services as well as data analytics required in operations of IoT devices. Fog provides various types of data services in IoT environment such as Data Filtering, segregation, Aggregation, Data Encryption, Catching etc. [13]. Hence it can be concluded that fog environment will retain all job requests which are short-span and more frequent. On the other hand, cloud environment will handle jobs which are less frequent, long term and require large amount of resources.

In the proposed model, the load balancing algorithm is being applied on the jobs submitted to cloud.

Evaluating Load Status for Node Allocation

As described earlier the partitions groups certain number of data centers. Each datacenter accommodates one

Table 3.1
Role of various modules

Stepwise Process	**Client Module (CM)**	**Balancer Module (BM)**	**Central Controller Module (CCM)**
Step-1	Capture details of a service request being submitted by a user.		
Step-2	Based upon pre-defined categories, segregate the jobs deciding if it is to be processed locally in fog environment or cloud environment.		
Step-3	Send the job to appropriate end, based upon decision made in Step-2.		
Step-4	Prepare estimates of required resources.		
Step-5		If the current node is overloaded then notifies the Balancer Module of the current partition.	
Step-6		Notified about the new job submission. Receives resource requirement details from Client Module.	
Step-7		Traverse all the nodes in the partition and compares their resource availability status with resource requirement details.	
Step-8		Allocates job to the best suitable node based upon the resource availability. Or notifies the Central Controller Module if no node is available locally in the partition.	
Step-9			Notified by the Balancer Module, collects status information from all the balancer Modules.
Stepwise Process	**Client Module (CM)**	**Balancer Module (BM)**	**Central Controller Module (CCM)**
Step-10			Picks the best suitable partition based upon load status.
Step-11			Picks the best suitable node in that partition, based upon resource availability. Notify concerned Balancer Module.
Step-12		Allocate job to notified node. Update load status and resource availability details of partition and node respectively.	
Step-13			If all partitions have a high load status, then traverse the entire list of individual nodes and pick the one with maximum available resources.
Step-14			Compare this nodes resource with the requirement. If sufficient resources are available, Notify concerned Balancer Module.
Step-15		Allocate job to notified node. Update load status and resource availability details of partition and node respectively.	
Step-16			If none of the nodes possess sufficient resources, then notify the client Module.
Step-17	Get notification from Controller Module to wait.		
Step-18	Capture details when the job is processed and the response is sent to the client.		

or more hosts, which are physical instances of resources which a client can request. These hosts can create k number of VMs as per requirement. Task allocation is done to these Virtual machines (nodes). There is a need to evaluate the availability at two levels as given below:

1. Determining the partition Pi, to which a job can be assigned.
2. Determining the node Ni, inside partition Pi, to which a job can be assigned.

As partitions do not have any load status of their own, they depend upon the load status S of individual nodes grouped inside them. Four predefined load states are considered, which are common for partitions and nodes and are calculated as follows:

1. **Idle load state**
   (a) A node N_j falls under idle state if its available resources $R(n)_j$ exceed a threshold $t(n)_{idle}$. i.e.

   $$N_{idle} = R(n)_j > t(n)_{idle}$$

   where $t(n)_{idle}$ is the threshold at which a node is considered idle. N_{idle} shows the idle status of node N.

   (b) A Partition P_i falls under idle state if the number of idle nodes in this partition has exceeded a threshold $t(p)_{idle}$ i.e.

   $$P_{idle} = Count(N_{idle}) > t(p)_{idle}$$

   where t(p)idle is the threshold at which a partition is considered idle. Pidle shows the idle status of partition P.

2. **Normal load state**
   (a) A node N_j falls under normal state if its available resources $R(n)_j$ exceed a threshold $t(n)_{normal}$. i.e.

   $$N_{normal} = R(n)_j > t(n)_{normal}$$

   where $t(n)_{normal}$ is the threshold at which a node is considered normal. N_{normal} shows the normal status of node N.

   (b) A Partition P_i falls under normal state if the number of normal nodes in this partition has exceeded a threshold $t(p)_{normal}$ i.e.

   $$P_{normal} = Count(N_{normal}) > t(p)_{normal}$$

   where t(p)normal is the threshold at which a partition is considered normal. Pnormal shows the normal status of partition P.

3. **Overloaded load state**
   (a) A node N_j falls under ovld state if its available resources $R(n)_j$ exceed a threshold $t(n)_{ovld}$. i.e.

   $$N_{ovld} = R(n)_j > t(n)_{ovld}$$

   where $t(n)_{ovld}$ is the threshold at which a node is considered ovld. N_{ovld} shows the ovld status of node N.

   (b) A Partition P_i falls under ovld state if the number of ovld nodes in this partition has exceeded a threshold $t(p)_{ovld}$ i.e.

   $$P_{ovld} = Count(N_{ovld}) > t(p)_{ovld}$$

   where t(p)ovld is the threshold at which a partition is considered ovld. Povld shows the ovld status of partition P.

4. **Full load state** A Partition P_i falls under full state if the number of full nodes in this partition has exceeded a threshold $t(p)_{full}$ i.e.

$$P_{full} = Count(N_{full}) > t(p)_{full}$$

where t(p)full is the threshold at which a partition is considered full. Pfull shows the full status of partition P.

Task Assignment
The proposed system is focused upon static VM allocation in a partitioned cloud environment or fog environment. Clients of cloud and fog system submit their requests to nodes. If the current node is having sufficient resources to handle request, then it will process request itself. If no then a suitable node is searched in the current partition. If none of the nodes has sufficient resources, then search is continued in other partitions. The complete process is described below.

Whenever a task is submitted to a node N_j in a partition, initially the load state of same node $R(n)_j$ is evaluated. If found that the current node is capable to accept more jobs (i.e. N_{idle} or N_{normal}) with specified resource requirements, the task is assigned to it. In addition, the load status of the partition and available resources will be updated. If found that current node is not capable to accept task (i.e. N_{ovld}), then other nodes in same partition are evaluated for their load state.

Following possible results can be there after this load state evaluation:
Case-1: If a single node N_j is found with idle or normal load state, assign the task to this node, update the load status of partition $S(p)_i$ and node $R(n)_j$.
Case-2: If multiple nodes can accept the task, create a list of all such capable nodes $N_{capable}$.
Case-3: If no node in current partition is found with idle or normal load state, then let balancer handover the request to the controller. Now controller will call the balancers of other partitions and will collect the load status of all partitions by calling searchPartition () method.

Algorithm 3 Partition Selection

```
1: procedure PartitionSelection($taskId$, $taskLength$, $pesNumber$, $taskFileSize$, $taskOutputSize$, $utilizationModelCpu$, $utilizationModelRam$, $utilizationModelBw$)
2:     if $partition_{state}$ [ $partition_{id}$] = idle or $partition_{state}$ [$partition_{id}$]= normal) then
3:         $selected_partition \leftarrow partition_{id}$
4:         if $node_{state}$ [ $node_{id}$] = idle or $node_{state}$ [ $node_{id}$] = normal then
5:             $selected_node \leftarrow node_{id}$
6:         else
7:             for all $node_{id} \in partition_{id}$ do
8:                 if $node_{state}[node_{id}] \leftarrow high$ then
9:                     continue
10:                else
11:                    $selected_node \leftarrow node_{id}$
12:                end if
13:            end for
14:        end if
15:        $partition_{state}[partition_{id}]$ ++
16:    else
17:        searchPartition ($partition_{state}$ [ ], $node_{state}$ [ ] , $node_{mem}$, $node_{cpu}$,$mem_{req}$, $cpu_{req}$)
18:    end if
19: end procedure
```

Controller Module will traverse through all the remaining partitions by invoking their balancers. Wherever it finds a partition with P_{normal} or P_{idle} load state, it will add that partitions id in a list of capable partitions $P_{capable}$.

While traversing all the partitions three cases are possible:

Case-1: If found single partition then assign the task to the balancer Bi of that partition and update its load status.

Case-2: If multiple partitions are capable to accept the task, create a list of all such capable partitions. Once this list is complete, we can choose the best possible partition in the context of available resources through searchPartitionnohigh ().

Case-3: If no partition is found with idle or normal load state, try to find individual nodes in overloaded partitions P_{ovld}. Call searchPartitionhigh () method.

Algorithm 4 Determining Availability of Capable Nodes

```
    procedure SEARCHPARTITION($partition_{state}$ [ ] , $node_{State}$ [ ], $node_{mem}$, $node_{cpu}$, $mem_{req}$, $cpu_{req}$)
2:      for all $partition_{id} \in partition_list[]$ do
            if $partition_{state}[partition_{id}] = idle$ or $partition_{state}[partition_{id}] = normal$ then
4:              $selected_partition \leftarrow partition_{id}$
                if $node_{state}[node_{id}] = idle$ or $node_{state}[node_{id}] = normal$ then
6:                  $capable_i \leftarrow partition_{id}$
                    i++
8:              else
                    continue
10:             end if
            end if
12:         if $capable_{size} > 1$ then
                searchpartitionNoHigh ()
14:         else
                searchpartitionHigh ()
16:         end if
        end for
18: end procedure
```

Algorithm 5 Partition Selection if Available with Normal State

```
    procedure PARTITIONHIGH($capable$[ ], $partition_{state}$ [ ], $node_{state}$ [ ], $node_{mem}$ [ ], $node_{cpu}$ [ ], $req_{mem}$,
    $req_{cpu}$)
        for all $partition_{id} \in capable[\ ]$ do
3:          if $max_{resource[i]} < capacityatpartition_{resource[i]}()$ then
                $max_{resource[i]} = capacityatpartition_{resource[i]}() maxnode_{resource[i]} = capable[k]$
            end if
6:          if $max_{resource[i+1]} < capacityatpartition_{resource[i+1]}()$ then
                $max_{resource[i+1]} = capacityatpartition_{resource[i+1]}()$
                $maxnode_{resource[i+1]} = capable[k]$
9:          end if
        end for
    end procedure
```

Algorithm 6 Partition Selection if Not Available with Normal State

```
    procedure PARTITIONNOHIGH($capable$ [ ], $partition_{state}$ [ ] , $node_{state}$ [ ], $node_{mem}$ [ ], $node_{cpu}$ [ ], $req_{mem}$, $req_{cpu}$)
        for all $partition_{id} \in capable[]$ do
            if $max_{mem} < capacityatnode_{mem}$ then
 4:             $max_{mem} \leftarrow capacityatnode_{mem}$
                $max_{n}ode_{mem} \leftarrow capable[i]$
            end if
            if $max_{cpu} < capacityatnode_{cpu}$ then
 8:             $max_{cpu} \leftarrow capacityatnode_{cpu}$
                $maxnode_{cpu} \leftarrow capable[i]$
            end if
            allocation( )
12:     end for
    end procedure
```

Algorithm 7 Calculation of Capacity of a Node

```
    procedure CAPACITYATNODERESOURCE($partitionid$, $nodestate$[ ] , $noderesource$[ ])
        for all $node_{id} \in partition_{id}$ do
            $capacity_{resource}[node_{id}] = node_{resource}[node_{id}] - node_{state}[node_{id}] * consumption$
        end for
 5:     return $capacity_{resource}$
    end procedure
```

All the nodes are given an amount of various resources. Which node is in an idle, normal or overload state, is entirely calculated based upon the available (free) amount of these resources. This amount is being said the capacity of a node for that resource. To calculate the capacity of nodes the amount of resources under use is deducted from total allocation of that resource in the node.

Algorithm 8 Calculation of Capacity of a Partition

```
    procedure CAPACITYATPARTITION($partition_{id}$,$node_{s}tate$[ ],$node_{r}esource$[ ])
        for all $node_{id} \in partition_{id}$ do
            $capacity_{resource}[node_{id}] \leftarrow node_{resource}[node_{id}] - node_{s}tate[node_{id}] * consumption$
        end for
        return $capacity_{resource}$
 6: end procedure
```

All the nodes are given an amount of distinct resources. Which partition is in an idle, normal, overload state, is calculated based on the number of nodes in that partition, falling within a category based on a threshold value for various load states. This load state is being considered as a capacity of a partition for that resource. To calculate the capacity of partitions, capacity of resources at each node for each resource is used as an input.

Once partitions with maximum capacity of various resources i.e. $Max(S(p)_j)resourcex)$ have been calculated and inside those partition, nodes with maximum capacity i.e. $Max(R(n)_j)resourcex)$ have been identified, we must select an appropriate node which can cater the requirement of all resources. For example, we can start with the node, which has maximum available memory. We must check if this node has enough processing units also. If yes, the task is allocated to this node. If no, then a node with maximum capacity of available CPU resources will be chosen and will be checked to see if it has enough memory available. A node allocation is possible only if it has sufficient instances of each resource available as per requirement of tasks. If no such node is found, then the task allocation is delayed until some partition is not capable to cater all resources requirement;

Table 3.2
Time and Space Complexity of Algorithms and its Modules

Sr. No.	Step	Time Complexity	Space Complexity
1.	Testing the current node for availability of resources	O (1)	O (1)
2.	Testing other nodes in the current partition	O(Count($P_{current}$(N)))	O (1)
3.	Testing other partitions		
	a) Each partition has same no. of nodes	O(Count(P)*Count(N)) where N represents the multiplication of partition count and node count	O(Count(P_i(N))) where N represents no. of nodes in a partition under consideration.
	b) Partitions have different no. of nodes	O(Sum(Count(P_i(N)))) where N represents sum of all the nodes in all partitions	O(Count(P_i(N))) where N represents no. of nodes in a partition under consideration
4.	Testing among multiple capable partitions .	O(Count($P_{capable}$)) where N represents no. of partitions	O(Count(P_i(N))) where N represents no. of nodes in partition under consideration
5.	Testing among multiple capable nodes	O(Count($P_{capable(i)}$(N))) where N represents no. of nodes in partition under consideration	O(Count(P_i(N))) where N represents no. of nodes in partition under consideration

allocation () function is called for this purpose.

Again in this situation, the decision is made based upon following criteria:

1. Find out the node with maximum availability of $resource_x$ i.e. $Max(R(n)_j)resource_x)$
2. Check if this node has sufficient availability of all other resources which can be calculated by comparing the availability with the requirement. i.e.
   $Max(R(n)_j)resource_x) > requirement_{resource(x+1,x+2,..,k)}$
3. If node N_j satisfies the above said criteria, then select this node and allocate the task. Else evaluate the node $Max(R(n)_{j)resource_x+1})$ and repeat these steps.
4. If none of the nodes satisfy above criteria, then balancer notifies the controller to search in other partitions.

Algorithm 9 Node allocation to a task

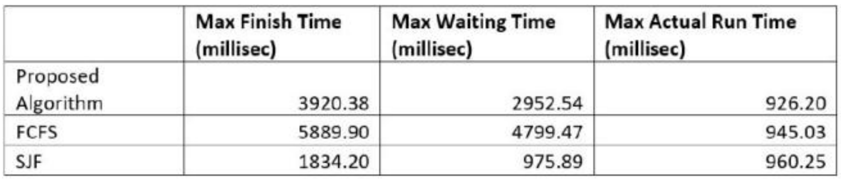
```
procedure ALLOCATION($capacityatnode_{mem}[\quad], capacityatnode_{cpu}[\quad], req_{mem}, req_{cpu}, max_partition_{mem}, max_partition_{cpu}$)
    if $req_{mem} <= capacityatpartition_{mem}(minpartition_{mem}, capacityatnode_{mem}$ then AND $req_{cpu} <= capacityatpartition_{cpu}(maxpartition_{mem} capacityatnode_{cpu}$
        $selected_partition \leftarrow maxpartition_{mem}$
    else if $req_{mem} <= capacityatpartition_{mem}(maxpartition_{cpu}, capacityatnode_{mem}$ then AND $req_{cpu} <= capacityatpartition_{cpu}(maxpartition_{cpu} capacityatnode_{cpu}$
        $selected_partition \leftarrow maxpartition_{cpu}$
    else
7:      statement: No allocation possible currently
    end if
end procedure
```

Time and Space Complexity During the entire process, time and space complexity varies as per entities under consideration. Initially, when the job was submitted at that time only one node availability status is evaluated. Hence, the complexity at this step is O (1). In all the other cases complexity class O(N) is applicable and value of N keeps on changing as per the case. Following is the summary of This entire process involves following steps along with their corresponding complexities.

In Table 3.2, P stands for Partition and N stand for a Node. Overall timIn the above table, P stands for Partition and N stand for a Node. Overall time and space complexity will be affected by the count of partitions

Table 4.1
Implementation Setup Parameters

Sr. No.	Parameter Name	Parameter
1.	No. of Partitions	2
2.	No. of Brokers	2
3.	No. of Datacenters	4 (2 Datacenters with each partition)
4.	No. of Hosts	2 Hosts each Datacenter
5.	No. of VMs	6 VMs in each Datacenter
6.	No. of tasks	40 tasks with each broker



	Max Finish Time (millisec)	Max Waiting Time (millisec)	Max Actual Run Time (millisec)
Proposed Algorithm	3920.38	2952.54	926.20
FCFS	5889.90	4799.47	945.03
SJF	1834.20	975.89	960.25

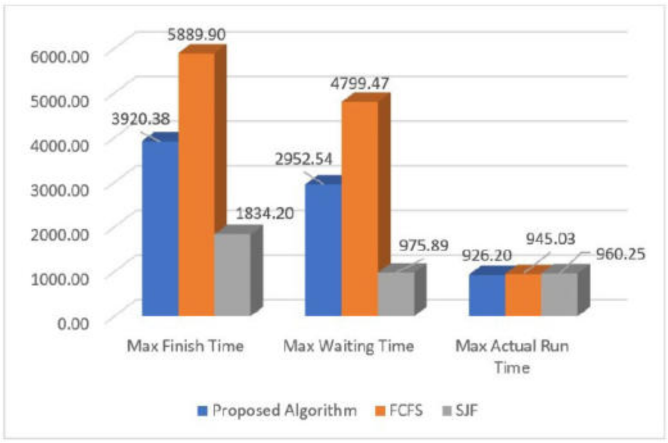


Fig. 4.1. *Average Max Execution Time Comparison*

and nodes in partitions. In all the cases linear alternatives are chosen to keep the complexities low.

4. Implementation and Results. This section shows the results of the evaluating the proposed algorithm. The algorithm is implemented in CloudSim by following setup details. The implementation is done on this small set of attributes for sampling purposes with certain assumptions.

For evaluating the performance of the proposed algorithm, the results have been compared with FCFS and SJF algorithms within same setup. The experiment is conducted by executing each algorithm 5 times. The VMs and tasks are heterogenous in their sizes. And the comparison results are shown below.

The results are compared for average of Finish time of tasks, waiting time of tasks and actual run time of tasks. Further, for all these three evaluation parameters are measured for total, average and maximum time of tasks. According to generated results proposed algorithm is producing better results when compared to FCFS algorithm, but the performance of SJF algorithm is better in all the cases. Fig. 4.1 shows the comparison of Maximum Execution time of a task during simulation. During most of the execution instances, it is found that the results of the proposed algorithm are better in case of Maximum Actual Run time of tasks. In Fig. 4.2 comparison is made between average total waiting time, average total finish time and average maximum actual time of all the tasks. It concludes that the proposed algorithm is capable to perform better than the FCFS algorithm in the same setup. Fig. 4.3 compares the actual run time of the tasks. Like other cases, it too shows that the proposed algorithm produces better results in comparison of FCFS but SJF algorithm is still better than this.

	Average Finish Time (millisec)	**Average Waiting Time (millisec)**	**Average Actual Run Time (millisec)**
Proposed Algorithm	1839.89	1110.11	668.08
FCFS	2352.28	1605.14	695.51
SJF	979.57	276.00	706.38

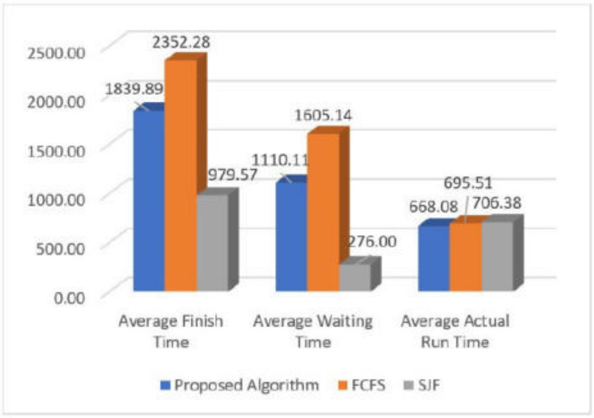


FIG. 4.2. *Average Time Comparison*

	Total Finish Time (millisec)	**Total Waiting Time (millisec)**	**Total Actual Run Time (millisec)**
Proposed Algorithm	93834.15	56615.47	34072.09
FCFS	160563.31	107551.58	46196.64
SJF	39182.74	11040.08	28255.36

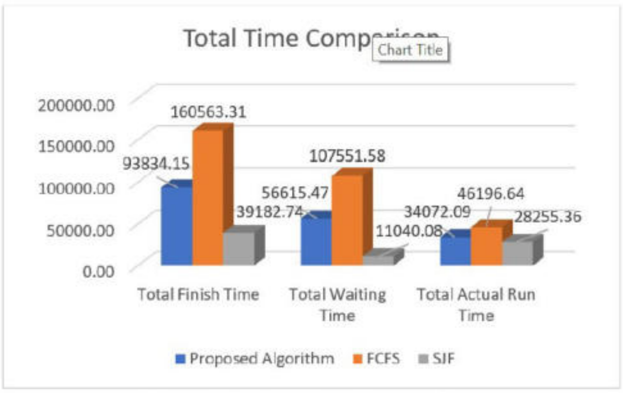


FIG. 4.3. *Total Time Comparison*

From the experimental results above, it can be seen that in most of the cases the output from SJF algorithm is better than proposed model. It looks impressive when talking about small scale local applications. But when we are focusing upon public cloud, the number of submitted jobs can be huge and their length unpredictable. In such environment, chances of resource starvation increase in manifolds. The consequences can be in terms of delayed response, longer waiting time for larger jobs, violated SLAs and so many others. In short, SJF algorithm focuses on jobs on the basis of a single parameter i.e. job length. This approach cannot be considered very beneficial for huge systems such as public cloud.

5. Conclusion. In the light of results produced till now the proposed algorithm works better in many of the cases. Choosing virtual machines on the basis of minimum execution time is helpful in reducing the waiting time of tasks. Also, adding one more load state will certainly reduce the number of unhandled tasks. But further work can be done by finding techniques to improve the Makespan, Failure/non-served instances of tasks and better ways to share the tasks load among available VMs. Better evaluation parameters can be associated and results can be improved to out-perform more advanced and complicated algorithms. Better techniques are required for heterogenous tasks and VMs which are a reason to add non-predictive load allocation requirements. Introduction of fog stratum in this system can further share the load by shifting latency sensitive jobs to the fog devices. Locally processing a share of jobs at fog layer will prove helpful in reducing bandwidth congestion, traffic flow over the network and wide area network propagation delays. Fog layer also allows a user to choose the optimal options in terms of compatibility, minimal geographical distance and decentralized/local communication and processing. Combining public cloud partitioning and fog computing can produce much more efficient results. Finally, partitioning technique can be very much useful to handle public clouds.

REFERENCES

[1] Ariel Orda Amir Nahir and Danny Raz. Replication-based load balancing. Transactions on Parallel and Distributed Systems, pages 1–15, 2015.

[2] Danny Raz Amir Nahir, Ariel Orda. Replication-based load balancing. IEEE Transactions on Parallel and Distributed Systems, 27:494–507, feb 2016.

[3] G. Suseendran Anand Nayyar, Vikram Puri. Logarithmic spiral based local search in artificial bee colony algorithm. Balas V., Sharma N., Chakrabarti A. (eds) Data Management, Analytics and Innovation. Advances in Intelligent Systems and Computing, 839:513–525, 2018.

[4] Rajeshwar Singh Anand Nayyar. Ant colony optimization computational swarm intelligence technique. 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2016.

[5] Rajkumar Buyya Anton Beloglazov, Jemal Abawajy. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Elsevier, 5:20068 – 20082, 2017.

[6] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H. Luan, and Hao Liang. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet of Things Journal, 3:1171–1181, 2016.

[7] Junjie Pang Gaochao Xu and Xiaodong Fu. A load balancing model based on cloud partitioning for the public cloud. TSINGHUA SCIENCE AND TECHNOLOGY, pages 34–39, 2013.

[8] Kenji Yoshihira Hui Zhang, Guofei Jiang and Haifeng Chen. Proactive workload management in hybrid cloud computing+b17. IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, 11:90–100, 2014.

[9] Gaolei Li, Jun Wu, Jianhua Li, Kuan Wang, and Tianpeng Ye. Service popularity-based smart resources partitioning for fog computing-enabled industrial internet of things. IEEE Transactions on Industrial Informatics, pages 4702–4711, 2018.

[10] Gaolei Li, Jun Wu, Jianhua Li, Kuan Wang, and Tianpeng Ye. Service popularity-based smart resources partitioning for fog computing-enabled industrial internet of things. IEEE Transactions on Industrial Informatics, pages 4702–4711, 2018.

[11] Gavriil Tzortzakis Michael Pantazoglou and Alex Delis. Decentralized and energy-efficient workload management in enterprise clouds. IEEE TRANSACTIONS ON CLOUD COMPUTING, 10:1–14, 2017.

[12] Ahmed Hamza Nour Mostafa, Ismaeel Al Ridhawi. An intelligent dynamic replica selection model within grid systems. Proceedings of the 8th IEEE GCC Conference and Exhibition, Muscat, Oman, pages 1 – 6, 2015.

[13] Aditya Brahmachari Pijush Kanti Dutta Pramanik, Saurabh Pal and Prasenjit Choudhury. Processing iot data: From cloud to fogits time to be down to earth. ResearchGate, pages 124–148, 2018.

[14] Pankaj Sharma Priti Singh. Load degree calculation for the public cloud based on cloud partitioning model using turnaround time. International Journal of Computer Science and Information Technologies, 2015.

[15] Deepak Puthal, Mohammad S. Obaidat, Priyadarsi Nanda, Mukesh Prasad, Saraju P. Mohanty, and Albert Y. Zomaya. Secure and sustainable load balancing of edge data centers in fog computing. IEEE Communications Magazine, 56:60–65, 2018.

[16] Sudeep Tanwar Shriya Kaneriya, Anand Nayyar, Jai Prakash Verma, Sudhanshu Tyagi, Neeraj Kumar, M. S. Obaidat, and Joel J P C Rodrigues. Data consumption-aware load forecasting scheme for smart grid systems. 2018 IEEE Globecom Workshops (GC Wkshps), 2019.

[17] Rimmy Yadav ; Avtar Singh Sidhu. Fault tolerant algorithm for replication management in distributed cloud system. 2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE), pages 78–83, 2015.

[18] Rajesh Kumar Simar Preet Singh, Anand Nayyar and Anju Sharma. Fog computing: from architecture to edge computing and big data processing. The Journal of Supercomputing, pages 1–36, 2018.

[19] Simar Preet Singh, Anand Nayyar, and Rajesh Kumar Anju Sharma. Fog computing: from architecture to edge computing and big data processing. The Journal of Supercomputing, pages 1–36, 2018.

[20] Anand Nayyar Sonal Sharma, Sandeep Kumar. Logarithmic spiral based local search in artificial bee colony algorithm. Duong T., Vo NS. (eds) Industrial Networks and Intelligent Systems. INISCOM 2018, 257:15–27, 2019.

[21] Stavros Souravlas and Angelo Sifaleras. Trends in data replication strategies: a survey. International Journal of Parallel, Emergent and Distributed Systems, pages 1 – 19, 2017.
[22] Angelo Sifaleras Stavros Souravlas. Binary-tree based estimation of file requests for efficient data replication. IEEE Transactions on Parallel and Distributed Systems, 28:1839 – 1852, 2017.
[23] Antonio Scala Stefano Sebastio. A workload-based approach to partition the volunteer cloud. IEEE Conference on Collaboration and Internet Computing, pages 2010–2018, 2015.
[24] Divya Mohandass Suguna R and Ranjani R. A novel approach for dynamic cloud partitioning and load balancing in cloud computing environment. Journal of Theoretical and Applied Information Technology, 62:662–667, 2014.
[25] Chenhua Shi Xiuli He, Zhiyuan Ren and Jian Fang. A novel load balancing strategy of software-defined cloud/fog networking in the internet of vehicles. China Communications, 13:140–149, 2016.
[26] Jui-Pin Yang. Elastic load balancing using self-adaptive replication management. IEEE Access, 5:7495–7504, 2017.
[27] Jui Pin Yang. On minimizing energy cost in internet-scale systems with dynamic data. IEEE Access, 5:20068 – 20082, 2017.
[28] Jui-Pin Yang. Intelligent offload detection for achieving approximately optimal load balancing. IEEE Access, 6:2169–3536, 2018.
[29] Hongmin Wang Yonglin Leng, Zhikui Chen and Fangming Zhong. A partitioning and index algorithm for rdf data. IEEE Access, pages 29836 – 29845, 2018.
[30] Juan F. P'erez Zhan Qiu. Evaluating replication for parallel jobs:an efficient approach. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS), pages 2288–2302, 2016.

Edited by: Anand Nayyar
Received: Mar 9, 2019
Accepted: Apr 3, 2019

DOI 10.12694/scpe.v20i2.1525
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 317–333. http://www.scpe.org © 2019 SCPE

ANALYTICAL AVAILABLE BANDWIDTH ESTIMATION IN WIRELESS AD-HOC NETWORKS CONSIDERING MOBILITY IN 3-DIMENSIONAL SPACE

MUKTA[*] AND NEERAJ GUPTA[†]

Abstract. Estimation of available bandwidth for ad hoc networks has always been open and active challenge for the researchers. A lot of literature is proposed in the last 20 years to evaluate the residual bandwidth. The main objective of the work being admission of new flow in the network with the constraint that any existing current transmission is not affected. One of the prime factors affecting the estimation process is the collision among packets. These collisions trigger the backoff algorithm that leads to wastage of the usable bandwidth. Although a lot of state of art solutions were proposed, but they suffer from various flaws and shortcoming. Other factor contributing to the inaccuracy in existing solution is the mobility of nodes. Node mobility leads to instability of links leading to data losses and delay which impact the bandwidth. The current paper proposes an analytical approach named Analytical Available Bandwidth Estimation Including Mobility (AABWM) to estimate ABW on a link. The major contributions of the proposed work are: i) it uses mathematical models based on renewal theory to calculate the collision probability of data packets which makes the process simple and accurate, ii) consideration of mobility under 3-D space to predict the link failure and provides an accurate admission control. Extensive simulations in NS-2 are carried out to compare the performance of the proposed model against the existing solutions.

Key words: QoS, available bandwidth, analytical, collision, mobility, fixed point analysis, 3 dimensions

AMS subject classifications. 68M10, 68M20, 60K05, 60K25

1. Introduction.

Recently, the development of multimedia applications like live movies, video on demand, e-learning etc. has necessitated the provision of QoS in MANETs. All these applications have diversified requirement in terms of QoS parameters. Some are sensitive to delay while others may demand guaranteed bandwidth for the smooth flow in the network. Thus, an accurate prediction of the available resources is required for the QoS flow guarantee. Bandwidth is one of the scarce resources and should be distributed wisely among the different flows such that admission of new flows doesn't deteriorate the performance of existing flows. Therefore, an accurate estimation of its availability is much required for the true admission control. Imprecise estimation of the ABW leads to false admission control whose bandwidth consumption is more than the estimated ABW. Some approaches like BRuIT [1], CACP [2], AAC [3] didn't address the impact of collisions while estimating the ABW thus provide false admission control, whereas ABE [4] addressed this issue but may results erroneous estimation. It was observed that estimation errors are mainly due to two reasons: first is the bad computation of the considered network criteria such as collision and integrating the same in the estimation; the second is failure to capture each of the network criteria affecting the bandwidth.

ABE [4] focused on few main challenges to evaluate the ABW on a link: idle period synchronization, collision probability and average backoff period between two transmissions. Authors in [4] utilizes the rate of exchange of HELLO packets to compute the collision probability. To handle the discrepancy that arises due to the smaller size of HELLO packets, the resultant is interpolated using Lagrange interpolating polynomial to get the collision rate of data packets. This leads to followings limitations: i) polynomial is calculated at a node for a specific scenario and same is applicable for all nodes or for any scenarios, ii) experiment needs to be executed in advance to calculate the collision probability, iii) HELLO packets that are sent late or lost due to network congestion or overloaded medium are combined with the computed collision rate which overestimates its impact. To remove all these limitations, we proposed to use an analytical approach using fixed point analysis based on renewal theory for the computation of collision probability under heterogeneous network conditions. Most of the available literature assumed homogeneous condition to simplify the process involved. But real time network conditions are heterogeneous in nature consisting of nodes having different capabilities and resources such as transmission rate, battery life, radio range, etc. The major benefits of our proposed solution include: predict the result set when the network parameters get varied, simple and easy approach to compute the collisions without using Hello packets.

[*]Computer Science Department, K. R. Mangalam University, Gurgaon, India, (mukta.mittal2006@gmail.com).
[†]Computer Science Department, K. R. Mangalam University, Gurgaon, India, (neerajgupta3729@gmail.com).

Another cause of error is neglecting the important criteria such as mobility which is not addressed in many of the available literature. Mobility of nodes causes the frequent link breakage and re-construction of links leading to delay and loss of data packets. To overcome the above issue, the current paper considered the bandwidth loss due to mobility of nodes under 3-dimensional space and observed its effects on admission control. Results obtained using our proposed solution is compared with existing solutions which show the improvement done by our proposed approach.

The paper is organized as follows: Section 2 gives an insight into existing available bandwidth estimation techniques and admission control solutions. Section 3 presents our proposed approach AABWM for the estimation of ABW considering all the network criteria affecting the bandwidth under heterogeneous conditions. Section 4 introduces the mobility model under 3-dimensional space to predict the link persistent factor for the improvement in ABW estimation and thus the admission control. Section 5 presents the comparative analysis of results obtained using our proposed solution with the existing approaches under two cases: stationary nodes and mobile nodes. Finally, Section 6 concludes the paper.

2. Related work. The bandwidth estimation techniques can be classified into three main categories: Active techniques, Passive techniques and Analytical techniques. Active techniques emit probe packets at multiple rates between sender and receiver node. By measuring the packets inter-arrival time at the receiver node estimates the end-to-end ABW along a path. The emission rate of these probe packets is increased gradually by the sender node until the congestion arises in the network. SLoPS [5] and TOPP [6] approaches come under this category. Active techniques burden the network with probe packets which can lead to network congestion and consuming precious network resources. Loss of probe packets due to network congestion or overloaded medium may provide the erroneous measurement. These drawbacks make such techniques inefficient to be employed in mobile ad hoc networks. To avoid these drawbacks, the passive techniques monitor the radio channel activities including transmission, reception or idle periods of channel in its vicinity over a certain period of time. The available bandwidth is determined by evaluating the channel usage ratio of nodes in the network. These techniques are non-intrusive in nature. Some major solutions that falls in this category includes: BRuIT [1], CACP [2], AAC [3]. Most of these techniques failed to address the problem associated due to collisions among the packets. The packet collisions consumes the network resources and adversely affect the ongoing transmission in the network.

Authors in [4] addressed these challenges and proposed a solution by incorporating the impact of collisions and time period consumed due to backoff on the bandwidth estimation. To calculate the collision probability [4] utilizes the HELLO packets. By counting the total number of received HELLO packets in a given interval of time and comparing this number with the actual number of received HELLO packets provides the collision probability of HELLO packets. The authors interpolate the resultant using Lagrange interpolating polynomial to compute the collision probability for the data packets of varying size. This approach suffers from the limitations as mentioned in Sec. 1 which makes this technique inefficient for ABW estimation. IAB [7] is a similar approach as that of [4] except that it differentiates the channel busy state due to packet transmission or reception from that of carrier sensing mechanism. Authors in [8] proposed a solution DLI-ABE for bandwidth estimation by employing the Distributed Lagrange interpolating polynomial which is to be calculated separately for each node in any scenario before transmitting the data. All the above-mentioned approaches rely on frequent exchange of Hello packets for calculating the collision probability which contradicts the non-intrusive nature of the passive technique. Moreover, the above literature assumes that the position of the node is static, i.e. there is no mobility. No satisfactory solution is obtained from any of above techniques to estimate the collision probability and their effect on backoff. Analytical techniques provide the reliable solution for the above problem. These techniques are having advantage of predicting the result set by varying the network parameters. Different analytical models available in the literature are based on: i) Bi-dimensional Markov chain, ii.) Means value argument, iii) fixed point analysis based on renewal theory. Based on the traffic conditions, these networks are further divided into two sub-categories: saturated and non-saturated networks. In saturated conditions node always has next packet available to transmit after completing the current transmission; whereas, in non-saturated condition node's buffer may be empty some time.

Author in [9] presented the first analytical model based on the bi-dimensional Markov chain to analyze the performance of IEEE 802.11 DCF under the assumption of saturated and idle channel conditions. The models

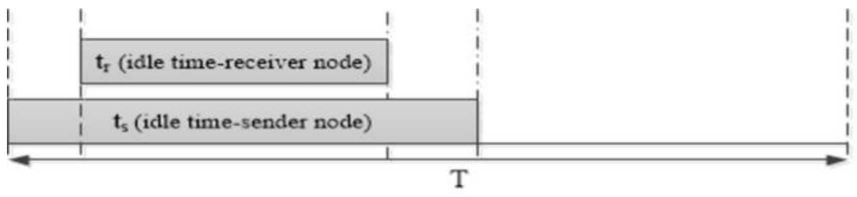

t_r (idle time-receiver node)
t_s (idle time-sender node)
T

FIG. 3.1. *Total overlap of idle time periods.*

based on the Markov chain were extended to incorporate various challenges like error-prone channel, non-saturated environment, heterogeneous traffic conditions and many other. The major work in literature can be referred in [10, 11, 12, 13, 14, 15]. The solution based on Markov chain are complex and mathematical expressions are computationally extensive. Authors in [16] presented a simple approach using fixed point analysis based on renewal theory. The results obtained are synchronous with the result of [9]. Authors in [17] extended the approach of [16] for non-saturated network condition. Most of analytical models didn't addresses the problem associated with mobility of nodes. The authors in [18] presented a novel approach named Improved Bandwidth Estimation through Mobility incorporation (IBEM) for the estimation of ABW on a link by incorporating the mobility criteria during bandwidth estimation. The authors in [18] ignore the effects of elevation and height while predicting the link failure due to mobility of nodes. Authors in [19] calculate the collision probability based on Markov chain under the assumption of saturated condition.

3. AABWM approach for ABW estimation. The main contribution of this paper is to determine the available bandwidth in mobile ad hoc networks using the analytical models based on renewal theory. To address the various challenges for predicting ABW in mobile ad hoc networks the work is divided into four sections: i) idle period synchronization ii) impact of collision probability iii) effect of average backoff period iv) mobility issues.

3.1. Synchronization of idle time periods between sender and receiver node. For efficient communication to happen, the information about the bandwidth consumption at each node and its neighborhood is much necessitated. The ABW at a node is evaluated by monitoring the radio channel activities of the node in its vicinity. A node is considered as busy when it is either transmitting or receiving information on the channel. Thus, the ABW at a node is evaluated by measuring the idle and busy period of nodes in the network. The upper bound for estimating ABW can be expressed as [20]:

$$ABW_i = \frac{t_i}{T}.C_{max} \tag{3.1}$$

where, t_i represents the idle periods sensed by node i during the total measurement duration T and C_{max} is the raw channel capacity.

Let t_s and t_r represents the idle periods of sender s and receiver r respectively, evaluated using (3.1) during the measurement period T. Fig. 3.1 and Fig. 3.2 depicts the two extreme cases (full overlap and no overlap) of node's overlap period respectively to evaluate the ABW on a link.

Fig. 3.1 represents the case where idle period of both sender node and receiver node completely overlap to each other. In other words, the channel is idle at both the ends make transmission of packet a successful event. Thus, the communication is feasible only when the idle periods of sender (t_s) and receiver (t_r) overlap each other for the period of the transmission as stated in [3]. This type of communication is possible using a precise clock synchronization mechanism. Thus, the ABW on a link constituting between sender s and receiver r is given by [3] as:

$$ABW_{link(s,r)} = \min(\frac{t_s}{T}, \frac{t_r}{T}) \tag{3.2}$$

Fig. 3.2 represent the case when the idle periods of both nodes (sender and receiver) are never synchronized to each other despite the medium availability at both sides. Thus, in this scenario no communication is feasible

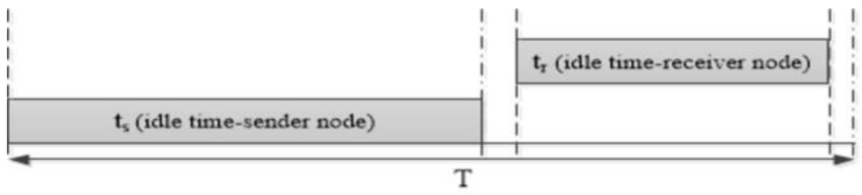


FIG. 3.2. *No overlap of idle time periods.*

and the available bandwidth on a link is counted as null. It can be concluded that the ABW on a link is dependent on the overlap period of idle time at both sender node and the receiver node for successful communication to happen.

Authors in [4] propose to incorporate a probabilistic approach and used the average of the idle time periods of sender and receiver to evaluate the ABW on a link. Thus, the ABW on a link is given as:

$$ABW_{ABE(s,r)} = \frac{t_s}{T} \cdot \frac{t_r}{T} . C_{max} \tag{3.3}$$

Where t_s and t_r represent the idle periods of sender(s) and receiver(r) respectively, during the measurement period T; C_{max} is the maximum channel capacity.

3.2. Collision Probability using Fixed point analysis (FPA). The ABW calculated using (3.3) provides a conservative solution for estimation of ABW. Collision impacts the bandwidth estimation process adversely. Collisions can occur due to various reasons including Hidden terminal problem and Exposed terminal problem. A packet transmitted by a node may collide with the other packet transmitted by other nodes in its vicinity leading to delay and wastage of bandwidth. Thus, it is imperative to accurately predict the collisions in the network otherwise the estimation becomes fallacious. The authors in [4] addressed this issue and presented a mathematical framework for the estimation of ABW on a link on account of the collision. Mathematically the same can be represented as:

$$ABW_{ABE(s,r)} = \frac{t_s}{T} \cdot \frac{t_r}{T} . C_{max} . (1-p) . (1-k) \tag{3.4}$$

Where p represents the collision probability of data packets, and k represent the proportion of bandwidth consumed due to backoff mechanism in the event of the collision.

To calculate the collision probability the authors in [4] make use of the HELLO packets. A counter was deployed at the receiver end to count the number of HELLO packets that are received at the destination in real time. The collision probability of HELLO packets is calculated by dividing the value of the above counter by the actual number of the HELLO packets that should have been received in idealized conditions. Since the size of the HELLO packets is small in comparison to the data packets the above results are interpolated to predict the collision probability of data packets. However, there are several drawbacks attached to this process. It can be observed from Fig. 3.3 that the estimated values of data packet collisions are far away from the real values of data packets collisions measured using NS2. This over-evaluation is because HELLO packets which are delayed or lost due to network congestion or overloaded medium are combined with the collision rate of HELLO packets and further interpolating the same overvalued its effect. This revealed the abortive behavior of [4] approach for the computation of collision probability.

To overcome the computation failure existing due to passive technique proposed in [4], our work uses an analytical approach using fixed-point analysis (FPA) based on renewal theory for the calculation of collision probability. The proposed solution is influenced by [17]. The main assumption being ideal channel conditions. The only reason for the unsuccessful packet transmission is the collision of packets. It is also assumed that all stations use the different backoff parameter thus having different collision probability (p_l) and attempt rate (or transmission probability) (τ_l) where l=1, 2... n. Let n be the number of contending nodes, the attempt by l^{th} node is successful if all other contending nodes are silent. Mathematically, it can be written as:

$$1 - p_l = \prod_{j \neq l} (1 - \tau_j); for j, l = 1, 2, ..., n \tag{3.5}$$

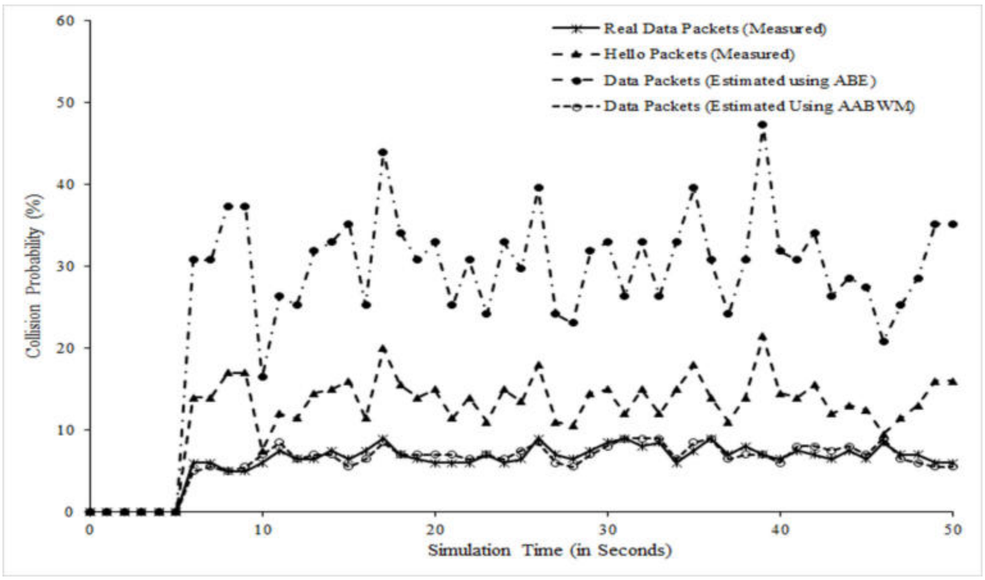


FIG. 3.3. *Estimated results of collision probabilities of packets.*

Let q denotes the probability of a non-empty buffer, i.e. at least one packet is available for transmission after transmitting the current one. According to [17] the attempt rate of a node in non-saturated network condition can be obtained by scaling down the attempt rate of a node in saturated condition with the probability of non-empty buffer. Thus, we have

$$\tau_l = q.\tau_{ls} \tag{3.6}$$

where τ_{ls} represent the l^{th} node's attempt rate (or probability) under saturated network conditions. To solve the fixed-point equation given by (3.5) and (3.6), there is a need to express τ_l i.e. q and τ_{ls}) in terms of collision probability (p_l).

The attempt rate of a node in saturated condition (τ_{ls}) is defined as the ratio of average number of attempts to the average number of backoff time (in slots) spent by a node till the successful transmission of the packet. According to [16], we have:

$$\tau_{ls} = f(p) = \frac{E[attempts]}{E[backoff]} \tag{3.7}$$

where the average attempt rate is given as:

$$E[attempts] = 1 + p_l + {p_l}^2 + ... + {p_l}^m \tag{3.8}$$

According to IEEE 802.11 standard whenever a packet suffers collision, the exponential backoff mechanism is triggered; the backoff associated with the initial transmission attempt by a node is b_0 and it is doubled after every unsuccessful attempt till the maximum limit m is reached. The average backoff time per packet is given as:

$$E[backoff] = b_0 + p_l(2b_0) + ... + {p_l}^{m'}(2^{m'}b_0) + ... + {p_l}^m(2^{m'}b_0) \tag{3.9}$$

where

$$b_0 = \frac{CW_0 + 1}{2} \tag{3.10}$$

CW_0 is the minimum contention window size = 31 slots. And, m' represent a constant where contention window size reaches its maximum value CW_{max}, represented as:

$$CW_{max} = 2^{m'} CW_0 \tag{3.11}$$

Thus, using (3.5) and (3.6) we get,

$$\tau_l = \frac{q.(1 + p_l + {p_l}^2 + ... + {p_l}^m)}{b_0 + p_l(2b_0) + ... + {p_l}^{m'}(2^{m'}b_0) + ... + {p_l}^m(2^{m'}b_0)} \tag{3.12}$$

For simplicity of expression we assumed Poisson traffic arrival rate, however the solution is equally applicable to other traffic arrival types also such as Constant Bit Rate (CBR), exponential on/off traffic and Pareto on/off traffic. Using queuing theory, the traffic intensity ρ is given as:

$$\rho = \rho(\lambda, p) = \lambda.ST \tag{3.13}$$

where λ is packet arrival rate, ST is a mean service time of a packet which is dependent on the average backoff time i.e. E[backoff] waiting till the successful transmission of the packet and the average slot time E_s depending on the channel activity i.e. idle slot time, successful attempt and collision occurrence. Thus, (3.13) can be re-written as:

$$\rho = \lambda.E[backoff].E_s \tag{3.14}$$

The average time E_s spent per slot is given as:

$$E_s = (1 - P_{tr})\sigma + P_{tr}P_{s_i}T_{s_i} + P_{tr}(1 - P_{s_i})T_{c_i} \tag{3.15}$$

where P_{tr} represent the probability that at least one transmission going on in the network; σ is the slot time and P_{si} represents the probability that the transmission by i^{th} station is successful.

$$P_{tr} = 1 - \prod_{i=1}^{n}(1 - \tau_j) \tag{3.16}$$

$$P_{s_i} = \tau_i \prod_{j \neq 1}(1 - \tau_j) \tag{3.17}$$

T_{s_i} and T_{c_i} represent the expected time for successful transmission and unsuccessful transmission respectively. These transmission times are governed by the access mechanism employed i.e. Basic access mechanism and RTS/CTS access mechanism.

$$T_{s_i}^{basic} = DIFS + T_{(H+DATA)} + SIFS + T_{ACK} \tag{3.18}$$

$$T_{c_i}^{basic} = DIFS + T_{(H+DATA)} + ACK_{timeout} \tag{3.19}$$

$$\begin{aligned} T_{s_i}^{RTS/CTS} &= DIFS + T_{RTS} + SIFS + T_{CTS} + SIFS \\ &+ T_{(H+DATA)} + SIFS + T_{ACK} \end{aligned} \tag{3.20}$$

$$T_{c_i}^{RTS/CTS} = DIFS + T_{RTS} + CTS_{timeout} \tag{3.21}$$

where DIFS is Distributed Inter-Frame Space time; SIFS is Short Inter-Frame Space time; T_{RTS}, T_{CTS}, and T_{ACK} are the time required for RTS, CTS, and ACK control packet transmission respectively; $T_{(H+DATA)}$ is the time required for the transmission of DATA packet along with the PHY header and MAC header. $CTS_{timeout}$

and $ACK_{timeout}$ are the waiting time required before declaring unsuccessful transmission for RTS and DATA packet respectively.

To solve (3.12), the value of q needs to be determined. The probability of non-empty buffer is dependent on the node's buffer size as well as offered load. We have taken two extreme cases of buffer sizes to evaluate the value of q; one for small buffer and another for infinite buffer. This is expressed by [17] as:

$$\begin{aligned} q =& 1 - e^{-\rho}; \textit{ for small buffer model} \\ & \min(1,\rho); \textit{ for infinite buffer model} \end{aligned} \tag{3.22}$$

Substituting the value of q from (3.22) in (3.12), we get the following equations under two different cases as considered:
Case-I: Small Buffer case

$$\tau_l = \frac{(1-e^{-\rho}).(1+p_l+{p_l}^2+...+{p_l}^m)}{b_0+p_l(2b_0)+...+{p_l}^{m'}(2^{m'}b_0)+...+{p_l}^m(2^{m'}b_0)} \tag{3.23}$$

Case-II: Infinite Buffer case

$$\tau_l = \frac{(\min(1,\rho)).(1+p_l+{p_l}^2+...+{p_l}^m)}{b_0+p_l(2b_0)+...+{p_l}^{m'}(2^{m'}b_0)+...+{p_l}^m(2^{m'}b_0)} \tag{3.24}$$

Solving the coupled nonlinear equations (3.5) and (3.23) provide the precise estimation of collision probability p_l and the attempt rate τ_l of the l^{th} node.

3.3. Special case for Homogeneous Network Model. Consider homogeneous network conditions where all stations are having same backoff parameters. Let n be the number of stations in the network, then average attempt rate τ and collision probability p of each station are same. Under the decoupling approximation, the probability of collision p of an attempt by a node is obtained by reducing the equations mentioned in Sec. 3.2 as:

$$1-p=(1-\tau)^{n-1} \tag{3.25}$$

The expected time spent per state reduces to:

$$E_s=(1-P_{tr})\sigma+nP_sT_s+(P_{tr}-nP_s)T_c \tag{3.26}$$

where, the probability of successful transmission P_s of any sending station is given by:

$$P_s=\tau(1-\tau)^{n-1} \tag{3.27}$$

The probability P_{tr} that at least one transmission is going on in the network reduces to:

$$P_{tr}=1-(1-\tau)^n \tag{3.28}$$

The probability of non-empty buffer under homogeneous network condition is obtained by re-writing (3.23) and (3.24). Thus, we have:
Case-I: Small Buffer case

$$\tau = \frac{(1-e^{-\rho}).(1+p+p^2+...+p^m)}{b_0+p(2b_0)+...+p^{m'}(2^{m'}b_0)+...+p^m(2^{m'}b_0)} \tag{3.29}$$

Case-II: Infinite Buffer case

$$\tau = \frac{(\min(1,\rho)).(1+p+p^2+...+p^m)}{b_0+p(2b_0)+...+p^{m'}(2^{m'}b_0)+...+p^m(2^{m'}b_0)} \tag{3.30}$$

Solving the coupled nonlinear equations (3.25) and (3.29) provide the estimation of collision probability p and the average attempt rate (or transmission probability) τ of the node. We calculated the collision probability using (3.25) for the simplicity reasons and plotted the same in Fig. 3.3 under same simulation scenario. It can be clearly seen that the computed values using our fixed-point approach (3.25) closely match with the real values of collision probability measured using NS2 which proves the accurate calculation of collisions in the network.

3.4. Bandwidth consumed due to backoff mechanism . Whenever a packet transmission occurs between sender and receiver node, the packets may suffer collision at sender s or receiver node r. In both cases the exponential backoff mechanism is triggered at sender node. The time lost due to backoff mechanism is not used for transmitting the packet, even if the medium is idle. Thus, a proportion of idle time wasted due to backoff mechanism having a significant impact on the ABW. The average backoff time associated with the collision probability p_l is obtained using (3.9). Thus, the loss of local ABW of sender node due to backoff mechanism can be mathematically represented as given in [19]:

$$LocalABW_s = \frac{(t_s - E[backoff])}{T} C_{max} \tag{3.31}$$

where t_s is the idle time of sender(s) obtained using (3.1) and E[backoff] is the average backoff time calculated using (3.9) and C_{max} is the raw channel capacity.

3.5. Link Prediction Factor in 3-Dimensional space . Mobility causes the frequent link breakage and re-construction leading to delay and data losses which eventually impact the ABW. Limited literature is available on the issue of mobility while calculating the available bandwidth. Most of the work addressing the mobility has considered two dimensional space only. The current work predicts the link expiration caused due to mobility of nodes under 3-dimensional space. The approach is equally applicable to all channel environments.

Fig. 3.4 represents the 3-dimensional mobility model for the movement of sender and receiver nodes. Let the initial position of sender station and receiver station be (X_s, Y_s, Z_s) and (X_r, Y_r, Z_r) respectively. Sender station is moving in 3-dimensional world with velocity V_s making an angle γ with z-direction and angle α with x-direction. As V_s is making an angle of γ in z-direction, the vector component of V_s in X-Y plane is $V_s.\cos\gamma$ and vector component of V_s in z-axis is $V_s.\sin\gamma$. Further, as the angle with respect to x-direction is α, the vector component of $V_s.\cos\gamma$ in x-axis is $V_s.\cos\gamma.\cos\alpha$ and in y-axis is $V_s.\cos\gamma.\sin\alpha$.

Receiver station is moving in 3-dimensional world with velocity V_r making an angle δ with z-direction and angle β with x-direction. As V_r is making an angle of δ in z-direction, the vector component of V_r in X-Y plane will be $V_r.\cos\delta$ and vector component of V_r in z-axis is $V_r.\sin\delta$. Further, as the angle with respect to x-direction is β, the vector component of $V_s.\cos\delta$ in x-axis is $V_s.\cos\delta.\cos\beta$ and in y-axis is $V_s.\cos\delta.\sin\beta$.

After time t let the coordinates of sender station and receiver station are (X_s, Y_s, Z_s) and (X_r, Y_r, Z_r) respectively. The new coordinates of the sender and receiver station can be mathematically written as:

$$X'_s = X_s + tV_s \cos\gamma \cos\alpha \tag{3.32}$$

$$Y'_s = Y_s + tV_s \cos\gamma \sin\alpha \tag{3.33}$$

$$Z'_s = Z_s + tV_s \sin\gamma \tag{3.34}$$

$$X'_r = X_r + tV_r \cos\delta \cos\beta \tag{3.35}$$

$$Y'_r = Y_r + tV_r \cos\delta \sin\beta \tag{3.36}$$

$$Z'_r = Z_r + tV_r \sin\delta \tag{3.37}$$

Therefore, the distance between sender and receiver after time t in X, Y and Z directions can be determined as:

$$X'_s - X'_r = (X_s - X_r) + t(V_s \cos\gamma \cos\alpha - V_r \cos\delta \cos\beta) \tag{3.38}$$

$$Y'_s - Y'_r = (Y_s - Y_r) + t(V_s \cos\gamma \sin\alpha - V_r \cos\delta \sin\beta) \tag{3.39}$$

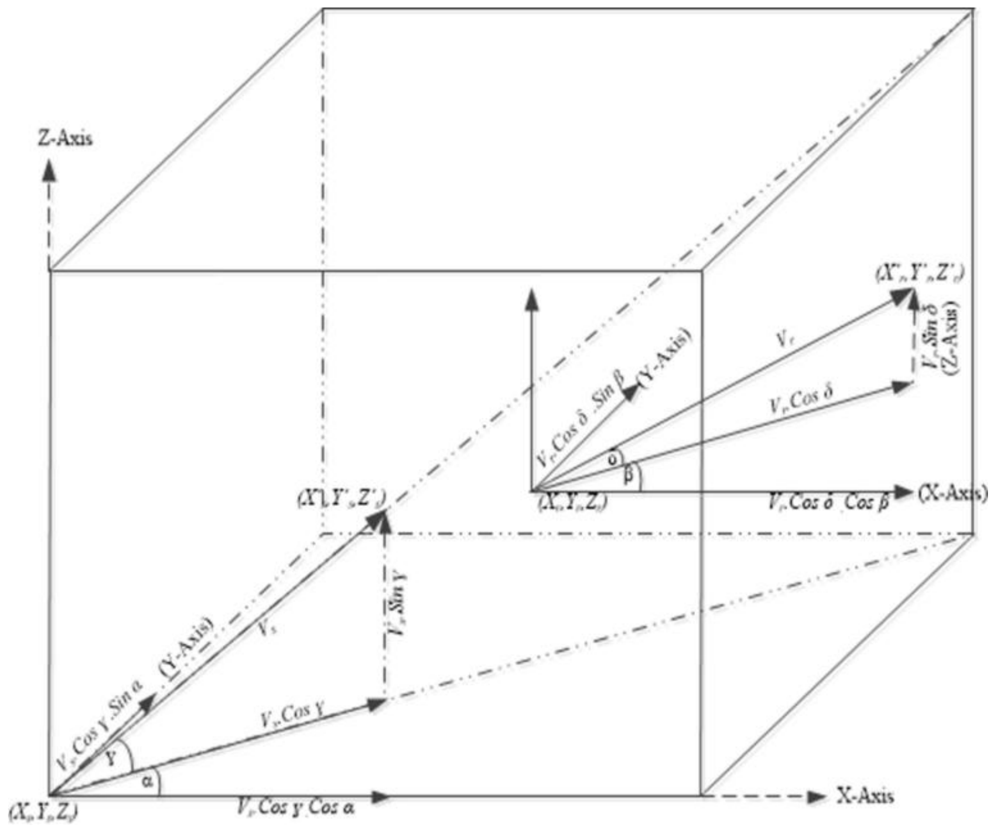


FIG. 3.4. *3-Dimensional reference model.*

$$Z'_s - Z'_r = (Z_s - Z_r) + t(V_s \sin\gamma - V_r \sin\delta) \tag{3.40}$$

Let the vector distance between sender and receiver after time t be D. Then, using Pythagoras Theorem the D can be calculated as:

$$D^2 = (X'_s - X'_r)^2 + (Y'_s - Y'_r)^2 + (Z'_s - Z'_r)^2 \tag{3.41}$$

Substituting the values from (3.38), (3.39) and (3.40) in (3.41), we get

$$\begin{aligned} D^2 = &= ((X_s - X_r) + t(V_s \cos\gamma \cos\alpha - V_r \cos\delta \cos\beta))^2 \\ &+ ((Y_s - Y_r) + t(V_s \cos\gamma \sin\alpha - V_r \cos\delta \sin\beta))^2 \\ &+ ((Z_s - Z_r) + t(V_s \sin\gamma - V_r \sin\delta))^2 \end{aligned} \tag{3.42}$$

This can be re-written as:

$$D^2 = (a + tf)^2 + (b + tg)^2 + (c + th)^2 \tag{3.43}$$

where $a = X_s - X_r$, $b = Y_s - Y_r$ and $c = Z_s - Z_r$ are the initial distance between sender and receiver station in X, Y and Z directions respectively. The relative velocity component between the two in X, Y and Z directions respectively can be mathematically expressed as $f = V_s.\cos\gamma.\cos\alpha V_r.\cos\delta.\cos\beta$, $g = V_s.\cos\gamma.\sin\alpha V_r.\cos\delta.\sin\beta$; and $h = V_s.\sin\gamma V_r.\sin\delta$. Solving (3.43) for the value of time t, we get:

$$t = \frac{-(af + bg + ch) \pm \sqrt{(f^2 + g^2 + h^2).D^2 - (ag - bf)^2 - (bh - gc)^2 - (ah - cf)^2}}{(f^2 + g^2 + h^2)} \tag{3.44}$$

Global positioning system in [21] facilitates the coordinate's information. With each Hello packet, station broadcasts starting and terminus coordinates along with its velocity. The sender station, receiving the hello packet from receiver station evaluates the link availability time duration t and thereby the mobility factor MM.

$$MM = \frac{t}{T} \tag{3.45}$$

where T is the measurement period, t is the duration for which the sender-receiver stations are within communication range.

The final ABW on a link(s,r) constituting between sender s and receiver r is computed by combining all the major factors discussed in the sub-sections of the current section (i.e. Sec. 3) of this paper. The final ABW estimated on link(s,r) is given as:

$$ABW_{AABWM} = (\frac{t_s - E[backoff]}{T}.\frac{t_r}{T}.C_{max})(1-p_l)(MM) \tag{3.46}$$

where t_s and t_r represent the sum of idle times sensed at sender and receiver node respectively computed using (3.1) during measurement period T; E[backoff] represent the local ABW wasted at the sender node as given by (3.9); C_{max} is the maximum channel capacity; p_l represents the collision probability of data packets as driven in Sec. 3.2 obtained by solving (3.5) and (3.23) or (3.24); and a MM link persistence factor representing the mobility effect of nodes in 3-dimensional space is given in (3.46).

3.6. Throughput efficiency (S). Saturation throughput of a system is defined as the maximum data that the system can transfer successfully under steady state conditions. Mathematically, the throughput S of a system can be expressed as the summation of each individual source station's throughput efficiency.

$$S = \sum\nolimits^{n}_{i=1} S_i \tag{3.47}$$

where S_i represents the throughput efficiency of i^{th} source station, $i \in [1, n]$ and n is the total number of source stations in the network. The throughput efficiency of the individual source station can be calculated as the fraction of the time used by source station for successfully transferring the payload information and is expressed as:

$$S_i = \frac{P_{si} L_i}{E_s} \tag{3.48}$$

where P_{si} represents the probability that the transmission by i^{th} station is successful and is derived in (3.17); L_i is the time taken to transmit payload data by i^{th} source station; and E_s is the expected time spent per state as derived in (3.15).

3.7. Protocol Design. To provide comparative analysis and visualizing the impact of mobility factor in (3.46), our proposed approach AABWM employs AODV [22] as routing protocol. Assuming a node having data to transmit. It first determines it's local ABW using (3.1) based on idle/busy periods sensed by the node by monitoring the radio channel activities. If it's local ABW is enough for the flow transmission, it broadcasts a route request packet (RREQ) consisting of the required bandwidth for its flow. All the nodes within its range receive this RREQ packet, and perform admission control by comparing the required bandwidth within the RREQ packet with the computed ABW using (3.46) on the link composed with its fore goer node. If bandwidth is enough, then node combines its own information with the fore goer and forwards the RREQ packet otherwise drop the packet. When the intended destination node receives the packet, it reverts with the uni-cast RREP (route reply packet) packet to the initiator along with the reverse path.

If the bandwidth estimation is done after admitting the flow and link has no enough bandwidth then the node sends a route error packet (RERR) to the initiator. On receiving the RRER packet, the initiator takes immediate action to prevent the losses and stops the ongoing transmission.

4. Validation and Simulations. In this section a comparative analysis is done between the AODV protocol, ABE [4], IAB [7], IBEM [18] and our proposed approach AABWM incorporated in AODV [22] protocol. The results are obtained through simulations in NS-2.

TABLE 4.1
Simulation parameters values for comparative analysis

Raw channel capacity	2 Mbps
Topology size	700x700x700 meter
Routing protocol	AODV
Medium access protocol	CSMA
Packet size	1000 Bytes
Transmission range	250 m
Carrier sensing range	550 m
Simulation time	100 seconds
Flow-1 CBR	250 kbps
Flow-2 CBR	500 kbps
Flow-3 CBR	400 kbps
Flow-4 CBR	550 kbps
Flow-5 CBR	450 kbps
Flow-6 CBR	300 kbps
Minimum contention window (CW_{min})	31 slot
Maximum contention window (CW_{max})	1023 slot
Maximum retry limit (m)	6

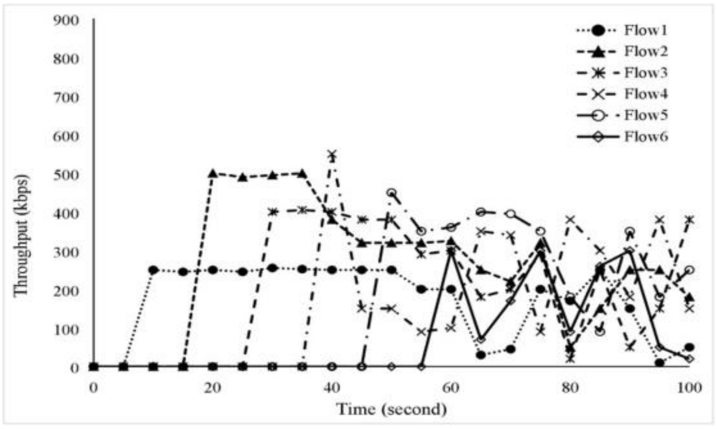


FIG. 4.1. *Throughput of flows using AODV [22] without any admission control.*

4.1. Simulation Scenario. A network consists of 48 nodes randomly distributed over a small area (700m x 700m x 700m) is considered. Six CBR single-hop flow connections (Flow-1 to Flow-6) are established. Each flow is having a different transmission rate to make the network heterogeneous. The parameters used for simulation are derived from IEEE 802.11b and are listed in Table 4.1.

4.2. Simulation results and Comparative Analysis. The comparative analysis is split into two cases: a) Case-I, consists of stationary nodes to evaluate the accuracy of our proposed approach AABWM with the traditional approaches; and b) Case-II, consists of mobile nodes which are free to move in 3-dimensional space, thereby giving us an opportunity to visualize its effect on link persistence and thus on the admission control.

Case-I: Considering no mobility of nodes

Flows 1 to 6 are generated at an interval of 10 seconds starting from t = 5 second. Fig. 4.1 to Fig. 4.5 plots the throughput of six flows obtained when AODV [22], ABE [4], IAB [7], IBEM [18] and AABWM approaches are activated respectively.

In AODV [22], all flows are admitted as no admission control is employed. However, with the introduction of flow-4(550kbps) at a time, t=35s channel capacity is reached; the performance of existing flows gets deteriorated. The throughput further got worse by the introduction of flow-5(450kbps) and flow-6(300kbps).

When ABE [4] is used, due to the underestimation of ABW, flow-4 (550kbps) and flow-5 (450kbps) are rejected. However, flow-6 (300kbps) got admission at t=55s as it was accommodated under ABW estimated by

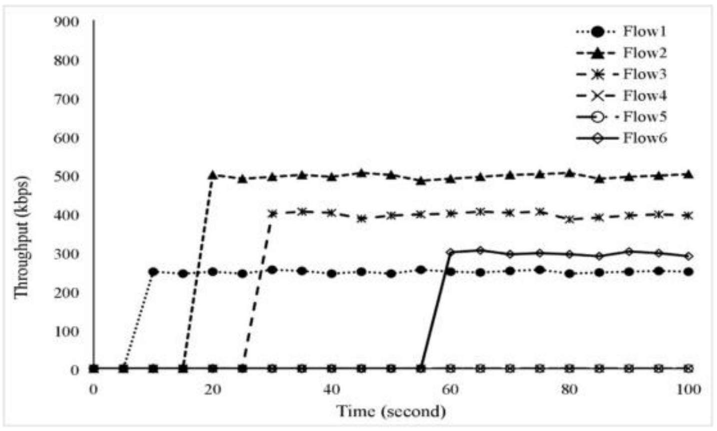


Fig. 4.2. *Throughput of flows using ABE [4] for admission control.*

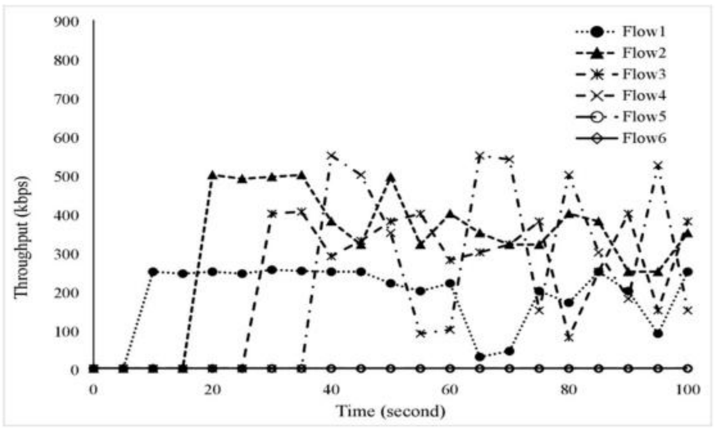


Fig. 4.3. *Throughput of flows using IAB [7] for admission control.*

ABE [4]. The system resources are not optimally utilized, the total system throughput is far below maximum channel capacity. This underestimation of ABW is due to the bad computation of collision probability and assimilates the same in bandwidth estimation.

In IAB [7], over-estimation of ABW takes place, thereby allowing flow-4(550kbps) which leads the channel congestion and all existing flows got suffered in terms of throughput performance. Due to collisions arising from network congestion, a lot of bandwidth got wasted; thereby degrading overall system performance. However, flow-5(450kbps) and flow-6(300kbps) are rejected as ABW estimated using IAB [7] is not enough for these flows.

Fig. 4.4 represent the throughput of six flows along the simulation duration when IBEM approach is enabled. To calculate the collision probability IBEM assumed the saturated condition as in [19] and ignore the empty state of nodes buffer. Thus, overestimating the impact of collisions leading to underestimate the ABW; thereby admission of flows (flow-4 and flow-5) is not permitted. Further, Flow-6(300kbps) is accepted as ABW estimated using IBEM is enough for this flow.

The throughput of all six flows is shown in Fig. 4.5 for proposed approach AABWM is enabled. Due to the more precise estimation of ABW, at t=35s. flow-4 (550kbps) is rejected due to unavailability of ABW, however at t=45s flow-5(450kbps) is allowed by the admission control protocol as its required bandwidth is below the estimated ABW obtained using (3.46). Further, at t=55s, flow-6 (300kbps) is rejected as no enough bandwidth left for allowing this flow admission. We can observe that all admitted flows exhibit a stable throughput performance. Moreover, the network resources are optimally utilized as clearly seen by the safe admission of flow-5 (550kbps) instead of flow-6 (300kbps).

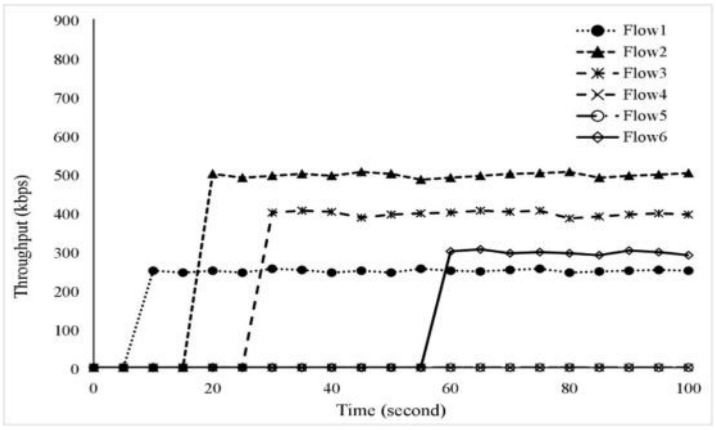


FIG. 4.4. *Throughput of flows using IBEM [18] for admission control.*

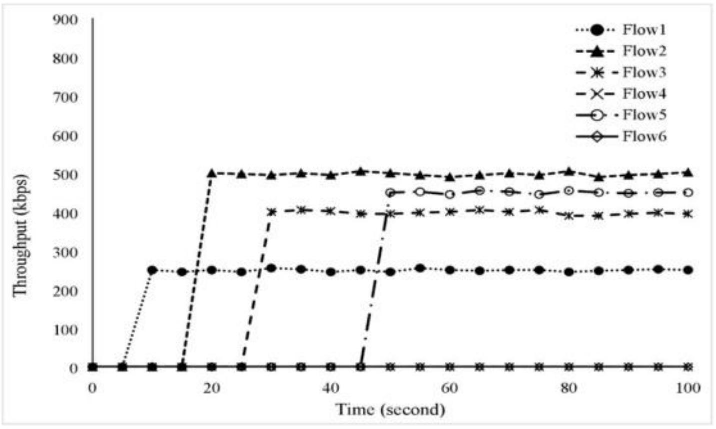


FIG. 4.5. *Throughput of flows using proposed AABWM for admission control.*

Case-II: Considering the 3-Dimensional mobility of nodes

To visualize the impact of mobility of nodes, the Random way point mobility model is used in the 3-dimensional space to assign random movement to the nodes with random direction and velocity. Initially the flows 1 to 6 are generated at random time interval. The start and stop of flows are automated by the admission control protocol depending on the ABW estimation technique employed. The simulation results obtained using AODV [22], ABE [4], IAB [7], IBEM [18] and AABWM under the influence of 3-dimensional movement of nodes is represented in Fig. 4.6 to Fig. 4.10.

Fig. 4.6 represents the throughput of all flows when the AODV [22] routing protocol is enabled. Absence of any admission control in AODV [22], all flows are admitted leading to high network congestion and increased collisions. There is no QoS guarantee given to any flow, even the flows having a shorter duration of link existence (Flow-4 and Flow-5) is allowed causing a lot of bandwidth and data loss.

Fig. 4.7 depicts the throughput of all six flows when ABE [4] approach is activated. Inability of ABE [4] to predict link failure caused due to mobility permit the admission of short persisted flow-4(550kbps) at time t=10s and flow-5(450kbps) at t=15s. Although the admitted flows are less in ABE [4] as compared to AODV [22] and flow's throughput are also stable, but still due to false admission of flows (Flow-4 and Flow-5) the network resource is not adequately utilized. Incomplete transmission due to link failure causes loss of data and thus the bandwidth.

Fig. 4.8 depicts the case of IAB [7], where an overestimation of ABW is observed. It is also inefficient in predicting the link expiration due to absence of mobility criteria in the ABW estimation. This leads to serious

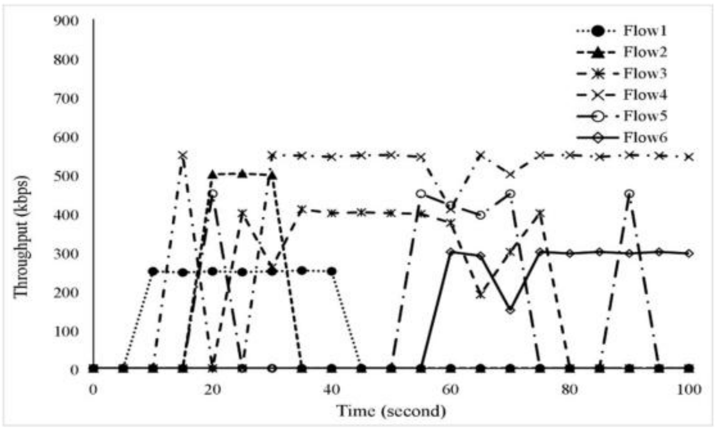


Fig. 4.6. *Throughput of flows using AODV [22] without any admission control.*

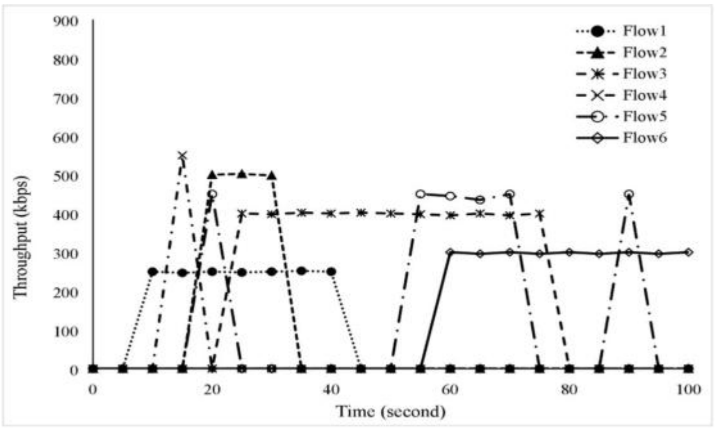


Fig. 4.7. *Throughput of flows using ABE [4] for admission control.*

network congestion by allowing short lived flows such as flow-4(550kbps) at t=25s and flow-5(450kbps) at t=55s despite of non-availability of enough ABW leading to performance degradation of existing flows. We observed poor individual flow's throughput performance and a lot of data packets loss in this scenario due to higher collisions.

Fig. 4.9 depicts the throughput of the flows when IBEM [18] approach is enabled. IBEM restricted its study to 2-dimensional plane only and ignored the movement of nodes under 3-dimensional space. Thus, it predicts the link failure of flow-5(450kbps) at t=15s and not allowed it to admit at this instance. IBEM admitted short lived flows (flow-4 at t=10s and flow-5 at t=85s) due to the inability to predict link failure at 3-dimension leading to data losses. Moreover, the admission of flow-3(400kbps) is delayed and started at t=25s due to false admission of flow-4(550kbps). IBEM performed better than earlier approaches, but still admission control is severe in this due to ignoring the effects of elevation and height while predicting the link failure.

Fig. 4.10 shows the received throughput when AABWM is enabled. Short lived flow-4 at (t=10s and t=25s) is rejected by admission control protocol due to the prediction of link failure. Similarly, flow-5 at (t=15s and t=85s) is also not permitted due to identification of link failure under 3-dimension. Rest all other flows (flow-1, 2, 3 and 6) are admitted and fitted well in the network gaining stable throughput. The proposed solution AABWM is more accurate in estimating the ABW. The approach AABWM also computes the collision probability analytically without stressing the network and can work well during the dynamic environment.

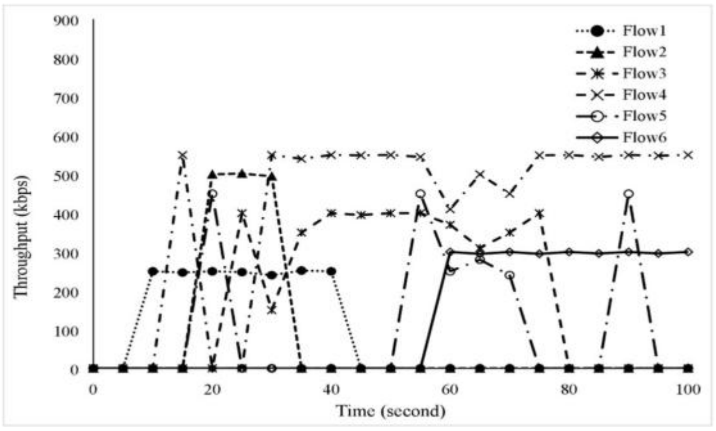


FIG. 4.8. *Throughput of flows using IAB [7] for admission control.*

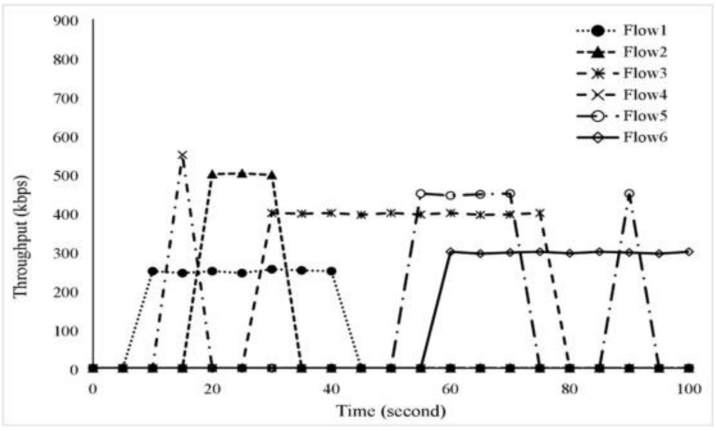


FIG. 4.9. *Throughput of flows using IBEM [18] for admission control.*

4.3. Packet Loss Ratio. The packet loss ratio is defined as the ratio of the number of data packets lost to the number of data packets sent by the sender. The number of lost data packets is calculated as the difference of the number of packets sent by the sender and the number of packets received at receiver node. Mathematically, this can be written as:

$$Packet\ Loss\ ratio = \frac{number\ of\ sent\ packets - number\ of\ received\ packets}{number\ of\ sent\ packets\ by\ the\ sender} \tag{4.1}$$

Fig. 4.11 depicts the packet loss ratio measured under the effect of mobility in the above experiment using NS2. We observe the packet loss ratio is negligible for all flows when AABWM is incorporated in admission control protocol. Whereas, other techniques resulting in an elevated packet loss ratio for flow 1 to 6.

4.4. Data Packets Delivery. The data packets delivery is calculated as the total number of data packets received at receiver node. The data packets delivery does not necessarily mean the useful data, rather it represents the total number of data packets delivered irrespective of the communication is completed. Fig. 4.12 represents the data packets delivery under the influence of node's mobility.

5. Conclusion. The current paper presents an analytical approach for estimating the available bandwidth in mobile adhoc network. The introduction of analytical models overcomes the disadvantages associated with active and passive approaches as discussed in the literature. The proposed analytical model is based on fixed point analysis based on renewal theory which helps in reducing the computations in comparison to Markov

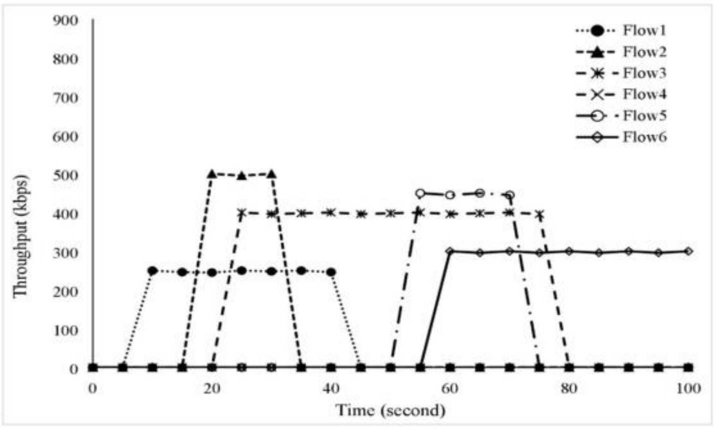


FIG. 4.10. *Throughput of flows using proposed AABWM approach for admission control.*

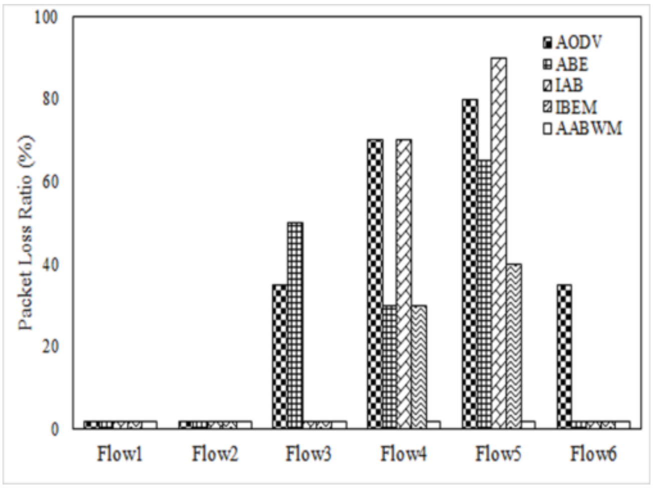


FIG. 4.11. *Packet loss ratio.*

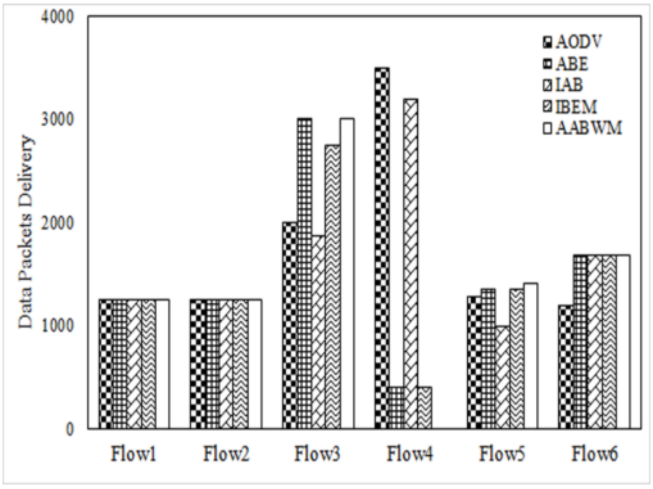


FIG. 4.12. *Data packet delivery.*

Chain based solutions. The link failure due to mobility of nodes leads to wastage of bandwidth resources and needs to be addressed while evaluating the available bandwidth. We incorporate the mobility issue and observed that link failure due to mobile nodes adversely affect the bandwidth estimation efforts. The proposed model is compared against the existing approaches in terms of bandwidth consumption and admission control solution using ns-2 simulations. The results indicate that in both static environment and dynamic environment the proposed model has most stable performance. The packet loss ratio is found to be least in comparison to other algorithms. The increased value of packet delivery ratio indicates the improvement in terms of delivery and throughput achieved by the proposed solution.

REFERENCES

[1] C. Chaudet, and I. Gurin Lassous, *BRuIT: Bandwidth Reservation under InTerference Influence*, European Wireless (EW 02), Florence, Italy, 2002, pp. 466–472.

[2] Y. Yang, and R. Kravets, *Contention-Aware Admission Control for Ad hoc Networks*, IEEE Transactions on Mobile Computing, 4 (2005), pp. 363–377.

[3] R. de. Renesse, V. Friderikos, and H. Aghvami, *Cross-layer cooperation for Accurate Admission Control Decision in Mobile Ad hoc Networks*, IET Communications, 1 (2007), pp. 577–586.

[4] C. Sarr, G. Chaudet, G. Chelius, and I. G. Lassous, *Bandwidth Estimation for IEEE 802.11-Based Ad hoc Networks*, IEEE Transactions on Mobile Computing, 7 (2008), pp. 1228–1241.

[5] C. Dovrolis, P. Ramanathan, and D. Moore, *What do Packet Dispersion Techniques Measure?*, Proc. IEEE INFOCOM, 2001, pp. 905–914.

[6] Y. Xiao, S. Chen, X. Li, and Y. Li, *A New Available Bandwidth Measurement Method Based on Self-Loading Periodic Streams*, in Wireless Communications, Networking and Mobile Computing, Shanghai, China, 2007, pp. 1904–1907.

[7] H. Zhao, E. Garcia-palacios, J. Wei, and Y. Xi, *Accurate Available Bandwidth Estimation in IEEE 802.11-based Ad Hoc Networks*, Computer Communications, 32 (2009), pp. 1050–1057.

[8] S. S. Chaudhari, and R. C. Biradar, *Collision Probability based Available Bandwidth Estimation in Mobile Ad Hoc Networks*, Applications of Digital Information and Web Technologies (ICADIWT), 2014, pp. 244–249.

[9] G. Bianchi, *Performance Analysis of the IEEE 802.11 Distributed Coordination Function*, IEEE Journal on Selected Area in Communication, 18 (2000), pp. 535–547.

[10] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, *Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement*, Proc. IEEE INFOCOM'02, 2 (2002), pp. 599–607.

[11] P. Chatzimisios, A. C. Boucouvalas, and V. Vitas, *Performance Analysis of the IEEE 802.11 MAC protocol for Wireless LANs*, International Journal of Communication Systems, 18 (2005), pp. 545–569.

[12] E. Ziouva, and T. Antonakopoulos, *CSMA/CA Performance under High Traffic Conditions: Throughput and Delay Analysis*, Computer Communications, 25 (2002), pp. 313–321.

[13] C. H. Foh, and J. W. Tantra, *Comments on IEEE 802.11 Saturation Throughput Analysis with Freezing of Backoff Counters*, IEEE Communication Letters, 9 (2005), pp. 130–132.

[14] Y. S. Liaw, A. Dadej, and A. Jayasuriya, *Performance Analysis of IEEE 802.11 DCF under Limited Load*, Proc. Asia-Pacific Conference on Communications, 2005, pp. 759–763.

[15] D. Malone, K. Duffy, and D. Leith, *Modeling the 802.11 Distributed Coordination Function in Nonsaturated Heterogeneous Conditions*, IEEE/ACM Transactions on Networking, 15 (2007), pp. 159–172.

[16] A. Kumar, E. Altman, D. Miorandi, and M. Goyal, *New Insights from a Fixed Point Analysis of Single Cell IEEE 802.11 WLANs*, Proc. IEEE INFOCOM, 2005, pp. 1550–1561.

[17] Q. Zhao, D.H.K. Tsang, and T. Sakurai, *A Simple and Approximate Model for Nonsaturated IEEE 802.11 DCF*, IEEE Transactions on Mobile Computing, 8 (2009), pp. 1539–1553.

[18] R. Belbachir, Z. M. Maaza, and A. Kies, *The Mobility Issue in Admission Controls and Available Bandwidth Measures in MANETs*, Wireless Personal Communications, 70 (2013), pp. 743–757.

[19] R. Belbachir, Z. M. Maaza, A. Kies, and M. Belhadri, *Collision's Issue: Towards a New Approach to Quantify and Predict the Bandwidth Losses*, Global Information Infrastructure Symposium, Trento, Italy, 2013, pp. 1–7.

[20] L. Chen, and W.B. Heinzelman, *QoS-Aware Routing based on Bandwidth Estimation for Mobile Ad Hoc Networks*, IEEE Journal on Selected Areas in Communications, 23 (2005), pp. 561–572.

[21] E. Kaplan, and C. Hegarty, *Understanding GPS: Principles and Applications*, Artech House, 2005.

[22] C. Perkins, E. Belding-Royer, and S. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*, RFC Editor, 2003.

Edited by: Anand Nayyar
Received: Mar 15, 2019
Accepted: Apr 5, 2019

DOI 10.12694/scpe.v20i2.1526
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 335–347. `http://www.scpe.org` © 2019 SCPE

PLACEMENT STRATEGY FOR INTERCOMMUNICATING TASKS OF AN ELASTIC REQUEST IN FOG-CLOUD ENVIRONMENT

R. SRIDHARAN[*] AND S. DOMNIC [†]

Abstract. Cloud computing hosts large number of modern day applications using the virtualization concept. However, end-to-end network latencies are detrimental to the performance of IoT (Internet of Things) applications like video surveillance and health monitoring. Although edge/fog computing alleviates the latency concerns to some extent, it is still not suitable for applications having intercommunicating tasks. Further, these applications can be elastic in nature and demand more tasks during their life-time. To address this gap, in this paper a network aware co-allocation strategy for the tasks of an individual applications is proposed. After modelling the problem using bin packing approach with additional constraints, the authors propose a novel heuristic IcAPER,(Inter-communication Aware Placement for Elastic Requests) algorithm. The proposed algorithm uses the network neighborhood machine for placement, once current resource is fully utilized by the application. Using *CloudsimPlus* simulator the performance IcAPER algorithm is compared with First Come First Serve (FCFS), Random and First Fit Decreasing (FFD) algorithms for the parameters (a) resource utilization (b) resource fragmentation and (c) Number of requests having intercommunicating tasks placed on to same PM. Extensive simulation results shows IcAPER maps 34% more tasks on to the same PM and also increase the resource utilization by 13% while decreasing the resource fragmentation by 37.8% when compared to other algorithms in our consideration.

Key words: Fog computing, Cloud computing, Elasticity, VM placement, Intercommunicating task, Network latency, Autoscaling

AMS subject classifications. 68M14, 68M20

1. Introduction.

Due to proliferation of Internet-of-Things (IoT), lot of smart devices utilize information from sensors attached to them and applications of IoT devices have become more personalized, automatized, and intelligent. These devices can adapt to their surroundings, and communicate with other devices to create new forms of smart, intelligent, and autonomous services. Generally smart devices have inadequate computation power, storage and bandwidth. Therefore, new forms of services are backed by cloud resources as cloud provides virtualized elastic resource at low cost using pay-per-use model in their huge computing and storage resources.

However applications, such as video surveillance, real time gaming, real time streaming and augmented reality are too sensitive towards reliability and latency. Since cloud data centers are located near the core network, the information transmitted from/to the end devices of these applications from/to cloud leads to unacceptable latency besides the problems like location-awareness and geo-distribution of IoT applications.

Fog computing paradigm ensures low-latency, high-bandwidth and location-awareness to the IoT applications.This paradigm pushes the elastic resources of cloud namely, computing and storage to the network edges, i.e. near to the data generation point. As and when resources of fog servers are exhausted, naturally computation and storage are pushed back to cloud resource. Combined resources of fog-cloud environment is referred as fog Cloud Datacenter (fogCDC) in this paper. Fog computing comprises of micro-clouds (fog nodes) that are placed at the edge of data originator.These micro-clouds have the mechanism for processing the data before going on to the network. Hence, fog computing structure is useful in big IoT data analytics [1]. High level overview of fog platforms and architecture is presented in [2] along with implementation of Virtual Machines migration in fog computing. Since fog computing cooperates with cloud computing for its resource demand, together with IoT edge devices, a three layered service delivery model as shown in Fig. 1.1, is adopted for fog computing. Both cloudlet [3,4] that was built before the emergence of fog computing, but inherently coincides with fog and ParaDrop [5] that uses gateway (WiFi access point or home set-top box) for fog implementation, essentially highlight the role of virtualization in a fog architecture. It is to be noted that cloudlet is nothing but a task that is mapped on to an auto-destroyable VM, once the task completes its execution. Hence task and VM are interchangeably used in this paper. Group of VMs launched in the distributed fog-cloud servers working together for a specific surveillance task is demonstrated in [6]. Currently majority of the IoT systems and software are built on top of open Web-based technologies [7]. Authors of [8], review the current efforts in realizing big IoT data analytics on fog networking. Therefore, we can easily conclude that majority of application

[*]National Institute of Technology Trichy, Tiruchirapalli- 620 015. Tamil Nadu, India. (`sridharter@gmail.com`).
[†]National Institute of Technology Trichy, Tiruchirapalli- 620 015. Tamil Nadu, India (`domnic@nitt.edu`).

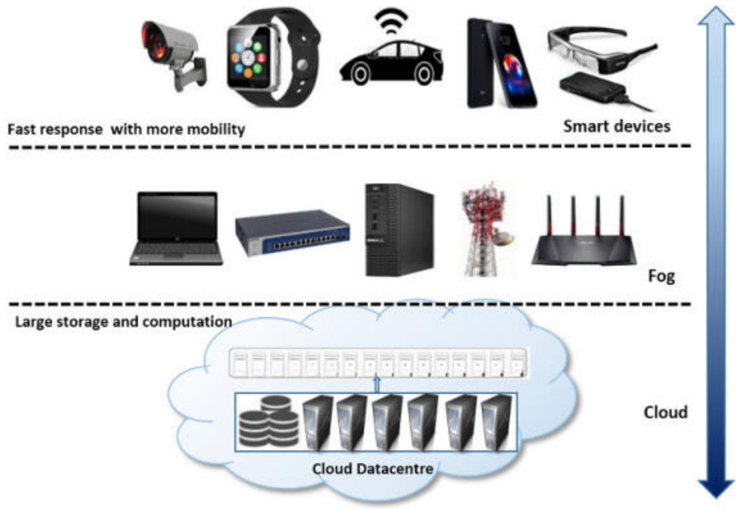

Fast response with more mobility
Smart devices
Fog
Large storage and computation
Cloud
Cloud Datacentre

FIG. 1.1. *Three layered architecture of Fog Computing*

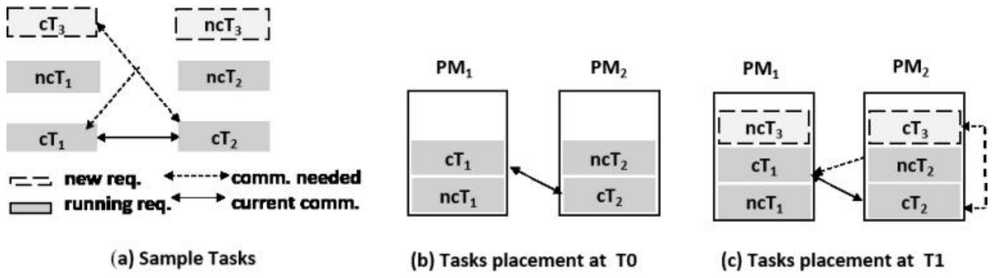


FIG. 1.2. *Placement of sample Tasks using Load balancing FCFS*

types can use VM based Fog-cloud environment for their processing.

Some of the VMs of these application need inter-communication. For example VMs hosting database task and hosting application task need inter communication in a three-tier web applications. Similarly VMs that hosts object tracker task in a video surveillance system that uses multiple cameras covering multiple areas, need inter-task communication. As these applications request sporadic resources, they depend heavily on elastic capability of the fog-cloud paradigm. Elasticity can be classified as two types: horizontal and vertical. In horizontal elasticity, dynamically the number of VMs are adjusted, whereas the resources (CPU, storage and Memory) of a VM are dynamically adjusted in vertical elasticity [9]. In such a scenario mapping of tasks to the suitable Physical Machine (PM)[1], called Virtual Machine Placement (VMP), is a challenging cloud problem.

To summarize, a cloud request can be classified as those having inter-communicating VMs and those not having inter-communicating VMs, which can be further classified as needing elasticity and not. Authors, confines this work to find suitable placement for application tasks that need inter-communication and horizontal elasticity as well, hereafter called as simply elastic request. Other requests are referred as non-elastic requests, which may or may not have inter-communicating VMs. Applications having inter-communicating VMs, gives better performance, when they are placed together on to the same PM [10] as there will not be any network latency between these VMs. But general VM placement schemes fail to recognize this and place the VMs on to PMs in an arbitrary manner.

Consider an example of five requests that are currently(time t_0) executing in a datacenter. Let the first (ncT_1) and third $(ncT_2\ )$ requests, having one tasks each, does not require inter-communication. Let the second

[1]PM denotes fog server as well as the servers of cloud Datacenter

request having two task *(cT_1 and cT_2)*needs inter-communication. Assume that the fourth *(ncT_3* and fifth *(cT_3)* are newly (time t_1) arrived requests. While the forth request is non-intercommunicating type, the fifth request is an additional task of second request that needs inter communication with *(cT_1 and cT_2).* Further, these tasks follow the communication pattern represented by arrows, as shown in Fig. 1.2(a). Assuming that we have two PMs to place these tasks. Using the widely used VM placement algorithm First Come First Serve (FCFS) along with load balancing, placements of tasks at t_0 resembles to that of Fig. 1.2(b). Placement of tasks at t_1 is shown Fig. 1.2(c). If all the tasks are placed onto single PM itself, the other PM could be switched off to reduce the overall power consumption. Else placing all the inter-communicating tasks *(cT_1, cT_2 and cT_3)* on to the same physical machine reduces the power consumption used for communication, as network latency among these tasks is nil. Since fog computing uses the cloud resources once their resources are fully utilized, placements for inter-communication task of a single request can happen both at fog resources and cloud resources as well. Particularly, when elastic requirements of this request have to be addressed, placements using both fog and cloud resources in unavoidable.

Motivated by this observation, in this paper, after formulating Inter-communication Aware Task Placement of Elastic Request (IcAPER) problem, we describe in detail, the proposed IcAPER algorithm that improves the resource management of fogCDC. Performance of individual fog applications is also bound to improve using this algorithm, as it implements co-allocation for tasks of these applications. Co-allocation is also adhered in cloud resources once fog server is fully utilized. This is achieved by maintaining the identity of tasks with reference to its application across fogCDC. In summary, contributions of this paper are as follows:

- A mathematical modeling of Inter-communication Aware Task Placement of an Elastic Request (IcAPER) problem. This model considers requests of individual IoT application having one or more tasks along with its communication pattern and elastic needs during their lifetime for placement instead of individual tasks, a widely followed practice is presented.
- Guided by IcAPER model, IcAPER algorithm that minimize the network latency and improves the datacenter efficiency is proposed.
- Effectiveness of the proposed algorithm is demonstrated in a simulation environment via (a) reduced VM migrations, (b) resource fragmentation, (c) improved resource utilization and (d) Number of requests having intercommunicating tasks placed on to same PM.

Rest of this paper is organized as follows. *Section 2* reviews the available literature related to the problem. While Section 3 describes the problem formulation and its modeling, Section 4 describes details the proposed IcAPER algorithm along with illustration of IcAPER placement for the sample task described in Fig. 1.2(a). Section 5 presents the implementation and analysis of IcAPER and Section 6 concludes this work.

2. Related Works.

Fog computing, relatively a new distributed computing paradigm, extends the cloud service to the edge of the network [11]. Fog device, any element in the network capable of hosting application modules, include resource-rich servers and resource-poor devices such as gateways, mobile, smart vehicles, smart cameras etc. As both fog resource and cloud resources can host VMs, this work reviews the general VM placement, a multi-parameter optimization problem in this section. Simulation environments [12,13,14] have been used in evaluating the solutions for this problem .

Multi-dimensional bin-packing problem is used to model the VMP, where PMs are considered as bins and VMs are considered as items, with the goal of finding minimum number of bins to hold all the items. Since bin-packing is a NP-hard problem, heuristic based approximation algorithms are proposed for VMP. First Fit (FF) algorithm [15], a greedy method is the most popular algorithm used for solving bin-packing problem. FF keeps filling the current bin until the PM could not hold the new VM. Even though this approach is time-effective, it usually occupies more PM, than that of an optimal solution. Modified version of FF is proposed by Panigrahy et al. [16] where VMs are sorted in decreasing order before applying FF on the sorted items. This is called First Fit Decreasing (FFD) algorithm which ensures that large items are placed first.

Authors of [17] proposed a dynamic mapping between new VMs and the PMs using probability theory, and demonstrate that the energy consumption of DC is minimal. While authors of [18] proposed the VMP solution for a balanced PMs interms of load in a DC, others [19, 20] tried to minimize the power consumption of DC by employing consolidation techniques to ensure that active PMs are always minimal by maximizing their utilization. Mahfuzur et al., [21], proposed static initial VM placement where, VMs are co-placed based on their resource requirement compatibility. In this approach, PMs and VMs are differentiated as resource critical and non-critical based on their historical usage. But they have not considered the elastic requirements of an application hosted by these VMs. To minimize the power consumption, a decentralized decision based VMP is proposed by Feller et al., [22], in which, a peer-peer (P2P) communication model is proposed among PMs. Deploying this communication model, periodic consolidations of VMs are accomplished by using cost-aware Ant Colony Optimization (ACO) techniques. Cui et al., [23] brought out the importance of policy and network awareness before considering VM migrations. The authors demonstrated an efficient VM management scheme which reduces the communication cost while considering PMs to migrate the VMs.

Further, some contributions have explored two-stage approach as well for VMP problem. Beloglazove et al., [24] proposed two stages for the VMP: (i) initial placement of VMs and (ii) optimizing the current placement. In this, using CPU utilization of the VMs, a Modified Best Fit Decreasing (MBFD) algorithm is used for initial placement and the second phase is triggered whenever an overloaded or an under-loaded PM is detected as a result of a well-defined CPU threshold. Without considering elasticity, Zheng et al., [25] subdivided the VMP as two sub-problems. A Best Fit (BF) algorithm, used for incremental placements acts as first step followed by another step in which, periodical trigger of VM consolidation focusing on reducing power consumption and minimization of resource wastage.

An interesting two-phase online VMP is proposed by Shi et al., [26] where PMs and VM requests are represented as vectors. During the first phase, based on cosine similarity of the vectors, PM types are assigned to VM requests and based on resource request, VMs are categorized for each PM type. Using utilization thresholds of PMs, reconfiguration of VMs is triggered in the second phase. Like [24] here also authors have not taken into account elasticity at all. Another two phase approached for VMP proposed by Fabio et al., [27] considers complex IaaS environment including elasticity but did not consider the intercommunication patterns while placing the additional VM of an elastic requests. Mishra et al., [28] proposed algorithm to map the task onto VM and the VMs to PM. This algorithm minimizes the makespan and task rejection rate while reducing the energy consumption by reducing the active number PMs. The reviewed works, except [21,27,10,23] solve VMP problem as simple placement strategy using the current PM resource utilization for new VM placement, consolidation and energy consumption. Authors [21,27] include the elasticity in their works but not intercommunication whereas, authors of [10, 23] include network awareness but not elasticity.

As explained in Section 1, tasks placement accomplished after considering both the network awareness and intercommunicating needs of the tasks, together with the knowledge of prior placement of tasks from this application is bound to increase the application performance. Hence, this work propose to use prior tasks placement knowledge of an intercommunicating elastic application, for the placement of its additional tasks, while ensuring reduced overhead of fogCDC and also increased application performance.

3. Problem Statement and Proposed Model. This section explain the problem that is addressed through this work and its mathematical programming model followed tasks placements using IcAPER for the sample tasks described in Section 1.

3.1. Problem statement. The focus of this work is to improve task placement of an elastic IoT application requests for overall betterment of fogCDC management. By and large, and also by design, only volume of data generated by an IoT application can change over period of time and hence need more processing power to arrive timely decisions, which can be addressed by elasticity of fogCDC. These tasks requests can be classified into three types, namely (a) new VM requests, (b) migrating VMs requests and (c) incremental VM requests. Requests wanting elasticity service are expected to give the minimum number of VMs (*VMmin*) that needs to be running permanently during the lifetime of those applications. They are also expected to give maximum number of VMs (*VMmax*) that can be allocated at any given point of time to cater to the application fleets. In effect, the number of VMs dedicated to these applications at any time shall be between *VMmin* and *VMmax.* This requirement is in consistent with popular cloud service providers like Google and Amazon. In addition users are also expected specify need of intercommunication to their tasks. Since, application logic remains same for the life-time of an IoT applications, using the historical data, the service providers can also easily identify inter-communication needs of these tasks.

Therefore, we can assign application driven unique identifier for these tasks. Using this identifier we can easily influence the task scheduler for a desired resource mapping for these tasks. Efficiently placed *VMmin* and incremental VMs of these requests can give various benefits to both user and fog-cloud service provider. These incremental VM requests, when solved as a fixed placement problem result in cascading benefits. Firstly, inter VM communication latency is reduced as network is not involved, which is bound to increase the application performance. Secondly, placing *VMmin* VMs together reduces the potential migrations during cloud server consolidation. This reduced migration and network latency leads to minimal power consumption in fogCDC. Thirdly, as individual PMs are balanced with the placement of VMs, serving both communication depend applications and applications made up of non-communicating tasks, SLA violations are bound to be minimal with respect to inter-communicating tasks.

3.2. Proposed Model. The proposed work consider the communication pattern of the tasks of a request and also the elastic needs of requests as additional constraints for the general VMP problem (GVMP), to formulate the IcAPER problem. IcAPER can be viewed as a bin packing problem where PMs are treated as bins and VMs are treated as items. This is an optimization problem in which number of PMs (bins) used to place the VMs (items) are to be minimized. This work segregate the requests into three types: (a) requests for placing group of tasks having inter-communication along with elastic needs (r^e_{comu}), (b)without elasticity but needs inter-communication (r_{comu}) and (c)all other requests (r).

To formalize the problem we make the following assumptions:

- Let r_1, r_2, r_αbe the requests defined by a four-tuple r_k= $\{u, VM_{type}, VM_{min}, VM_{max}\}, 1 \leq k \leq \alpha$ where VM_{min} and VM_{max} are number of VMs, $VM_{type} = \{micro, small, medium, large, xlarge\}$ and u= $\{r^e_{comu}, r_{comu}, r\}$.
- $p = \{p_1, p_2, p_n\}$ and $v = \{v_1, v_2, v_m\}$ as the identifier of PMs and VMs respectively
- Both PMs and VMs are constrained with D-dimensional resource vectors R (number of CPUs, memory size, storage space, bandwidth speed etc.).
- n represents total number of PMs, m represents total number of requests and N represents VM_{min}, Then from the above assumptions we have,

$$r_k = \sum_{i=1}^{VM_{min}} v_i \tag{3.1}$$

Hence,

$$m = \sum_{k=1}^{\alpha} r_k \tag{3.2}$$

Let us introduce a function ϕ to indicate the placement schema $\phi(r_k) = p$ if VMs of r_k th request are placed on the pth PM. The objective of GVMP is, given a request sequence S, and PMs $p, 1 \leq p \leq n$ with capacity C, find a placement function ϕ such that

$$minimize \sum_{i=1}^{n} p_i \tag{3.3}$$

subject to the following two constraints.

- **Resource constraints:** During the placement of VMs belonging to r_k requests, the PMs identified to host these VMs are subject to primary resource constraints:

$$\forall 1 \leq p \leq n, 1 \leq d \leq D \quad \sum_{r_k=1}^{m} \sum_{v=1}^{r_k} X_{vp} * R^d_v \leq C \tag{3.4}$$

where $R^d_v(1 \leq d \leq D)$ is the resource demand of v^{th} VM and

$$X_{vp} = \begin{cases} 1, & \\ 0, & \text{otherwise} \end{cases}$$

- **Placement constraints:** Each VM of user request r has to be placed to run on a single PM

$$\forall 1 \leq p \leq n, \quad \sum_{r_k=1}^{m} \sum_{v=1}^{r_k} X_{vp} = 1 \tag{3.5}$$

Each item has resource vector $R_i = \{R_i^1, R_i^2, .R_i^d\}$ and each dimension $l \in [1,..,d]$, denotes resource demand size R_i^l which is normalized 0..1 and capacity of each PM is also normalized to 1, then we have

$$X_{vp} \in \{0,1\} \quad \text{and} \quad n \in \{0,1\} \tag{3.6}$$

Next IcAPER is formulated by adding three more constraints, as described below.

- ***Same-PM* constraint:** In any request having inter-communication tasks (r^e_{comu}), let VMs v_1 and v_2 have co-allocation requirement and are required to be co-allocated onto same PM $p \in [1n]$ then we can combine VM v_1 and v_2 to one group and form a new VM v_{new}. For each resource dimension d, the size of the group VM is denoted by sum of VM v_1 and v_2.

$$i.e.R^d_{v_{new}} = \quad R^d_{v_1} + R^d_{v_2} \tag{3.7}$$

Hence the following constraint is added to GVMP:

$$\begin{aligned} \forall p \in [1n], \{\forall v_1, v_2 \in [1..VM_{min}]\} \in r_k \quad \& \\ r_k \in \{r^e_{comu}, r_{comu}\}, \quad X_{v_1 p} = X_{v_2 p} \end{aligned} \tag{3.8}$$

- ***Fixed-PM* constraint:** Let us define matrix A_{ij}, where $i,j \in \{1m\}$, to denote the request relation among individual tasks. As explained in previous section, we also set the individual element of this matrix with r_k, the application driven request identifier, to which the current tasks belongs.This identifier is uniquely identifiable across fogCDC. Assume that $r_k^{'}$ denotes the elastic requirements of request r_k, then for each tasks of an inter-communication needed elastic task group, we add the following fixed-PM constraint:

$$\forall r_k \in r^e_{comu}, \forall 1 \leq k \leq \alpha, \quad x_{vp} = r_k^{'} = r_k \tag{3.9}$$

Comment: All the above explained constraints are suitable for cloud resources only. This is because, in normal circumstances, there will be only a single resource rich fog server will be available per area to which all IoT applications send the data. Fog architecture depends solely on cloud resource for computation-heavy tasks. Hence the objective Eq. 3.3 with constraints of *cf. Eqs. 3.4, 3.5, and 3.8* are to be solved for an effective initial VMP of an individual request while the objective *cf. Eq. 3.3* with constraints of Eqs. 3.4 and 3.5 and Eqs. 3.4 and 3.9 to be satisfied during the additional VM requests of an intercommunicating elastic request.

Claim: IcAPER is NP-Complete.

Proof: As per constraint Eq. 3.8, resource demand vectors of all the tasks specified by VMmin r^e_{comu} and r_{comu} are combined to form a new VM having higher resource demand vector. Characteristics of VM, an equivalent to item in a bin-packing problem are not altered by adhering to this constraint. As stated in Eq. 3.6 we can equate the normalized IcAPER problem to that of normalized GVMP by dividing VM sizes with the corresponding resource capacities of PMs. Hence similar to GVMP, IcAPER is also NP-Complete.

Since minimization problem of IcAPER is NP-Complete, we cannot get the optimal placement solutions in polynomial time unless P=NP [29]. Hence we propose a algorithmic based solution in this paper.

4. Proposed IcAPER Algorithm. In this section, we present a VMP scheme guided by IcAPER problem modelling based on request type r^e_{comu}, r_{comu} and r rather than VM_{type} for optimal operations of fogDC. This scheme recognizes inter-communication needs of the requests for tasks placements. Initial placement focuses on placing the tasks of individual requests efficiently so that future migrations are minimal. This also ensures that the network overheads and overheads incurred for identifying suitable VMs and PMs that are candidates for these migrations, are also reduced. Subsequently, placement of elastic needs of requests having inter-communication are confined to the individual PM resulting in better application performance due to nil network latency. To develop such an algorithm, we make some simplifying assumptions as follows:

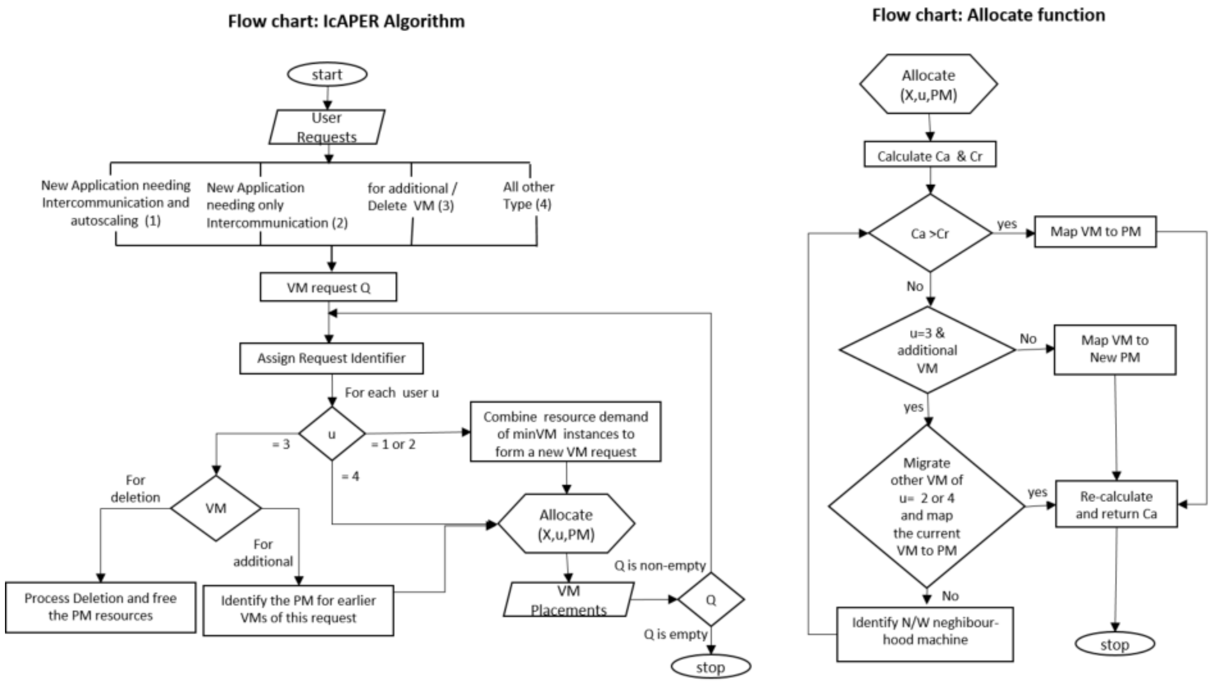

Flow chart: IcAPER Algorithm
start
User Requests
New Application needing Intercommunication and autoscaling (1)
New Application needing only Intercommunication (2)
for additional / Delete VM (3)
All other Type (4)
VM request Q
Assign Request Identifier
For each user u
u
= 3
= 1 or 2
= 4
Combine resource demand of minVM instances to form a new VM request
For deletion
VM
For additional
Allocate (X,u,PM)
Process Deletion and free the PM resources
Identify the PM for earlier VMs of this request
VM Placements
Q is non-empty
Q
Q is empty
stop
Flow chart: Allocate function
Allocate (X,u,PM)
Calculate Ca & Cr
Ca >Cr
yes
Map VM to PM
No
u=3 & additional VM
No
Map VM to New PM
yes
Migrate other VM of u= 2 or 4 and map the current VM to PM
yes
Re-calculate and return Ca
No
Identify N/W neghibour-hood machine
stop

FIG. 4.1. *Flow chart of the proposed Algorithm*

- fogCDC has enough capacity to serve all the requests.
- VM request can be initiated by user or cloud service. In case of elasticity, cloud service monitor shall initiate the event associated with increase and or decrease of VMs.
- Without loss of generality, we assume that by minimizing cross-traffics, overall IoT application performance will be improved.

In this paper, VM placement patterns are formulated in two steps. During the first step initial VMP is achieved subject to constraints Eqs. 3.4, 3.5, 3.8 and Eq. 3.9. Fixed placement pattern that includes network neighborhood machines, is adopted for the additional VMs of request type r^e_{comu} as the final step. We symbolize C_t for total (original) capacity and C_u for current utilization of active PM which is in consideration for placement, while resource demand of VM that needs to be placed, is defined by C_r. Note that all these parameters are defined in terms of resource vectors (number of CPUs, memory size, storage space, bandwidth speed etc.).

The Inter-communication Aware placement for tasks of an Elastic Request (IcAPER) algorithm, after extracting the VM details from the individual request, assemble the VMs according to various constraints listed in Section 3, and sends them to *Allocate* function along with candidate PM. *Allocate* function performs the actual allocation of VMs to given PM. Upon non-availability of enough resources in the PM, it initiates new PM and places the VMs. Flow chat of IcAPER algorithm and Allocate sub-function is presented in Fig. 4.1(a). The detailed description of the same follows.

4.1. IcAPER Algorithm Description. As a first step IcAPER algorithm collects individual requests, both from cloud service and users onto a batch B. Additional VM of elastic requests emanating from cloud services are also queued to the same batch. Note that each request is defined by four-tuple $\{u, VM_{type}, VM_{min}, VM_{max}\}$. Components of this four-tuple are already explained in Section 3. However if the request is emanated from cloud service, then combination of information indicating whether the request is to scale-up or scale-down the VM, and a unique request identifier through which the four-tuples can be extracted, defines the request.

At fixed time interval, the m requests are differentiated as fresh and additional request. After extracting VM details from VM_{type} of each request in m, they are segregated as r^e_{comu},r_{comu} and r. In adherence to

Algorithm 10 IcAPER Algorithm

```
Require : request, active number of PM $n$
Result : tasks placement                \* Refer Sect. 3 for description of symbols *\
1 fetch user requests $m$
2 while $m$ != NULL do
3     $r_k \leftarrow m.k$
4     for k = 1 to $\alpha$
5         $C_r = X.R^d$
6         if $r_k$ for deletion
7             process scale-down of VM
8         else if $r_k$ for addition
9             $Allocate\ (X, r_k, p_n^{'})$              \* adherence to fixed-PM constraint *\
10        else if $u \in r^e_{comu}$ or $u \in r_{comu}$
11            $X \leftarrow VM_{min}.R^d_v(VM_{type})$        \* adherence to same-PM constraint*\
12            $Cu = Allocate(X, u, P_n)$
13        else if $u \in r$
14            $X \leftarrow N.R^d_v(VM_{type})$
15            $Cu = Allocate(X, u, P_n)$
16    end for
17        $C_a \leftarrow (C_t - C_u)$
18 end while
```

same-PM constraint of the proposed IcAPER, the demand resource vectors of all the tasks, given by N, of new r^e_{comu} request and new r_{comu} are combined individually to form a new task and considered as single placement [line 10]. If the request is for scale-up or scale-down of VM from an existing r^e_{comu} type requests, then PM details are extracted from the request identifier and processed accordingly [lines 5-9]so as to adhere to fixed-Pm constraints. After the formulation of VMs, sub-function *Allocate* is called with the candidate PM [lines 12,15]. Before addressing the next request available capacity of the PM is re-calculated. [line 17]. Description of the *Allocate* function is presented next.

Algorithm 11 Allocate Sub-function

```
Input : $X$ (VM to be placed), request $r_k$, $p_n$ (PM to be used for mapping)
return : utilization of PM              \* Refer Sect. 3 for description of symbols *\
1 extract $u$ from $r_k$
2 $C_a \leftarrow (C_t - C_u)$ and $C_r = X.R^d$
3 if $u \in r^e_{commu}$ and $X$ is an additional VM
4     if $C_a > C_r$) map $X$ onto $p_n$
5     else if ($p_n \exists X^{'}.R^d \in r_{comm} \geq C_r$)
6         migrate $X^{'}$ from $p_n$ using FirstFit
7         map $X$ onto $p_n$
8     else map $X$ onto network neighborhood machine by repeating step 4-8
9     else if ($C_a > C_r$) map $X$ onto $p_n$
10    else map $X$ onto new PM
11 return $C_u$
```

4.1.1. Allocate function description. For Allocate function, request type along with the VM to be placed and candidate PM are given as input. After calculating available C_a and extracting resource demand vector C_r of current VM in consideration for placement [line 3] function proceeds to map the VM onto candidate PM given as the input. If the request is for an additional VM of r^e_{comu}, on availability of enough capacity, the request C_r is directly mapped on to the PM [line 5] else, migrate suitable VM or combination of VMs belonging to

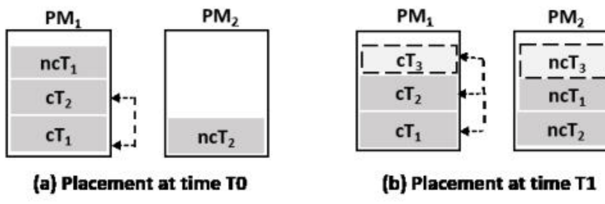


Fig. 4.2. *Placement of sample Tasks using IcAPER*

Table 5.1
Host configuration used in simulation

Core	RAM	VMM	Storage	MIPS
16	64 GB	Xen	1 TB	100,000

r from that PM using First Fit algorithm and map the additional VM [lines 6-8]. Upon unsuccessful placement suitable PM from the network neighborhood is identified and repeats steps 6-8 until successful placement happens. If the request is new one (r^e_{comu},r^e_{comu} and r), then on availability of enough resources, we directly map the VM onto the given PM, failing which new PM is initiated and mapping of VM is done[lines 10-11]. Finally the current utilization of the PM used for the latest placement process is returned to the Algorithm 1.

4.2. Cost Analysis of IcAPER Algorithm. Number of VM requests m in the batch queue will be a modest constant. For example, if scan interval of a batch is 5 minutes and on an average 50 VM requests get queued up, then Algorithm 1 needs to scan [lines 3-16] these 50 requests only and hence the cost is *O(m).* Cost of sorting the segregated batch B is *O(m log m)* [line 17, 18]. Since *eList* and *neList* lists are of size *O(m/2)*, the following loop[lines 19 - 28] will also execute *O(m)* times, implying that *Allocate*, will also be called *O(m)* times. The inner loop having break [line 26] will execute once sufficient non-elastic VMs are placed matching to that of an elastic VM. As the number of VMs in a PM is bounded by a constant, the cost of inner loop is *O(1).* Last loop [line 29-31] that places the leftover VMs also has a cost of *O(m).* Let n be the number of candidate PM in *Allocate* sub-function, then packing the VMs onto PM, results in cost of *O(n).* Hence *O(m log m) + O(n)* is the total cost of IcAPER, which is asymptotic to FFD and FCFS algorithms.

4.3. IcAPER for sample VMs. Let us look at the changes that the balanced FCFS VM placement scheme described in c*f. Sect. 1*, for the sample VMs as depicted in Fig. 1.2 (a) goes through during initial VMP (T_0) and subsequently T_1, according to IcAPER. IcAPER scheme for these VMs is depicted in Fig. 4.2. As per IcAPER, at time T_0 the initial requests of elastic user are placed together on to the same PM as shown in Fig. 4.2(a), whereas at time T_1, the VM placements resembles to Fig. 4.2(b), in which ncT_1 is migrated to second PM so that additional request cT_2 can be placed on to the first PM itself. This clearly reduces the cross PM communications and hence the network traffic resulting in conservation of energy. Here the trade off is between the cost incurred to migrate ncT_1 and expected application performance. We will explain the experimental setup and the analysis of results in the subsequent sections.

5. Implementation. Efficiency of any VMP algorithm, needs to be evaluated in a large-scale cloud platform. However, it is difficult to conduct the repeatable VMP experiments in a real production cloud platform. Hence, the proposed IcAPER algorithm is simulated using *CloudSimPlus* [30] simulator. The algorithms were implemented as an extension to SimpleVmAllocationPolicy, which determines how VMs are assigned to the host.

Since our focus in this work is to improve VMP that is common to both fog and cloud resources, we carried out the experiments using *CloudSimPlus* simulator having Xen as the virtual machine monitor (hypervisor). The proposed, Inter-communication Aware Elastic Request Placement (IcAPER) algorithm, written in Java, is tested on a Dell workstation with Intel Xeon 3.30Ghz and 16Gb memory, having x64 architecture. The simulation is performed over a datacenter made up of twenty homogeneous physical machines whose configuration is presented in Table 5.1. VM configuration of popular service provider Amazon EC2 [31] is adopted and the same is shown in Table 5.2.

Table 5.2
Virtual Machine configuration used in simulation

resource	Micro	Small	Medium	Large	Xlarge
vCPU	1	1	2	2	4
RAM (GB)	1	2	4	8	16

Table 5.3
Characteristics of requests in a batch

Characteristic	Number
Number of Request	40
Requests type r^e_{comu}	12
Requests type r_{comu}	10
Requests type r	18
New VM requests by type r^e_{comu}	22
New VM requests by type r_{comu}	32
New VM requests by type r	45
Requests for additional VM	7
Requests for delete VM	3

Since we are addressing the application elasticity along with inter-communication of its tasks in this work, for each experiment, we created three bunches of randomly generated requests in the ratio of 2:1:1 for time slot T_1, T_2 and T_3.The total resource requirement of these requests is approximately equal to the capacity of our datacenter. Note that each request is represented by a four-tuple as explained in Section 3. Characteristic of requests for single experiment is presented in Table 5.3. Column 8 is valid only for the request belonging to time T_1 onwards whereas, column 9 is valid for the request belonging to time T_2 onwards only. As explained in Section 1, each VM is attached with a cloudlet whose input file size is 1000 bytes. We have compared the proposed IcAPER algorithm with FCFS, Random and FFD algorithms for an experiment. To avoid transient anomalies we conducted ten such experiments and collected the statistics for evaluation parameters: (a) resource utilization, (b) resource fragmentation, (c) reduced VM migrations and (d) Number of requests having intercommunicating tasks placed on to same PM.

The results presented in Figs. 5.1 – 5.4 demonstrate that IcAPER algorithm is preferred one when compared with other algorithms which are considered for the comparison. From Fig. 5.1, we can find out that resource utilization by IcAPER increase when more requests are served than other algorithms. When individual PMs may have enough resources in one dimension (CPU, memory etc.) but not in all dimensions and hence requests cannot be placed on to it, resulting in resource fragmentation. The increase in resource fragmentation, over a period of time, implies scope for improvement in fogCDC management. As shown in Fig. 5.2 IcAPER algorithm perform best when compared with other algorithms. This work proposes to host all the tasks of r^e_{comu} and r_{comu} type requests, along with its additional(elastic) tasks on the same PM. When compared all the algorithm for this parameter, as depicted in Fig. 5.3, IcAPER outperforms all other algorithms.

Next evaluation parameter is potential migration needed in all algorithms that are in our consideration to ensure nil network latency. It is to be noted that as by design, IcAPER has to place maximum possible tasks on the same PM at T_1. However, from T_2 onwards all algorithm makes placement decision based on the availability of resource on PM and individual algorithmic constraints. For example, IcAPER forces the migration of tasks belongs to r_{comu} and r type request, so that co-allocation constraint for elastic needs of r^e_{comu} type requests can be adhered. As shown in Figure 5.4 IcAPER consistently outperforms other algorithms over multiple time slot. It is also to be noted that the application performance and cost of migration are directly proportional to the number of additional tasks requested by elastic request when fogCDC is deployed with IcAPER algorithm.

6. Conclusions. In this paper, after formulating Inter-communication Aware Elastic Request Placement (IcAPER) problem, pertaining to a fog-cloud environment, authors propose an efficient IcAPER algorithm to improve the resource utilization of fog-cloud datacenter. IcAPER considers individual request encompassing one or more VMs as a whole for placement along, with the life-time of the request rather than individual VMs. This proposed algorithm places all the tasks of requests needing intercommunication and elasticity, on to same PM

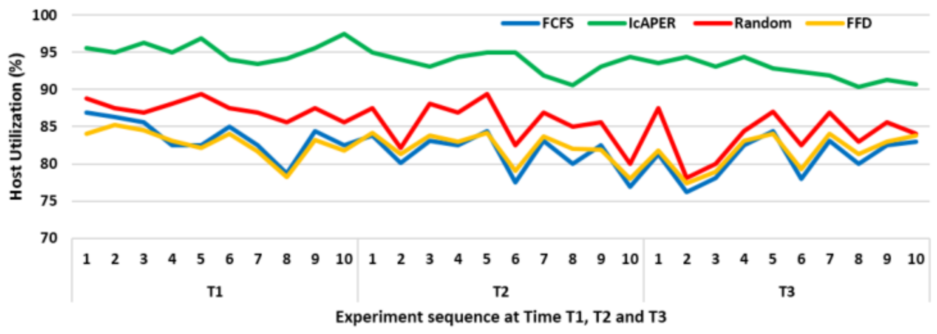


FIG. 5.1. *Physical Machine Utilization*

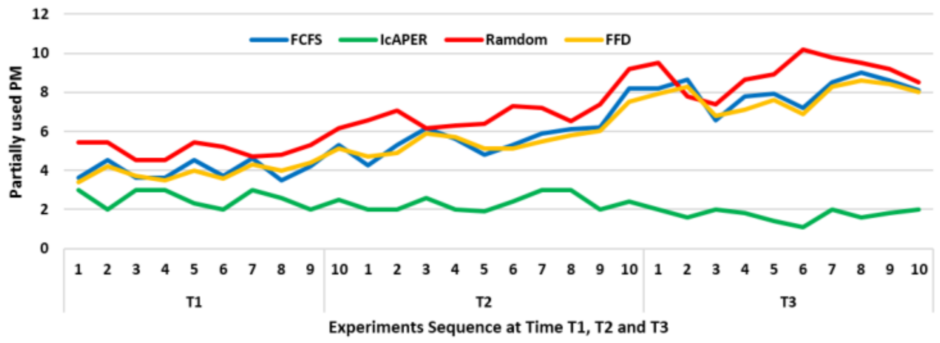


FIG. 5.2. *Physical Machine Fragmentation*

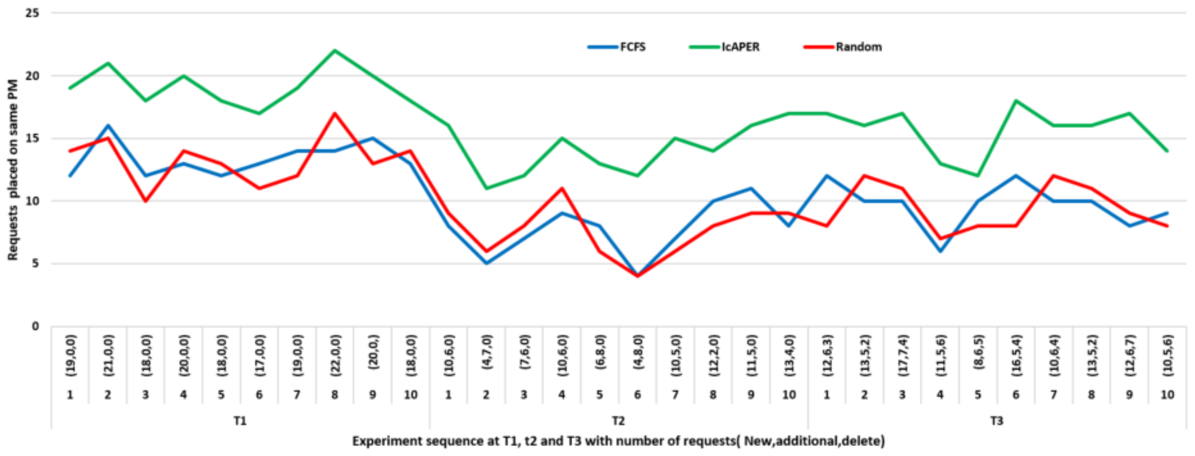


FIG. 5.3. *Number requests needing intercommunicating tasks mapped to same PM*

and/or on to network neighborhood PMs by implementing same-PM, fixed-PM and balanced-PM constraints for task placements. These constraints are adhered by maintaining the identity of tasks with reference to its application across fogCDC resulting in improved performance of fog application.

Performance of IcAPER is compared with FFD, FCFS and Random, well-known placement algorithms for metrics resource utilization, resource fragmentation, migrations needed to maintain minimal network latency and number of requests having intercommunicating tasks placed on to same PM for evaluation. Based on average taken out of ten experiments, the simulation results shows proposed technique is attractive compared to other algorithms that are in authors consideration. Studying the performance impact of IcAPER algorithm

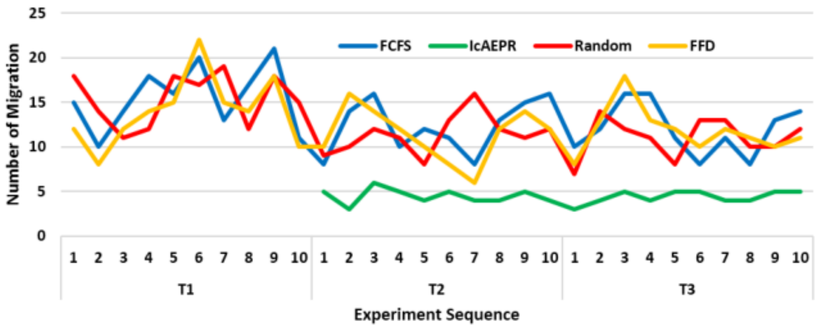


Fig. 5.4. *Potential Migration needed to minimize the latency*

to a real time video surveillance application using distributed camera will be of future interest. Pan-tilt-zoom (PTZ) parameters of multiple cameras requires low-latency communication between the cameras for which we can use the fog resources. Whereas, for processing the captured video frames and for long-term processing (analysis) of control strategy to obtain the optimal PTZ parameter cloud resource can be used.

REFERENCES

[1] S.Sing, A.Nayyar, R. Kumar, and A.Sharma, *Fog computing: from architecture to edge computing and big data processing* , in The Journal of Supercomputing, 2018, PP. 1-36
[2] S. Yi, Z. Hao, Z. Qin, and Q. Li, *Fog computing: Platform and applications*, in Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb). IEEE, 2015, pp. 7378
[3] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, *Cloudlets: at the leading edge of mobile-cloud convergence*, in IEEE International Conference on Mobile Computing, Applications and Services (MobiCASE), 2014.
[4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies , *The case for vm-based cloudlets in mobile computing* , in Pervasive Computing, 2009.
[5] D. F. Willis, A. Dasgupta, and S. Banerjee, *Paradrop: a multi-tenant platform for dynamically installed third party services on home gateways*, in ACM SIGCOMM workshop on Distributed cloud computing, 2014.
[6] J. Wang, J. Pan, and Flavio Esposito,*Elastic Urban Video Surveillance System Using Edge Computing*, In Proceedings of SmartIoT17, 2017. https://doi.org/10.1145/3132479.3132490.
[7] N. Mäkitalo, F. Nocera, M. Mongiello, and S. Bistarelli, *Architecting the Web of Things for the fog computing era*, doi: 10.1049/iet-sen.2017.0350
[8] M. Anawar, S. Wang, M. Zia, A. Jadoon, U. Akram,and S. Raza, *Fog Computing: An Overview of Big IoT Data Analytics*, 2018. ID 7157192 https://doi.org/10.1155/2018/7157192.
[9] . Wang, H. Chen, and X. Chen, *An availability-aware virtual machine placement approach for dynamic scaling of cloud applications*, in Proc. IEEE Ninth Inter. Conf. Ubiquitous Intelligence & Computing and Autonomic & Trusted Computing (UIC/ATC), pp.509-516, 2012.
[10] X. Meng, V. Pappas, L. Zhang, *Improving the scalability of data center networks with traffic-aware virtual machine placement*, in Proc IEEE INFOCOM2010, pp. 1-9, 2010.
[11] F. Bonomi, R. Milito, P. Natarajan, J. Zhu. 2014,*Fog computing: a platform for internet of things and analytics*, In: Big Data and Internet of Things: A Roadmap for Smart Environments Springer; 169-186.
[12] M. R. Garrey, R. L. Graham, and D. S. Johnson, *Resource constrained scheduling as generalized bin packing*, J. Combinatorial Theory, Series A, vol. 21, no. 3, pp.257-298, 1976.
[13] R. Buyya, S. K. Gar, and R. N. Calheiros, *SLA-oriented resource provisioning for cloud computing: Challenges, architecture and solutions*, in Proc. Inter. Conf. Cloud. Ser. Comput., pp. 1-10, 2011.
[14] A. Bhadani and S. Chaudhary, *Performance evaluation of web servers using central load balancing policy over virtual machines on cloud*, in Proc. Third Annual ACM Conference, ACM, 2010, Article no.16.
[15] V. V. Vazirani, *Approximation algorithms*, Springer, 2001.
[16] R. Panigrahy, K. Talware, Lincoln Uyeda, and U. Wideder, *Heuristics for Vector Bin Packing*, Microsoft Research, 2011.
[17] X. Zheng and Y. Cai, *Dynamic virtual machine placement for cloud computing environments*, in 2014 43rd International Conference on Parallel Processing Workshops, Sept 2014, pp. 121128.
[18] A. Singh, M. Korupolu, and D. Mohapatra, *Server-storage virtualization: Integration and load balancing in data centers*, in Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, ser. SC 08. Piscataway, NJ, USA: IEEE Press, 2008, pp. 53:153:12. [Online]. Available: http://dl.acm.org/ citation.cfm?id= 1413370.1413424
[19] G. Khanna, K. Beaty, G. Kar, and A. Kochut, *Application performance management in virtualized server environments*,

in Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP, April 2006, pp. 373381. [Online]. Available: http://dx.doi.org/10.1109/ NOMS.2006.1687567

[20] B. Speitkamp and M. Bichler, *A mathematical programming approach for server consolidation problems in virtualized data centers*, Services Computing, IEEE Transactions on, vol. 3, no. 4, pp. 266278, Oct 2010.

[21] M. Rahman and P. Graham, *Compatibility-based static VM placement minimizing interference*, J. Netw. and Comp. Appl.,vol. 84, pp. 68-81, 2017.

[22] E. Feller, C. Morin, A. Esnault, *A case for fully decentralized dynamic vm consolidation in clouds*, in: Cloud Computing Technology and Science (Cloud Com), 2012 IEEE 4th International Conference on, IEEE, 2012, pp. 2633.

[23] L. Cui, F. Tso, P. Pezaros, W. Jia, and W. Zhao, *PLAN: Joint policy- and network-aware VM management for cloud data centers*, in IEEE Trans. Parallel and Distributed Systems, vol. 28, no. 4, pp. 1163-1175, 2017.

[24] A. Beloglazov, J. Abawajy, R. Buyya, *Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing*, Future Generation Computer Systems 28 (5) (2012) 755768.

[25] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K.-M. Chao, J. Li, Virtual machine consolidated placement based on multi-objective biogeography-based optimization, inFuture Generation Computer Systems 54 (2016) 95122.

[26] J. Shi, F. Dong, J. Zhang, J. Luo, D. Ding, *Two-phase online virtual machine placement in heterogeneous cloud data center*, in: Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on, IEEE, 2015, pp. 13691374.

[27] F. López-Pires, B. Barán, L. Benítez, S. Zalimben, and A. Amarilla, *Virtual machine placement for elastic infrastructures in overbooked cloud computing datacenters under uncertainty*, Future Gen. Comp. Sys., Vol. 79, Part 3, pp. 830-848, Feb 2018.

[28] S.K.Mishra, P. Deepka, B. Sahoo, P.P. Jayaraman, S. jung, A.Y. Zomaya, R. Ranjan, *Energy-efficient VM-Placement in Cloud Data Center*,Sustainable Computing : Informatics and Systems, (2018) https::/ doi.org/10.1016/j.suscom.2018.01.002.

[29] B. Heidelberg, *Bin-packing, in Combinatorial Optimization, ser. Algorithms and Combinatorics 21. Springer, 2006, vol. 21, pp. 426-441.*

[30] http:// www.cloudsimplus.org

[31] https://aws.amazon.com/ec2/instance-types/

Edited by: Anand Nayyar
Received: Mar 15, 2019
Accepted: Apr 4, 2019

DOI 10.12694/scpe.v20i2.1529
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 349–364. http://www.scpe.org © 2019 SCPE

TRUST FACTOR BASED KEY DISTRIBUTION PROTOCOL IN HYBRID CLOUD ENVIRONMENT

VELLIANGIRI S. [*], RAJAGOPAL R. [†] AND KARTHIKEYAN P. [‡]

Abstract. In the Hybrid cloud deployment model, security is essential to restrict access while using resources such as virtual machine, platform, and application. Many protocols were developed to provide security via the cryptography technique, but these protocols rarely considered the trust factor which is an essential factor for cloud security. The existing Elliptic Curve Cryptography and Diffie Hellman key distribution mechanism failed to stress the trust factor, and further, they have provided not only higher complexity but also lower security and reliability. The proposed method comprised two stages: first stage, Group Creation using the trust factor and develop key distribution security protocol. It performs the communication process among the virtual machine communication nodes. Creating several groups based on the cluster and trust factors methods. The second stage, the ECC (Elliptic Curve Cryptography) based distribution security protocol is developed. The proposed Trust Factor Based Key Distribution Protocol reduced error rate, improve the key computation time and maximize resource utilization.

Key words: Cloud Computing, Elliptic Curve Cryptography, Clustering, Trust Factors, Key Distribution

AMS subject classifications. 94A60, 68M14

1. Introduction.

Cloud computing has emerged as a pivoted new field, differentiated from conventional computing by its focus on software as services, platform as services and infrastructure as services, these services are implemented by Cloud deployment models such as public cloud, private cloud or hybrid cloud. Cloud computing has added to the advances in computational and correspondence innovations - making financially achievable the combination of various clusters of heterogeneous network resources and services which, thus, have prompted the improvement of extensive large scale distributed system. A group key exchange protocol permits a set of people to agree upon a shared secret session key over an internet network [1, 2].

Cloud computing virtual resource can be characterized into three types such as compute, storage and network. Compute resource permit you to start virtual machines on cloud infrastructure. The main challenges of task scheduling in compute clouds are its highly dynamic environment, where the computing resources have their availability, access policies, security, reliability, etc. [3, 4]. Cloud computing has often jointly supported collaborative computing projects on the internet. Maximum of these projects have stringent security requirements. The capabilities for cloud systems and networks, such as broadband capabilities and distributed intelligence can greatly enrich reliability and efficiency, but they might also create much vulnerability if not deployed with the appropriate security controls. Providing security for such a large system may seem an unfathomable task, and if done incorrectly, can leave utilities open to cyber-attacks [5, 6].

Communicate among the group of members in securely and tried to reduce the complexity of informing the group key. The secure communications in the group or large group are more complex than Peer-to-peer communication because of the adaptability issue of the group key. Specifically, the expense of key establishment and renovating are generally pertinent to the size of the group and consequently turns into a performance bottleneck in attain scalability. To address this issue, our proposed approach named as the trust factor based key distribution protocol that combines the features of ECC key exchange protocol with the trust factor. ECC and Diffie Hellman key exchange protocols do not consider the trust factor in that which is relating to a "firm belief " with a treat like trustworthiness, reliability, and ability of the element for multicast communication in distributed computing [7, 8].

The primary goal of a cloud environment is increasing the domain to domain interactions, a secure environment and to ensuring the confidentiality of others. To achieve this "trust," notion needs to be considered so that the trustworthiness makes the geographically distributed method more reliable and attractive. The trust

[*]Department of Computer Science and Engineering, CMR Institute of Technology, Hyderabad - 501401, Telangana, India (velliangiris@gmail.com).

[†]Department of Computer Science and Engineering, Vel Tech Multi Tech Dr. Rangarajan Dr. Sakunthala Engineering College, Avadi -600062, Chennai, Tamilnadu, India (rajagopalrmail@gmail.com).

[‡]Department of Computer Science and Engineering, Karunya Institute of technology and Sciences, Coimbatore - 641114, Tamilnadu, India (nrmkarthi@gmail.com)

factor is an essential factor for cloud security. The above issues were resolved by using our proposed method of trust-based cloud security. In our proposed method has considered the priority of trust factor for enhancing the parameters of reliability and security. Hence to reduce this drawback in the existing protocols like ECC and Diffie Hellman key exchange, the trust factor based distribution security protocol using ECC is proposed, these trust factors based on key distribution security protocol, have been utilized to select the group leader among the communication nodes.

After that, the ECC method has been exploited in the secure communication process among group members. When compared to ECC and Diffie Hellman key exchange, the proposed method has given high performance in security. The proposed system provides improved scalability due to the trust factor Key Computation [9, 10].

Main contributions of this research work are summarized below:

- Trust Factor Based key distribution protocol is designed that helps the decision making during the key distribution.
- Trust Factor Based key distribution protocol performances are evaluated based on the theoretical analysis and experimental analysis.
- The objective of the research work is to minimize the error rate and computation time and maximize the resource utilization using Trust Factor Based key distribution protocol

The remainders of the paper are structured as follows, in section 2 crucial secure exchange protocol are reviewed. Section 3 we discussed the proposed system and implementation. In Section 4, we analyze and prove the correctness and safety of the proposed protocol and comparative study also reported in this section. Finally, we conclude the paper in Section 5.

2. Related Works. This section provides various recent related works regarding the secure key exchange protocol are reviewed. Wei et al. proposed a compiler which changes over secure group key exchange protocol into a secure passive keyless entry protocol. This method uses the labeled signature, public key encryption, the hash function. The proposed compiler is the main provably secure compiler without irregular prophets. As far as efficiency, the proposed protocol simply improved the two rounds of communications to the unique group key exchange protocol, which suggests that the derived password-authenticated group key exchange protocol is a static-round protocol as long as the secret group key exchange protocol is a static-round protocol [11]. Gonzales et al. proposed the model of security assessment for Infrastructure as a Service; this plan is confined to the single cloud service model, infrastructure as a service. The layers of the software stack underneath the guest operating system are coming under the control of the Infrastructures as service cloud service providers, key exchange security important in the software as service cloud service model [12].

Li et al. [13] introduced the blockchain based security model for centralized cloud storage. They found that the model has been contrasted with other two existing traditional model in respect of network transmission delay and security. Moreover, the system performance of conventional cloud design has been enhanced by the modified evolutionary algorithm which decreases the expenses on transmitting and replicas scheduling. Additionally, the proposed architecture has lower transmission delay compare with the genetic algorithm. Recently various exchange protocol proposed in a cloud environment concerning the healthcare environment. Challa et al. introduced a novelty of Secure Authentication Scheme (SAS) for healthcare in wireless sensor networks. To figure out the drawbacks in the LiuChung's existing method [14], the proposed framework enables a lawful client to modify/update the biometrics and password without prior communicating the trusted third party authority. Besides, it also permits a revocation scheme for disobeying nodes in the wireless sensor network. The proposed system uses BurrowsAbadiNeedham logic as well as the random oracle model. Additionally, the proposed scheme prevents replay attacks with the help of the widely-used AVISPA tool. The low communication costs and computation and along with high security make the proposed method fit for an extensive range of healthcare projects [15, 16].

Barman et al. designed a basic session-based exchange protocol that was worked like as biometric information of two conveying parties are pooled together to fabricate a cryptographic architecture are called a mutual locker. [17]. Zhou et al. have reported the key exchange challenges of k-nearest neighbors query over the encoded database in cloud services vendor. The revealed plan to accomplish secure outsourcing storage and k-nearest neighbors question in the cloud, where they utilized to improve dot product protocol and combined it into secure k-nearest neighbor's querying system. This technique also can counter-attack the assailant knowing the information that cloud data owner shares with query cloud client notwithstanding the encoded data [18]. Trapero et al. [19] have described sec Service Level Agreement (SLA) model created on defining a set of measurements for each security metric with which the cloud service providers can evaluate the strength of guaranteed service level objectives. For each service level objective, they have recognized a set of violations and detectable alerts, and they considered not only how to forecast, prevent, and eradicate secSLA violations, but also how to handle remedial and responsive activities in order to get-out producing more destruction, In the proposed secSLA trust factor is not included in the cloud consumers viewpoints. Kerberos is an authentication server whose main function is to authenticate client to server and server to client, the main security challenge in Kerberos is a key exchange between client and server. Talkhaby et al. proposed a new protocol with the powerful Diffie Hellman-DSA Key Exchange algorithm and the client's unique fingerprint impression tests. Biometric information utilized in the scheme comprehends the confinement of password-based authentication. In this work, using strong Diffie Hellman-DSA Key Exchange were performed as a Hard Key Exchange (HKE) mechanism with mutual authentication helped them to tackle the password predicting attack weakness in Kerberos version 5 protocols [20]. Nicanfar et al. proposed a Multilayer Consensus ECC-Based Password Authenticated Key-Exchange (MCEPAK) protocol, which allows secure communications between different layers of the smart grid control system and a home appliance. This method takes benefit of elliptic curve cryptography to provide a high-security level with a small key size while directing the possible resource limitations in the devices connected and they showed that proposed method decreases the security burden while giving elasticity against most of the attacks. Related to the X.1035 protocol, MCEPAK gains a lower load for the generation of the required passwords and hash function [21].

Hafizul et al. mentioned that ECC-based improves the security of key exchange of Peyravian-Zunic's secret key authentication method as well as expels the security defects Zhu et al.'s [22] method like clock synchronization issue and Identification and prevention of impersonation attack, etc. In this work, the proposed method enriches the computation of the symmetric key. It is used for private interchange of plaintext utilizing a symmetric key encryption algorithm. The performance examines of the proposed method is endorsed that the protection of data against impersonation attacks [23]. Murali et al. proposed a framework based on both traditional and quantum cryptography for cloud computing. This proposed method simplifies both the quantum channel and data channel for exchanging key and information correspondingly. The security framework based on quantum cryptography key distribution. The framework aims to support vital Quantum essential distribution communications with the cloud. This method is not suitable for the real cloud environment since the error rate is not an acceptable level of the cloud services providers [24].

Kumar et al. proposed a protocol that is based upon an RSA key distribution named as efficient centralized group key distribution protocol which is helpful from multiple points of view. It drastically decreases the computation load of the critical server by reducing the number of arithmetic operations needs to perform during crucial exchange and updating. However, the proposed method is not trusted for the cloud environment since it is considering the trust factor of cloud service providers [25].

Table 2.1 shows the merits and demerits of different key exchange protocol. Based on these the following conclusions have been made, the related works in the literature failed to incorporate certain factors such as Reliability and Recommendation which are identified as important aspects of key distribution in the hybrid cloud. We combined the features of ECC key exchange protocol using trust factor we introduced the method called Trust Factor Based Key Distribution Protocol (TFBKD); the proposed technique enables reduced computation and well-organized and secure group communication. It additionally ensured efficient rekeying for every communication session. Mehta and Singh talked about a few issues identified with exhibitions, security, and data transfer capacity issues in ASP .NET sites [26]. Singh et al. proposed technocrat ARIMA and SVR demonstrate (TASM), which gives a superior comprehension of the remaining burden and help to pick the estimating model. The gauging model additionally examined remaining testing for parameter setup of the models. Anyway, the proposed model works just in the web application situation [27]. Singh et al. present an Object-based Accountability Framework for data partaking in distributed computing. This structure provides responsibility to any client who is utilizing the cloud administrations utilizing a progressively particular and granular way utilizing article situated methodology. By utilizing this system, the client becomes more acquainted with that who can or who had gotten to his private data and when. This furnishes the client with a dependable choice to the client

Table 2.1
Comparison of the different key exchange protocol

Ref	Year	Technique	Advantage	Disadvantage
[11]	2018	Group key exchange protocol	High Efficiency	Protocol simply improved the two rounds of communications
[12]	2015	key exchange security	Efficiency and Reliability	Security provides only a single cloud service model Infrastructure as a service (IaaS)
[13]	2018	Blockchain-based security	Lower Transmission delay	It increases the computational load
[14]	2017	Secure Authentication Scheme (SAS)	High Efficiency	The Biometrics and password without prior communicating the trusted third party authority
[15][16]	2018	Biometric key authentication	Low communication cost	It prevents only replay attacks
[17]	2017	Crucial session based key exchange protocol	Efficiency and Reliability	Increases the computational load
[19]	2017	Service Level Agreement (SLA)	The strength of guaranteed service level objectives	The error rate is not an acceptable level of the cloud services cloud consumers
[20]	2016	Strong Diffie-Hellman Key Exchange	Efficiency and Reliability	Mutual authentication helped them to tackle for password predicting attack only.
[21]	2013	MCEPAK	Transmission delay	The trust factor is not included in the cloud consumers viewpoints.
[22][23]	2009 2013	Peyravian-Zunic's password authentication	Efficiency	Clock synchronization issue and Identification and prevention of impersonation attack
[24]	2016	Quantum cryptography key distribution	Low communication cost	The error rate is not an acceptable level of the cloud services providers
[25]	2018	Centralized group key distribution protocol	Efficiency and Reliability	The trust factor is not included in the cloud consumers's viewpoints.

to follow the exercises occurring over his private information [28].

3. Trust Factor Based Key Distribution Protocol. Cloud users submit a security demand(SD) to the resource broker to find the trust index (TI) of a cloud service providers, and resource brokers collect the truthfulness value from the cloud service providers, The resource Brokers map the task to the cloud service providers only if satisfies the following conditions ($TI >= SD$). The first difficulty in the group key communication is that interoperability between private cloud and public cloud. Data are passed across a multiple cloud deployment model. The second difficulty is establishing the trust relationship between the different cloud deployment models. To build a trust relationship between public cloud and private cloud, we introduced the method called Trust Factor Based Key Distribution Protocol (TFKD).

Our proposed method develops a trust factor based group key distribution security protocol to improve the security in the hybrid cloud deployment model. The security process is improved by considering the trust factors and ECC method. The proposed system mainly comprised of two stages namely, (i) Group Creation using the trust factor and (ii) Develop a hierarchy key distribution security protocol using ECC. These two stages are consecutively performed, and the security is increased more effectively and is discussed in section 3.1 and 3.2 respectively. The structure of our proposed Trust Factor Based Key Distribution Protocol using ECC method is illustrated in Fig. 3.1. We have designed a cloud service providers with the number of virtual machines communication nodes $C = \{c_1, c_2, c_3, ..., c_i\} 1 \leq i \leq N$ and each virtual machine communication node has varied number the jobs J. The virtual machine communication nodes jobs have the execution time also and it is represented as et. Moreover, the cloud service providers have the M number of resources, and it is denoted as $R = \{R_1, R_2, R_3, ..., R_i\} 1 \leq i \leq M$ and each resource has the processing time p_t. Before the grouping process, the virtual machine communications nodes want to allocate resources to complete their operations. During the

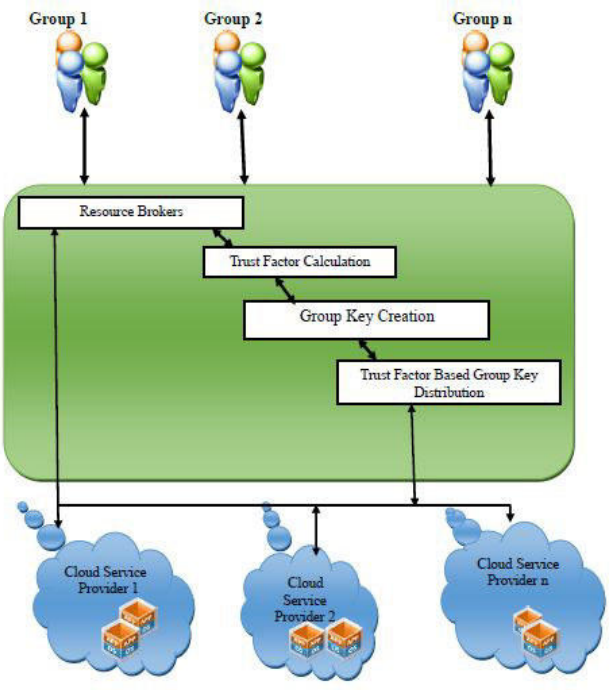


FIG. 3.1. *Structure of Trust Factor Based Key Distribution Protocol*

resource allocation, we have checked two conditions as stated below:

1. $J(e_t) < R_j(p_t)$
2. $J(e_t) > R_j(p_t)$

The jobs execution time, less than the corresponding allocated resource processing time means that we can avoid the job failure otherwise the job will be failed before it completes the operation. If condition (1) is satisfied, we can allocate the resources to the similar jobs. If the condition (2) is satisfied, we cannot do the resource allocation and go for other resources. Based on both the conditions (1) and (2) we allocate the resources to the jobs.

3.1. Group Creation Using Trust Factor. To perform the time-based clustering, we can create a database D, which contains joining time and the identification number of the communication node. Based on the time in the database, the virtual machine communication nodes are clustered. The clustered communication nodes are presented as $L = \{l_1, l_2, l_3, ..., l_g\}$ where g is a number of clustered groups and each cluster group l_g has a number of communication nodes.

$$I_g = \begin{cases} c_i, & c_i(j_t) > t_g \\ 0, & Otherwise \end{cases} \tag{3.1}$$

In Eq. 3.1, c_i is the virtual machine communication node, $c_i(j_t)$ is the joining time of the virtual machine communication node c_i and t_g is the time threshold value of the group g.

Trust factors: In cluster group lg a leader node is selected based on three trust factors. Here we have considered three trust factors namely: (RU), (RE), (RC).

1. Resource Utilization (RU): Resource utilization is calculated by checking the virtual machine communication node resource utilization time, i.e., which virtual machine communication node has utilized the maximum resource time.
2. Reliability (RE): Reliability factor is computed at different time slots, i.e. the virtual machine node which maintains the stable completion time at different slots.

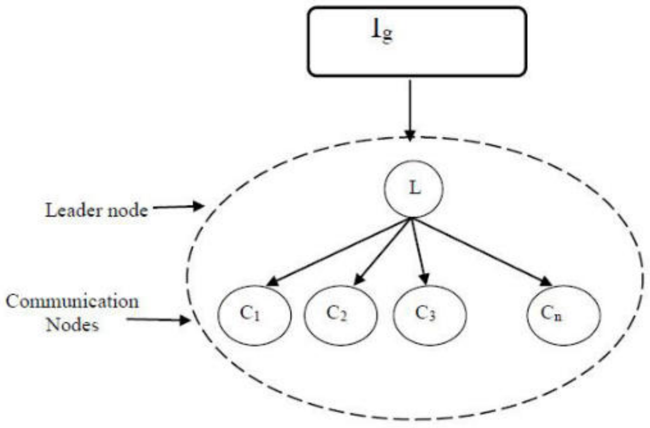


FIG. 3.2. *Cluster Model*

3. Recommendation (RC): the job completion time is calculated for all virtual machine communication nodes. The computed value is utilized as a node recommendation value.

Thus the created cluster group model with the group leader and members is shown in Fig. 3.2. The leader node is selected by averaging three trust factor values as,

$$a_i^{l_g} = \frac{RU_i^{l_g} + RE_i^{l_g} + RC_i^{l_g}}{3} \quad i = 1 \; to \; n \tag{3.2}$$

where $RU_i^{l_g}$, $RE_i^{l_g}$ and $RC_i^{l_g}$ are the resource utilization, reliability, and recommendation of the virtual machine communication node, i from the cluster group l_g and $a_i^{l_g}$ represent the average value of the virtual machine communication node i. Based on these above mentioned three trust factors and the computed average value, a leader node is to be selected from each cluster group. Next, the secure communication process is performed within the group members, and key distribution process is also explained in the next section.

3.2. Developing Trust Factor Based Key Distribution Protocol using ECC. We have utilized an ECC method for creating a private and public key for the group members. Based on the created public key, the group key is also generated for all cluster groups by the leader. The generated group key is updated when the new group member joins or leaves from the cluster group. The group members private and public keys are produced by the ECC method makes communication among the group's members more secure and also the generated keys are robust.

3.2.1. Key Generation by ECC. We generate keys to the group members by the elliptic curve cryptography. Elliptic Curve Cryptography (ECC) is also called public key cryptography, where each user or the device participating in the communication usually has a couple of keys: a public key and a private key, and these keys are used do the cryptographic function. Small key size is the main advantage of ECC.

The function of the ECC method is defined over two finite fields: Binary field and Prime field. The field is selected with a finite number of points for cryptographic algorithm function. Here, we used prime field function by picking a prime number P, and finitely enormous numbers of points are created on the elliptic curve, such that the created points b are between 0 to K. Then, we have arbitrarily chosen one basic point $P_0(x_0, y_0)$ for key exchange function and this point satisfies the equation of the elliptic curve on a prime field, which is defined as

$$y^2 \; mod \; P = x^3 + ax + b \; mod \; P \tag{3.3}$$

In Eq. 3.3, a and b are the parameters that define the curve and x and y are the coordinate values of the generated point b. We randomly select one basic point P_0 that satisfies the above-mentioned Eq. 3.1. To complete the key exchange process, we need to select a private key P_v on the sender side, which is a randomly selected integer less than P and generate a public key $P_u = P_v * P_0$. Now each cluster group l_g members has individual private P_v and public keys P_u. By using these public keys, a group key is to be generated for each cluster group by the leader which is explained in the following subsection.

Group Key Creation: Secure communication is established within or between the group members we need a key identification and a secure key among the members. The group members are communicated with each other by their private P_v and public P_u keys and also exchange the information. Within the cluster group l_g, a group key is created for providing authentication between/within the group members. Thus the created group key is updated when any of the single members leaves or joins from or within the group and group members leave or join from or within the group.

Group Key: Each cluster group has the group leader and group members. The group key is created by using the group members public keys. The group key formation equation is stated as,

$$(G_k)^{l_g} = n^{l_g} \sum_{i=1}^{n} (p_{u_i})^{l_g} \tag{3.4}$$

where $(G_k)^{l_g}$ is denoted as group key of a cluster group l_g, n is the total number of communication nodes (members) in the group l_g and $(p_{u_i})^{l_g}$ is the public key of the i^{th} virtual machine communication node in cluster group l_g.

Updating Group Key: The created group key is updated when a new member joins or leaves from the group. The process of key updating is given below,
(i) A single member joins into the cluster group l_g it means the group key is updated by using the equation as described as,

$$(G_k)^{l_g}_{new} = (G_k)^{l_g} + (n * p_{u_{i+1}}) \tag{3.5}$$

(ii) When a single member leaves from the cluster group l_g it means the group key is updated by using the equation as stated below,

$$(G_k)^{l_g}_{new} = (G_k)^{l_g} - (n * p_{u_{i-1}}) \tag{3.6}$$

By using these, a group, private and public keys, the communication process is performed among group members. The public and the private keys created by the ECC technique make our proposed technique attain more secured communication and also the group key is used to authenticate the members within the groups. The Fig. 3.3 depict the Process of Trust Factor Based Key Distribution Protocol.

Encryption and Decryption: Subsequently the communication process is carried between the group members and the leader nodes by using the ECC technique. The leader node encrypts the messages by using ECC and sends that encrypted message to their group members.

The encrypted plain text is sent in the form of,

$$e = (M_e, C_j) \tag{3.7}$$

$$M_e = M_0 * p_0 \tag{3.8}$$

$$C_j = (x, y) + M_0 * (S(p_v) * p_0) \tag{3.9}$$

In Eq. 3.7, M_e is encrypted plain text which is computed by an Eq. 3.8 i.e., the multiplication of the original message M_0 with the basic point p_0 and C_j is evaluated by an Eq. 3.9. In Eq. 3.9, $S(p_v)$ is the private key of the sender. This message e is sent to the receiver. The receiver receives the message e and decrypts the message by the following formula,

$$d = M_e - (R(p_v) * C_j) \tag{3.10}$$

where $R(p_v)$ is the private key of the receiver.

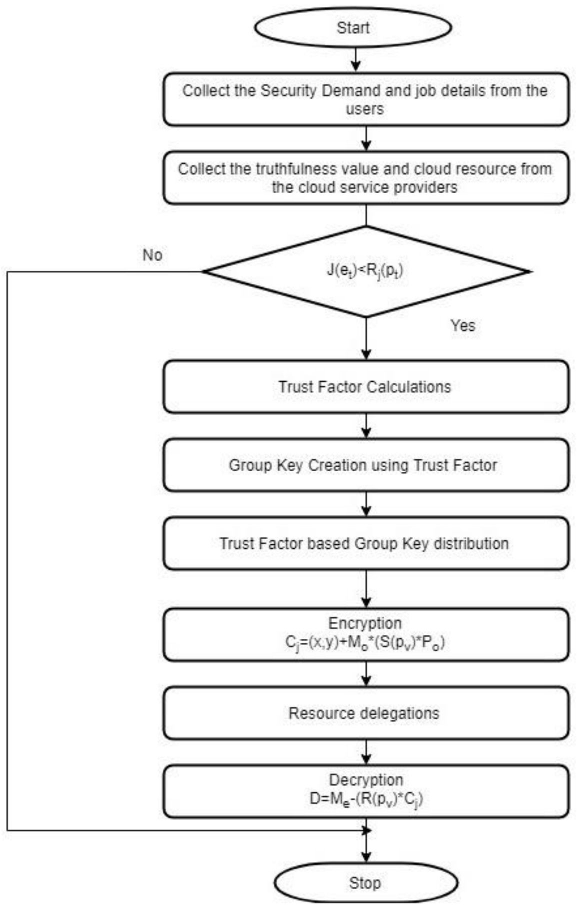

Start
Collect the Security Demand and job details from the users
Collect the truthfulness value and cloud resource from the cloud service providers
No
$J(e_t)<R_j(p_t)$
Yes
Trust Factor Calculations
Group Key Creation using Trust Factor
Trust Factor based Group Key distribution
Encryption $C_j=(x,y)+M_o*(S(p_v)*P_o)$
Resource delegations
Decryption $D=M_e-(R(p_v)*C_j)$
Stop

Fig. 3.3. *Process of Trust Factor Based Key Distribution Protocol*

3.3. Implementation of Trust Factor Based Key Distribution Protocol. The proposed protocol is implemented by using Open Nebula. Open Nebula is open source software for building cloud computing infrastructures and services. Open Nebula can be used to build public, hybrid and private cloud deployment model, and open nebula supports to integrate the existing cloud infrastructures [29] [30] [31]. Implementation of hardware and software details is listed in Table 3.1.

We have created two virtual datacenters using open nebula and public cloud deployment model using Amazon EC2. We integrated public cloud to existing open nebula private cloud environment which is depicted in Fig. 3.4. Users use the user interface which is written in Java and submit a job to the virtual machine present in the hybrid cloud. The submitted job is an interdependent job, and one job wants to communicate with another job running on the different virtual and different cloud environment model. If virtual machine securely intends to exchange the data between two different virtual machine, we need group key that should be transferred to all the virtual machine that is present in the hybrid cloud, so proposed Trust Factor Based Key Distribution Protocol is incorporated, and the job is communicated securely in the virtual data center. There are four crucial components in the implemented Hybrid Cloud deployment model such as Job Modeling Agent (JMA), Job Result status agent (JRSA), Trust server (TS) and Trust agent (TA) [32] [33].

TABLE 3.1
Implementation of hardware and software details

S.No	Hardware or Software	Details
1	Operating System	Supported host OS (CentOS, Ubuntu or RHEL) on all hosts
2	Hypervisor	Kernel-based Virtual Machine
3	Networking	CISCO SG-300 28 Port Managed Switch, The redundant 10Gbps storage network, Three 1Gb NIC interfaces ,1Gbps or 10Gbps network shared by service network, public and private virtual networks
4	Storage	IBM storage servers: iSCSI (40TB)
5	Authentication	X509
6	RAM	512 GB RAM
7	CPU	40 CPU Core (80 logical CPUs)
8	BUS	500-800MHz FSB
9	Amazon EC2 Public Cloud	#Instances Instance Type vCPU RAM(GB) OS 4 c3.2smal 8 4 Windows 1 m3.xlarge 4 15 Windows 2 c3.2xlarge 8 15 Windows 2 r3.2xlarge 8 32 Centos 1 r3.xlarge 4 32 Windows 1 m3.xlarge 4 15 Windows 2 m3.xlarge 4 15 Windows

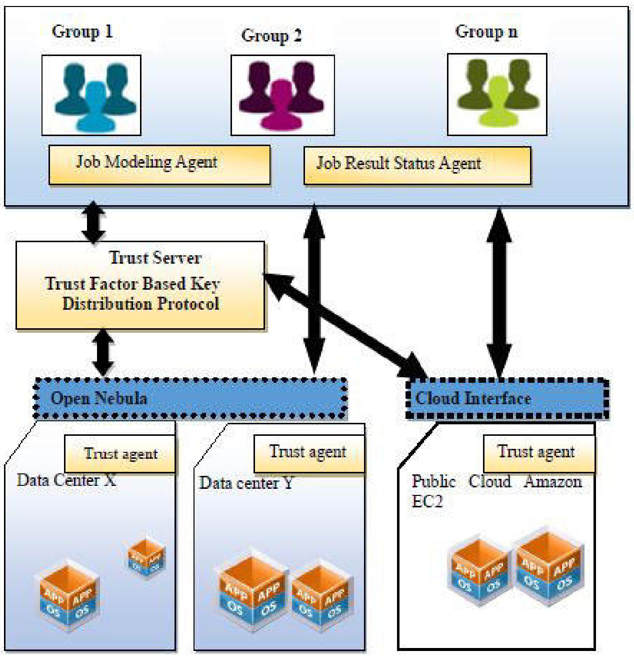


FIG. 3.4. *Integrated Key Distribution Protocol in Hybrid Cloud*

Job molding agent main function is to create a job; cloud consumers can specify the job completion time and the trust demand for their jobs. Their job can be scheduled for virtual machines according to their trust demand. Job result status agent obtains the results from the virtual machine. If the job execution is not completed in the virtual machines, then Job result status agent notify the failed job to the users by sending the failures notification message. A Trust server (TS) permits cloud consumers to maintain a trust relationship between

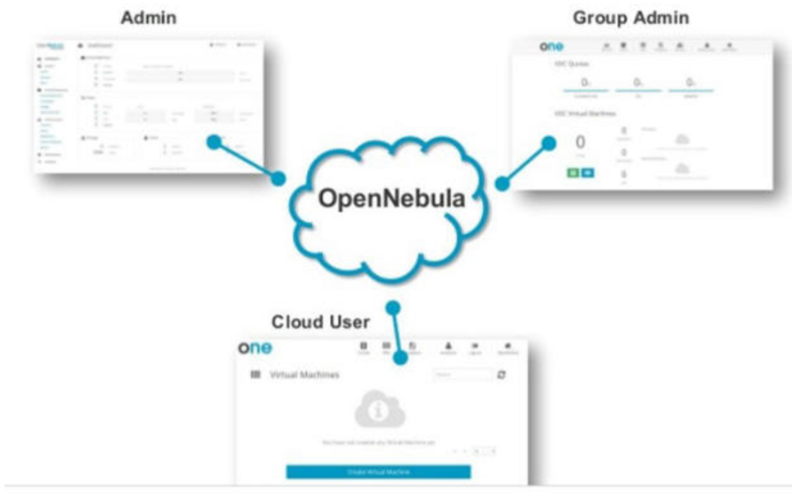


FIG. 3.5. *Open Nebula user account types*

the private cloud and public cloud to ensure cloud consumers can access virtual machines in the hybrid cloud. Trust agent collects the job success rate, virtual machine utilization, response capabilities, Reliability from the private cloud or public cloud and the transferred cumulative information to the Trust server to calculate the trust factor. If any virtual machine goes down, Trust agent generates the alarm signal and sent it to the Trust server for necessary action to be performed.

The Open Nebula and cloud interface permits a single point of entry to the hybrid cloud which consists of a trust agent and Datacenter. It secures authentication and authorization to the user. After authenticating the users private key, the server passes the job to the virtual machine communication node in the data center. Datacenter consists of Network Task Manager (NTM) and Target Method Interface (TMI). NTM receives job and starts to execute it. In the interim, NTM sends a response to JMA. After this, the cloud consumer can leave the system or surrender to another job. Throughout the execution, a Virtual space is formed to store all the information, including those wanted for execution and the files created by Abstract Job. NTM translates the Abstract Job into a collection of instructions that can be executed by Target Method Interface (TMI). For the duration of this conversion method, conversion tables in the incarnation database are used to map the abstract demonstrations to real instructions. TMI accepts incarnated job mechanisms from the NTM, and permits them to the confined batch schemes for execution. When the work is completed the virtual space is deleted.

The Open Nebula supports three types of account. All the account function is summarized below. The account interaction is illustrated in Fig. 3.5.
One admin: Special administrative account.
User: An OpenNebula user account.
Group: A group of Users.

4. Results and Discussion. The performance of the Trust Factor Based Key Distribution protocol is compared with the existing ECC and Diffie Hellman key exchange technique. In cloud hybrid cloud model; initially, we allocate the resources to each virtual machine communication nodes to complete their operations.

The resource allocation and their job completion performance are evaluated under three different number of communication nodes. The results that are obtained under these different job dataset sizes are given in Table 4.1.

After the resource allocation from the cloud system, the communication nodes are clustered by using their times. In our proposed method, we have taken the cluster limit as 8. The clustering result from our proposed technique is given in Table 4.2.

Table 4.1
Different Communication Nodes Job Completion Time with varying communication nodes

Virtual machine Communication nodes	Number of Jobs allocation respectively	Job completion time in seconds
10	4,5,10,10,8,8,10,4,6,5	38
20	2,2,6,4,2,2,5,1,9,3,3,3,4,1,2,5,4,5,6,2	18
30	1,3,3,2,1,5,2,3,4,3,4,3,1,4,1,4,3,2,3,1,1,1,3,1,2,3,4	13

Table 4.2
Time Based Clustering Results

Virtual machine Communication Nodes	Clustered Results			
	1	2	3	4
10	{2,3,7}	{4,8}	{9,1}	{5,6,10}
20	{12,13,3,6,8,1}	{11,7,2,15}	{4,9,14,18}	{5,10,17,20}
30	{14,21,27,28,11,8,13}	{0,4,12,17,20,24,15,23}	{3,6,10,19,22,25,29 }	{1,2,7,9,11,16}

In each cluster group, a leader node is selected based on the trust factors namely, (i) Resource utilization (RU) (ii) Recommendation (RE) and (iii) Reliability (RC). These three trust factors are calculated for all cluster group communication nodes. The sample calculated trust factors for the three cluster group results are given in Table 4.3.

Subsequently, in each cluster group, private keys and public key are generated by using the ECC technique whereas the group key is calculated by using these generated public keys and the private key of the group members. The security between the communications nodes is evaluated by creating random keys and measuring the security performances within the cluster group. The proposed and the existing methods error rate for the cluster groups is illustrated in above Table 4.4. The error rate is computed by the following formula:

$$ErrorRate = \frac{A_n}{T_n} X100 \tag{4.1}$$

where A_n is number of bits received and T_n is total number of bits in the original message. As it is seen from Fig. 4.1, our proposed technique has given a lower error rate than the ECC and Diffie Hellman algorithm. The existing algorithm obtains 40% average error rate performance in the communication within the cluster groups. Fig. 4.2 depict the rate of precision for man-in middle attack. The outcomes reveal that on categorization the information set with all the characteristics the average rate of error 89%, 83%, 79%, 66%,67%,97%,77% and 73% is acquired for ECC. Based on the usage of Diffe Hellman the error rate is increased. The error rate is decreased in Trust Factor Based Key Distribution Protocol is described during the usage of cluster formation with hybrid trust factor scheme.

From the Fig. 4.3, we can infer that key Renewal Computation time of proposed method provides 50 percentage lesser than the ECC and Diffie Hellman algorithm, this is because since the node is trusted, the computation virtual machine need not wait to assign new value while generating the key. Hence it is proved that our proposed technique provides more secure communication between the communication nodes than the existing method. Fig. 4.4 depict the Comparison of Key Renewal Computation time with different cluster size.

Fig. 4.5 depicts the comparison of Response time. The x-axis represents the different protocol and y-axis represents the response time in seconds. Response time is the length of time taken for a communication node to react to a given request or another communication node that is present in the different clusters. From Fig. 4.5, we can understand that the proposed method takes very less response time than ECC and Diffie Hellman protocol. This result is due to the management of the trust. The proposed approach enables efficient group communication.

Fig. 4.6 shows Resource utilization different method. The proposed method malicious node is not attacked by the trusted node, so it achieves the 91% resource utilization. In ECC and Diffie Hellman malicious node utilize the virtual machine resource this is due to ECC is not detecting the malicious node while they are

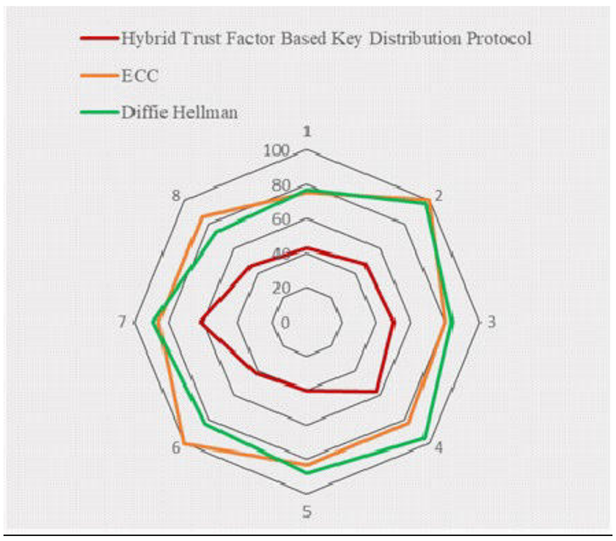


FIG. 4.1. *Radar chart Representation of the Proposed and Existing Methods Performance in terms of Error Rate*

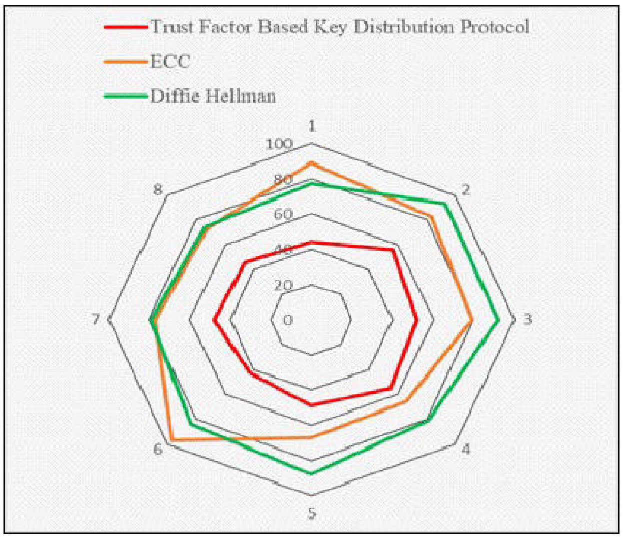


FIG. 4.2. *Error Rate for Man-in-Middle attack*

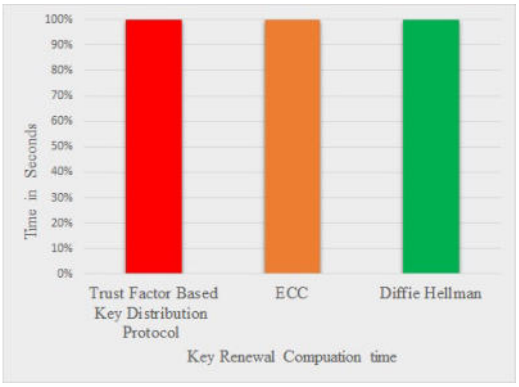


FIG. 4.3. *Comparison of Key Renewal Computation time*

Table 4.3
Trust Factors for Three Cluster Groups Communication Nodes and their Leader Nodes

Clusters	**Virtual Machine Communication Nodes**	**Resource Utilization**	**Recommendation**	**Reliability**
	2	1	24	13.33
1	3	0	34	9.88
	7	1	45	15.22
2	4	3	36	7.77
	8	3	57	10.33
3	9	0	52	11.44
	1	1	28	12.11
	5	0	31	11.66
4	6	3	39	18.0
	10	3	75	15.33

Table 4.4
Performance of Proposed and existing techniques Error Rate within the Cluster Groups

Cluster Groups	**Error Rate (in %)**		
	Trust Factor Based Key Distribution	**ECC**	**Diffie Hellman**
1	42.857	75	77
2	47.8	100	97
3	50	80	84
4	57	83.3	96
5	40	83.3	88
6	42.5	100	84
7	62	85.7	89
8	46	85.7	74

communicating with each other in the clusters.

Fig. 4.7 shows the communication overhead evaluation of Trust Factor Based Key Distribution Protocol with ECC and Diffie helman algorithm. Communication overhead time comparison, we consider that the user identity, virtual machine communication node identity, timestamp info, nonce , hash digest and Elliptic Curve Cryptography point $P = (P_x, P_y)$ are 256 bits, 128 bits, 64 bits, 32 bits correspondingly. Proposed method reduce commutation time between requester and virtual machine communication node compared to ECC and Diffie Hellman method. The reducing in the communication overhead time is suitable as the security is enhanced to an excessive point in the Trust Factor Based Key Distribution Protocol. The over-all investigational results confirm that the proposed protocol is efficient in terms of Communication overhead.

5. Conclusion and Future Work. We have proposed a Trust Factor Based Key Distribution protocol. The communication nodes were clustered using time-based clustering, and the leader was selected from among communication nodes by exploiting the trust factors. The ECC technique established the communication between the group members. The experimental results proved that our proposed Trust Factor Based Key Distribution protocol has reduced the error rate. Moreover, the comparative study shows that our proposed security protocols has given more secure communication and increase the resource utilization than the ECC and Diffie Hellman key exchange technique in the Hybrid cloud. Finally, the proposed method can be extended to allocate the job to the virtual machines by including defense capacity parameters such as intrusion detection, firewall, and antivirus capacity.

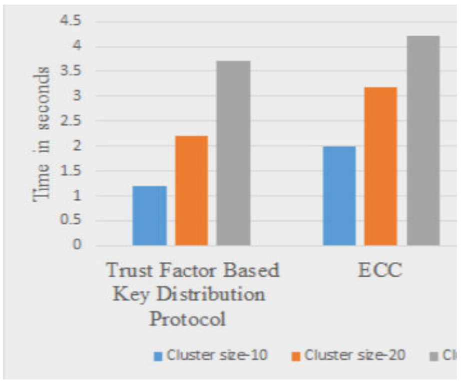


Fig. 4.4. *Comparison of Key Renewal Computation time with different cluster size*

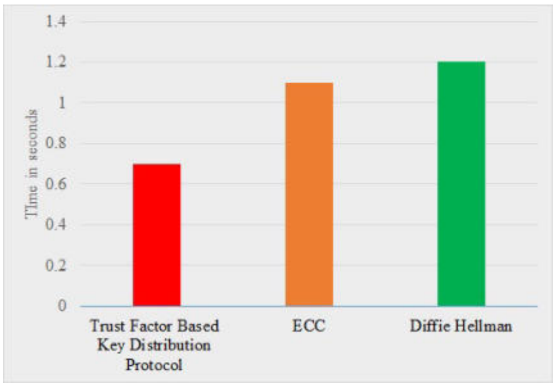


Fig. 4.5. *Comparison of Response Times*

REFERENCES

[1] Aleksandar Hudic, Paul Smith, and Edgar R Weippl. Security assurance assessment methodology for hybrid clouds. *Computers & Security*, 70:723–743, 2017.

[2] Ashish Singh and Kakali Chatterjee. Cloud security issues and challenges: A survey. *Journal of Network and Computer Applications*, 79:88–115, 2017.

[3] Shadi A Aljawarneh, Ali Alawneh, and Reem Jaradat. Cloud security engineering: Early stages of sdlc. *Future Generation Computer Systems*, 74:385–392, 2017.

[4] Gansen Zhao, Chunming Rong, Martin Gilje Jaatun, and Frode Eika Sandnes. Reference deployment models for eliminating user concerns on cloud security. *The Journal of Supercomputing*, 61(2):337–352, 2012.

[5] Mehedi Masud and M Shamim Hossain. Secure data-exchange protocol in a cloud-based collaborative health care environment. *Multimedia Tools and Applications*, pages 1–15, 2018.

[6] Antonio Celesti, Maria Fazio, Antonino Galletta, Lorenzo Carnevale, Jiafu Wan, and Massimo Villari. An approach for the secure management of hybrid cloud–edge environments. *Future Generation Computer Systems*, 90:1–19, 2019.

[7] Perumal Pandiaraja, Pandi Vijayakumar, Varadarajan Vijayakumar, and Raman Seshadhri. Computation efficient attribute based broadcast group key management for secure document access in public cloud. *J. Inf. Sci. Eng.*, 33(3):695–712, 2017.

[8] Flora Amato, Francesco Moscato, Vincenzo Moscato, and Francesco Colace. Improving security in cloud by formal modeling of iaas resources. *Future Generation Computer Systems*, 87:754–764, 2018.

[9] Maryam Farajzadeh Zanjani, Seyed Masoud Alavi Abhari, and Alexander G Chefranov. Group key exchange protocol based on diffie-hellman technique in ad-hoc network. In *Proceedings of the ACM 7th International Conference on Security of Information and Networks*, page 166, 2014.

[10] Y. Tan, J. Zheng, Q. Zhang, X. Zhang, Y. Li, and Q. Zhang. A specific-targeting asymmetric group key agreement for cloud computing. *Chinese Journal of Electronics*, 27(4):866–872, 2018.

[11] Fushan Wei, Neeraj Kumar, Debiao He, and Sang-Soo Yeo. A general compiler for password-authenticated group key exchange protocol in the standard model. *Discrete Applied Mathematics*, 241:78–86, 2018.

[12] Dan Gonzales, Jeremy M Kaplan, Evan Saltzman, Zev Winkelman, and Dulani Woods. Cloud-trusta security assessment

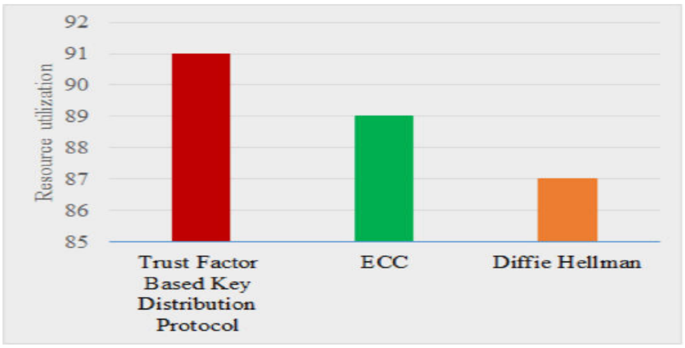


Fig. 4.6. *Comparison of Resource Utilization*

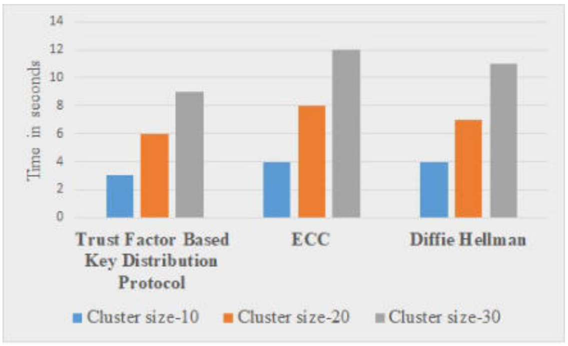


Fig. 4.7. *Comparison of Communication overhead with different cluster size*

model for infrastructure as a service (iaas) clouds. *IEEE Transactions on Cloud Computing*, 5(3):523–536, 2017.

[13] Jiaxing Li, Jigang Wu, and Long Chen. Block-secure: Blockchain based scheme for secure p2p cloud storage. *Information Sciences*, 465:219–231, 2018.

[14] Chia-Hui Liu and Yu-Fang Chung. Secure user authentication scheme for wireless healthcare sensor networks. *Computers & Electrical Engineering*, 59:250–261, 2017.

[15] Rajat Chaudhary, Anish Jindal, Gagangeet Singh Aujla, Neeraj Kumar, Ashok Kumar Das, and Neetesh Saxena. Lscsh: Lattice-based secure cryptosystem for smart healthcare in smart cities environment. *IEEE Communications Magazine*, 56(4):24–32, 2018.

[16] Sravani Challa, Ashok Kumar Das, Vanga Odelu, Neeraj Kumar, Saru Kumari, Muhammad Khurram Khan, and Athanasios V Vasilakos. An efficient ecc-based provably secure three-factor user authentication and key agreement protocol for wireless healthcare sensor networks. *Computers & Electrical Engineering*, 69:534–554, 2018.

[17] Subhas Barman, Samiran Chattopadhyay, Debasis Samanta, and Gaurang Panchal. A novel secure key-exchange protocol using biometrics of the sender and receiver. *Computers & Electrical Engineering*, 64:65–82, 2017.

[18] Lu Zhou, Youwen Zhu, and Aniello Castiglione. Efficient k-nn query over encrypted data in cloud with limited key-disclosure and offline data owner. *Computers & Security*, 69:84–96, 2017.

[19] Ruben Trapero, Jolanda Modic, Miha Stopar, Ahmed Taha, and Neeraj Suri. A novel approach to manage cloud security sla incidents. *Future Generation Computer Systems*, 72:193–205, 2017.

[20] Hamid Roomi Talkhaby and Reza Parsamehr. Cloud computing authentication using biometric-kerberos scheme based on strong diffi-hellman-dsa key exchange. In *Proceedings of IEEE International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pages 104–110, 2016.

[21] Hasen Nicanfar and Victor CM Leung. Multilayer consensus ecc-based password authenticated key-exchange (mcepak) protocol for smart grid system. *IEEE Transactions on Smart Grid*, 4(1):253–264, 2013.

[22] Lu Zhu, Sheng Yu, and Xing Zhang. Improvement upon mutual password authentication scheme. In *Proceedings of IEEE International Seminar on Business and Information Management*, volume 1, pages 400–403, 2008.

[23] SK Hafizul Islam and GP Biswas. Design of improved password authentication and update scheme based on elliptic curve cryptography. *Mathematical and Computer Modelling*, 57(11-12):2703–2717, 2013.

[24] G Murali and R Sivaram Prasad. Cloudqkdp: Quantum key distribution protocol for cloud computing. In *Proceedings of IEEE International Conference on Information Communication and Embedded Systems (ICICES)*, pages 1–6, 2016.

[25] Vinod Kumar, Rajendra Kumar, and SK Pandey. A computationally efficient centralized group key distribution protocol for secure multicast communications based upon rsa public key cryptosystem. *Journal of King Saud University-Computer*

and Information Sciences, 2018.

[26] Sahil Mehta and Parminder Singh. An approach to security, performance and bandwidth issues in asp. net websites. *International Journal of Computer Applications*, 70(27), 2013.

[27] Parminder Singh, Pooja Gupta, and Kiran Jyoti. Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. *Cluster Computing*, pages 1–15, 2018.

[28] Pradeep Singh, Parminder Singh, and Avinash Kaur. Object based accountability framework for information sharing in cloud computing. *International Journal of Computer Applications*, 115(19), 2015.

[29] C Yang, S Wang, and C Liao. On construction of cloud virtualization with integration of kvm and opennebula, 2011.

[30] Hyperconverged cloud architecture with opennebula and storpool, 2018.

[31] V Valliyappan and Parminder Singh. Hap: Protecting the apache hadoop clusters with hadoop authentication process using kerberos. In *Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics*, pages 151–161. Springer, 2016.

[32] Anand Nayyar. Private virtual infrastructure (pvi) model for cloud computing. *International Journal of Software Engineering Research and Practices*, 1(1):10–14, 2011.

[33] Anand Nayyar. Interoperability of cloud computing with web services. *International Journal of ElectroComputational World & Knowledge Interface*, 1(1), 2011.

Edited by: Anand Nayyar
Received: Mar 16, 2019
Accepted: Apr 4, 2019

DOI 10.12694/scpe.v20i2.1533
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 365–376. http://www.scpe.org © 2019 SCPE

INFLUENCE OF MONTORING: FOG AND EDGE COMPUTING

VIVEK KUMAR PRASAD, MADHURI BHAVSAR AND SUDEEP TANWAR[*]

Abstract. The evolution of the Internet of Things (IoT) has augmented the necessity for Cloud, edge and fog platforms. The chief benefit of cloud-based schemes is they allow data to be collected from numerous services and sites, which is reachable from any place of the world. The organizations will be benefited by merging the cloud platform with the on-site fog networks and edge devices and as result, this will increase the utilization of the IoT devices and end users too. The network traffic will reduce as data will be distributed and this will also improve the operational efficiency. The impact of monitoring in edge and fog computing can play an important role to efficiently utilize the resources available at these layers. This paper discusses various techniques involved for monitoring for edge and fog computing and its advantages. The paper ends with a case study to demonstarte the need of monitoring in fog and edge in the healthcare system.

Key words: IoT,Cloud Computing, Fog Computing, Edge Compuitng, Monitoring.

AMS subject classifications. 68M20, 68M14

1. Introduction. Cloud Computing [1] is an important platform for the industries. The establishments which trust heavily on data are progressively likely to make use of cloud, edge, and fog computing infrastructure. The aforementioned architecture will allow the organizations to take advantages of a variety of computing and data storage resources such as Industrial Internet of things(IIoT) and have their different layers with respect to cloud,fog, and edge [2]. The edge computing allows processing to be done locally at numerous decision points for the determination of dropping the network traffic flow. Fog computing and edge computing seem similar since they together include carrying intellect and processing nearer to the formation of information. However, the difference between the two lies in where the place of intelligence and compute power is positioned. A fog environment homes intelligence at the local area network (LAN). This architecture communicates data from endpoints to a gateway, then this is conveyed to sources for processing and yield transmission. Whereas the edge computing chairs processing power and intelligence in devices like embedded automation controllers.

The Fog and edge computing's main goal are to gather, process and analyze the data from the resources more efficiently as compared to the architecture of the outmoded cloud with similar objectives [3]; which are listed below:

- To decrease the volume of data directed towards the cloud
- To reduce the internet and network latency
- To progress system reply time in isolated mission-critical submissions(tasks).

The key variance between the dual theories is that both fog computing as well as the edge computing include the transfer of processing and intelligence abilities down nearer to where the information instigate; i.e. at the network edge [4]. Hence the main difference between the two constructions is exactly where the computing power and intelligence are kept and has been discussed here.

- Fog computing drives intelligence down to the local area network (LAN) level of network design and architecture, processing data in a fog node (or the gateway of IoT).
- Edge computing shoves the processing power, intelligence, and communication capabilities of an edge appliance or gateway directly into devices such as PACs (Programmable Automation Controllers).

In both the cases, the data will be generated from the source such as physical assets like sensors, pumps etc These devices achieve a task in the physical world like pumping water or sensing the biosphere around them; which are called as things that results into the (IoTs) Internet of Things.

The cloud is the base for edge and fog computing which has been depicted in Fig 1.1 where the cloud is used as the center for data analytics and business intelligence [5] and below cloud fog will be present and does analysis in the local network and after that, another layer is present which is called edge layer, where the computation is performed on the distributed devices (or nodes) itself, such as edge devices or smart devices as to disparate to mainly taking place in a integrated cloud environment.Even the response time too will change

[*]CSE Department, Nirma University, Ahmedabad, Gujarat, India (vivek.prasad@nirmauni.ac.in, madhuri.bhavsar@nirmauni.ac.in, sudeep.tanwar@nirmauni.ac.in).

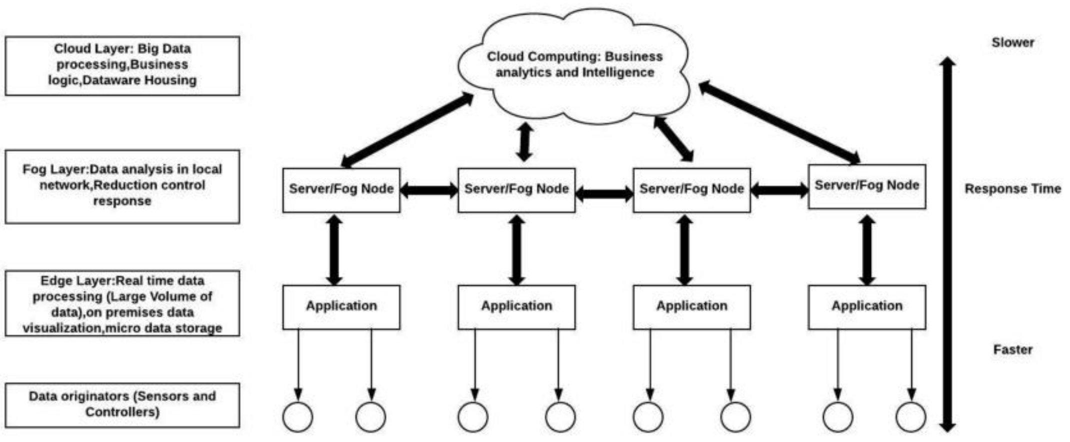

Cloud Layer: Big Data processing,Business logic,Dataware Housing
Fog Layer:Data analysis in local network,Reduction control response
Edge Layer:Real time data processing (Large Volume of data),on premises data visualization,micro data storage
Data originators (Sensors and Controllers)
Cloud Computing: Business analytics and Intelligence
Server/Fog Node
Server/Fog Node
Server/Fog Node
Server/Fog Node
Application
Application
Application
Application
Slower
Response Time
Faster

FIG. 1.1. *Data processing layer stack in cloud, fog and edge computing*

as we move from bottom to top, as the load itself will be distributed. The advent of technologies such as the Internet of Thing(IoT), Mobile edge computing (MEC) [6], software-defined networks [7] and network function virtualization (NFV) [8] are strongly challenging the cloud model for distributing its services among these technologies, even the prognostication regarding the number of connected applications and devices which will munch the cloud resources and will also generate massive volumes of data is a severe challenge for the current model for cloud computing. Hence there is a requirement for the advance of appropriate management systems which will enable an operator to aggregate, give description the availability of the distributed resources and implementing different kinds of services which will be managed by the operators itself. Nevertheless designing and architecting the compatible management system is a challenging task because of the Fog/Edge infrastructure; that is significantly differs from the traditional cloud with respect to dynamicity, heterogeneity and impending enormous distribution of the resources and networking environments. In this paper, the focus is on the various aspects of monitoring the services which are the key elements to any of the management systems of distributed computing environments. Several solutions has been proposed for the cloud, grid and clusters however none of them are well suited for the Fog/Edge Computing and the goal of this paper is to pave the way towards the monitoring systems which are well suited for Fod/Edge computing. The multilayer architecture depicted in Fig. 1.1 offers the following improvements over the classical cloud computing technique are as follow:

- The Network Traffic will reduce: Edge computer layer are able to filter needless data and significantly consolidate only important information that must be flowed into the cloud Computing and then received, processed and stored.
- Applications Performance will Increase: As latency will be reduced because the data processing is happening at the local edge nodes which are placed after to end-devices or end-users instead of the centralized cloud.
- Facilitating load balancing for the approaches used: Scaling the power of computation locally on the edge nodes instead of running entirely on the cloud.
- Minimizing the consumption of energy: The jobs can be offloaded from the end devices to edge nodes instead of cloud which are situated faraway. This will decrease the energy consumption of the end nodes or devices.

The contribution of the paper are listed below:

- To identify the various monitoring requirements in the dynamic environs for edge and fog computing
- The Qualitative study of major solutions which are prevailing
- To gather the influence of the deployment policy of functions for monitoring services for functional and non functional properties of edge and fog computing.

The rest of the article has been organized as follows: section 2 presents the problem statement and the key

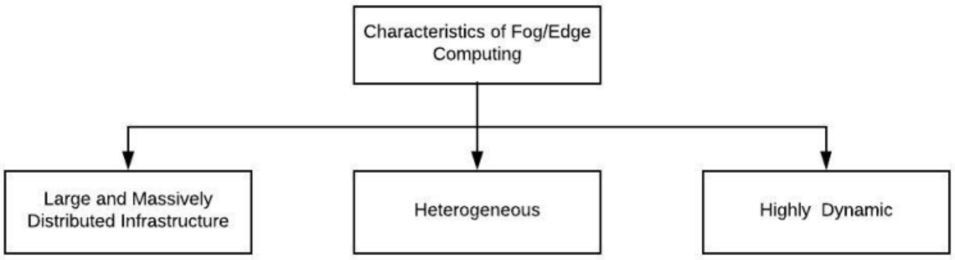

Characteristics of Fog/Edge Computing
Large and Massively Distributed Infrastructure
Heterogeneous
Highly Dynamic

FIG. 2.1. *Characteristics of the Fog/Edge Computing*

properties of the monitoring services and its associated metrics. Section 3 deliberates the study of the taxonomy of monitoring services in edge and fog computing.Section 4 highlights the perspective of the paper,than in section 5 various open issues has been described.Section 6 describes a case study on healthcare monitoring system, finally followed by the conclusion in section 7. We trust that this paper delivers contributions of interest for the research communal, analyzing the literature and shedding light on the present and forthcoming research matters on fog/edge computing monitoring and prediction.

2. Monitoring Properties and Metrics. As this has been discussed previously that Fog/Edge computing differs from cloud computing with respect to the certain characteristic as shown in Fig. 2.1, such as its distributed infrastructure, heterogeneous and highly dynamicity. These essential features complicate its management [9] and has been discussed below:

- Large and Massively Distributed Infrastructure [10]: The Fog/Edge machinists infrastructure is deployed across manifold sites. There can be distances among these resources and the connections can be in terms of wired (fiber or copper etc.) and wireless (wifi or microwave etc.) links. Here again, the distance and the type of link can affect the latency as well as the bandwidth too.
- Heterogeneous [11]: The Infrastructure is collection of many heterogeneous resources such as
  - Server Storage
  - Various Routers
  - Switches: General Purpose
  - Gateways, network links devised
  - Computer Servers
- Dynamicity [12]: The tasks which have small lifecycles are often migrated and instantiated. The same is with respect to the Software-defined network capabilities where the modifications of the network will happen and can be changed on the fly. This also happens in the lower level too, where Edge Devices (EDs) may join or leave the network permanently as per the service usages, policies , failures, and operations of the maintenance.

These resources have different characteristics in terms of capacity, usages, and reliability. The virtualization introduces another level of heterogeneity as the operated resources can be physical or virtual.

The monitoring facility on a Fog/Edge [13] with good availability must satisfy the following criteria: The services must be able to oversee a large number of resources. This should hold a instant growth of load or this should able to adapt to an increase of request rate from the perspective of different management systems. The system monitoring can be intended to allow its adaptation to any resource elimination or in the context of virtualized settings where the resources are of fewer consistent. Monitoring services rely on the network to observe the resources available on remote locations, in other words, this should ensure the retransmission of the signals in case of network let-downs. The monitoring levels for fog/edge computing can be classified as:

- Infrastructure or VMs computational resources.
- Dockers Containers
- Networks

Table 2.1
Metrics used for VM Usages

Goal	Measured metrics (VM-level)
Federated clouds Monitoring Mechanism	Memory,CPU,Network usage Disk, etc
Modeling resource usages of cloud tasks	Disk,CPU, Network usage
IaaS cloud monitoring	Disk, CPU, Memory
IaaS cloud monitoring	Memory, CPU
Provisioning Intelligent resource	Memory, CPU

Table 2.2
Common set of parameters of container table

Container level metrics	Salient aspects
rx_bytes	Bytes identified by the container
rx_packets	Packets identified by the container
tx_bytes	Bytes transferred by the container
tx_packets	Packets transferred by the container
Memory_usage	%memory usage of the container
io_service_bytes_write	Bytes written to block device by the container

- Measurement of application specific parameters.

Each one of the above has been described below:

- Infrastructure or VMs Computational Resources:VM level monitoring:-
  To have well-organized resource utility and avert any issues in monitoring and virtualized resources of the VMs used in the cloud infrastructure and edge nodes are serious issue. Performance optimization can be best attained by efficiently monitoring the utilization of the virtualized resources. Competences for monitoring such resources comprise of resources such as the usage of CPU, memory, storage, and network as shown in Table 2.1 along with their metrics.
  - Usage of CPU indicates the quantity of actively used CPU as a fraction of total obtainable CPU in a VM.
  - Usage of memory specifies the quantity of memory that is utilized on the designated Infrastructure.
  - Disk usage denotes to the quantity of data read or inscribed by a VM. Or this may also designate the percentage of used drive space.
  - Network usage is the capacity of traffic on a target network
- In contrast with VMs, the usage of containers which does not involve an Operating System (OS) to boot up to an additional method of server virtualization is swiftly growing in acceptance. An another reason for the same is the container images have a tiny size as compared to VM full images. The example for the container based virtualization platforms such as Container of Google (GKE) and Amazon EC2 container service (ECS) can be used as alternatives to the hypervisor-oriented approach. This is tranquil to pull container images of the application components across the nodes in the cloud. Even the agility is required as the migration technique is the best tool for numerous determinations such as load balancing, reallocating resources (scalability) and dealing with failures due to hardware etc.The the main intention here should be to reduce the downtime in the cloud. Table 2.2 and Table 2.3 indicates the common set of container level policies which can be monitored and useful with respect to the content of the task's adaptation.
- Network
  The Network infrastructures [14]conditions tend to change every time and hence the big challenge is to keep the performance of the network high. Even the end to end link quality also depends on the entire communication passing through the edge computing framework regardless of the fact that hat network technologies are being used by the network. The edge paradigm have four forms of connections of the

Table 2.3
Different container level system for monitoring

Goal	Measured Metrics of container level
Time critical tasks:Cloud application	Network Usage,disk,memory,CPU
Performance Assessment of virtualization technology	Network Usage,disk,memory,CPU
I/O bound application: Performance Assessment of virtualization technology	Disk
Bandwidth - intensive application: support application QoS	Usage of Network

Table 2.4
Netork:Link quality monitoring systems

Goal	Network Level Measured Metrics
Audio/Video streaming Service: Support Application QoS	Delay
Gaming System Cloud: Support Application QoS	Packet Loss,Delay,Throughput
Communication Services: Support Application QoS	Throughput, Packet Loss, Delay
Multimedia Services: QoE to the users	Jitter, Packet Loss, Delay, Throughput
Real time streaming services: Support Application QoS	Jitter, Packet Loss, Delay, Throughput

network among the application mechanisms and has been described below
- Communications amongst an edge node and cloud datacentre.
- Communications among any edge nodes itself.
- Communications amongst edge nodes and IoT users/objects.
- Communications amongst/within the cloud data centers.

Table 2.4 indicates the monitoring aspects for the network parameters as well as the metrics.
- The framework for the edge computing is used for any of the undecided dedicated application for any type of software purpose. As every application is running in the cloud requires to be extended to embrace the application level monitoring abilities to measure metrics to identify the current information about the performance and its services status. Table 2.5 summarizes the various works to monitor metrics of application level.

3. Major Monitoring services. Fig. 3.1 indicates the taxonomy for the edge computing monitoring requirements [15] and each branches for the same has been discussed below.

3.1. Edge Computing framework: Monitoring Requirements.
1. End to end quality link level [16]
   - Identify the end to end network QoS parameters: Certain parameters such as jitter, loss of packet, delay and throughput needs to be monitored for its quality level measurement (QoS) in regard with the end to end network statistics.
   - Assist for On-demand network configuration:To manage and control the virtual network resources dynamically the computing framework should provision programmable networks. Which will enhance the monitoring solutions within the edge computing framework.
2. Container level
   - Independent from the specific cloud infrastructure provider:The job will be easier if the monitoring solution will be designed for the specific cloud platform, the challenges arise when the generic monitoring solution will be required which may work with multiple providers cloud infrastructure.
   - Support dynamic resource management:The monitoring solution will deal with the collection and detection of the dynamic cloud environment. As compared with the centralized data center the edge computing requires agile monitoring approach, as here the end devices are often accessible and unreachable, the mobile devices will be there or they always keep on changing their states.
   - Assits for different kind of monitoring hardware visualization: The monitoring solution should be able to make available to the federated clouds in addition to the virtualized hardware.

Table 2.5
Application level monitoring systems

Goal	Application level measured metrics
Twitter:Cloud Application	Application throughput(data-items/ second), Response Time(time/ data item)
SLA violation detection: Cloud Application	Application throughput, response time
Web application: Cloud	Application throughput, response time
Audio and video services: Cloud Application	Response time
Live video streaming services: QoE to the users	Application throughput,video quality and dropped frame

3. Common terms
    - Muti-tenant cloud environment: This is used for identifying different types of users view and their rights to make use of the systems, for example, the scenarios where the different users will be sharing the same hardware resources and application instances, different users should be able to identify their measurable parameters and gain access to the data which are meant for them only.
    - Support desired monitoring time interval:The span of the interval for monitoring is needed to safeguard reliability and avoid overhead in the environment of the systems.
    - Assit for long term storage o measured data: The monitoring data should be optimally stored and can be used for the future perspective or deal with future adaptation tactics.
    - Support for scalability adaptation policies: The scalable monitoring solution is needed for elastic management of the infrastructure, which also support policies for scaling the monitoring solutions. As these techniques may be useful for autonomic self configuration of the system.
    - Assist to setup for automated alerts:The alerts indicate the management aspects of the resources. As an example, the monitoring technique should be able to generate alerts if the operational policies or metrics reaches its threshold value.
    - Customized based on monitoring requirements: This is the requirement of every monitoring techniques to changes its policies extension to allow other metrics to be generated/ invoked as and when the conditions of the specific cloud environs change.
4. Application level
    - Support application topology and reconfiguration: The monitoring system should be responsive to the different topologies and reconfiguration of the cloud-based environs in order to identify the analytics and operational policies collections for better response of the monitoring scheme.
    - Effective measurement of application performance:The small measurement intervals may destructively affect the invasiveness of monitoring tools and on the other aspect, the slow sampling rates will damage the accuracy of monitoring data. Hence the intervals must be determined optimally.
    - Support multi tier application:The monitoring system must be able to collect data from multi-tier applications and also should be able to cooperate with the other tier.
    - Adaptation to time varying applications:The solutions for the monitoring systems should be adaptable and should change over time (upgrades), as new definitions of tasks response time matters.

3.2. Fog Computing: Monitoring Requirement. Are summarized in Figure 3.2.

4. Perspectives of the Monitoring Approach. The fog computing [17] is acting like a connexion between the cloud, storage services, and IoT devices, fog computing is a part of cloud computing archetype which makes the cloud closer to the edge network. This delivers a virtualized computational model , networking of the resources and the storage in between clouds datacentre and the end devices. .The properties of fog computing is described in Table 4.1

- Assist to fog nodes configuration:The computational nodes with varied design and formations which are proficient to deliver infrastructure for fog computing at the edge of the network.Such as Vehicular network, cloudlets, servers,base stations and networking devices.These aforementioned environs are

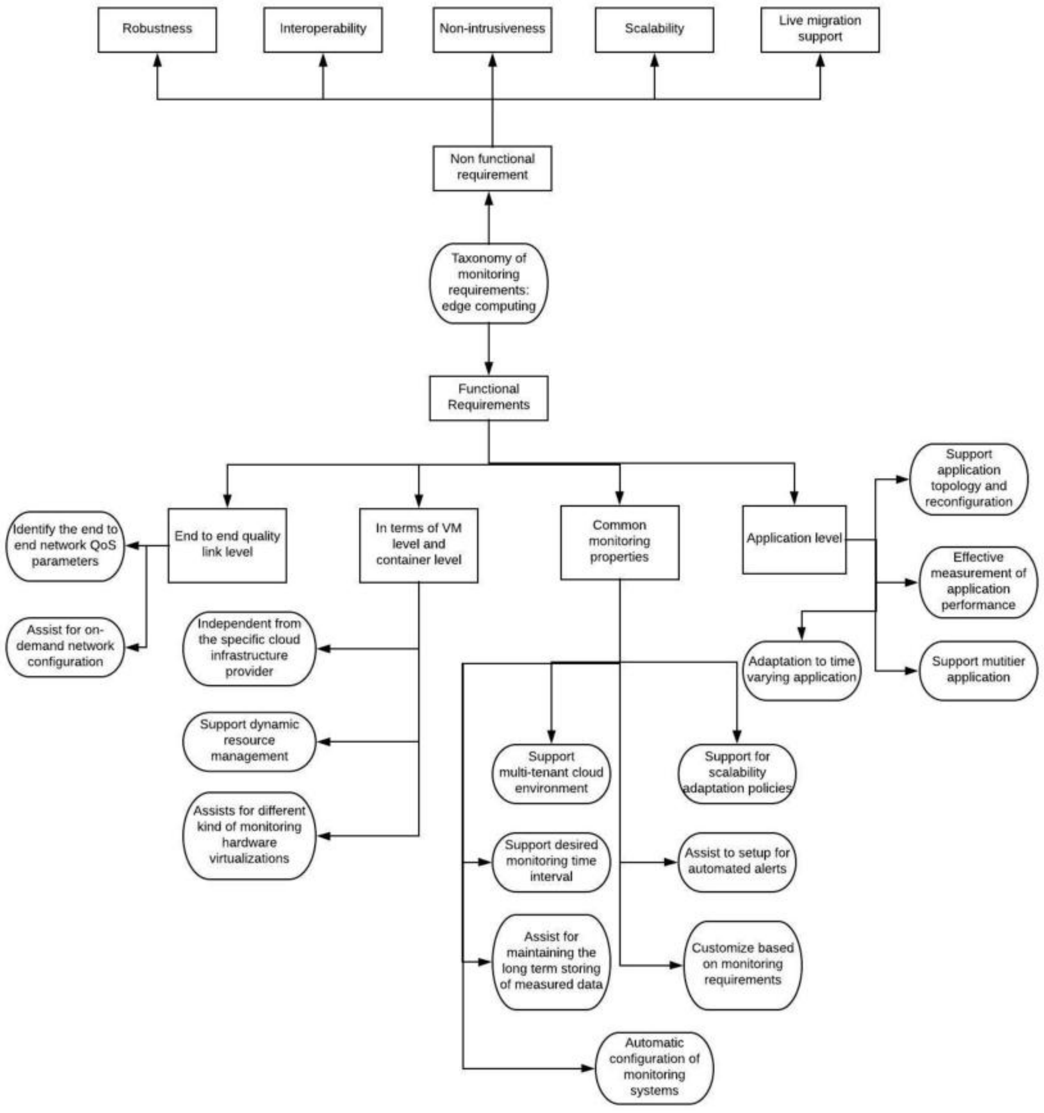

Robustness
Interoperability
Non-intrusiveness
Scalability
Live migration support
Non functional requirement
Taxonomy of monitoring requirements: edge computing
Functional Requirements
End to end quality link level
In terms of VM level and container level
Common monitoring properties
Application level
Identify the end to end network QoS parameters
Assist for on-demand network configuration
Independent from the specific cloud infrastructure provider
Support dynamic resource management
Assists for different kind of monitoring hardware virtualizations
Support multi-tenant cloud environment
Support for scalability adaptation policies
Support desired monitoring time interval
Assist to setup for automated alerts
Assist for maintaining the long term storing of measured data
Customize based on monitoring requirements
Automatic configuration of monitoring systems
Support application topology and reconfiguration
Effective measurement of application performance
Adaptation to time varying application
Support mutitier application

Fig. 3.1. *Edge Computing and Monitoring Needs*

Table 4.1
Properties of fog computing

Properties	**Fog Computing**
Latency	Less
Infrastructure	Limited computing and storage
Server nodes location	Edge of the local network
Client and server distance	One hop
Security Measures	Hard to identify
Attack on information	Maximum Probability
Deployment	Distributed
Awareness about the location	Yes

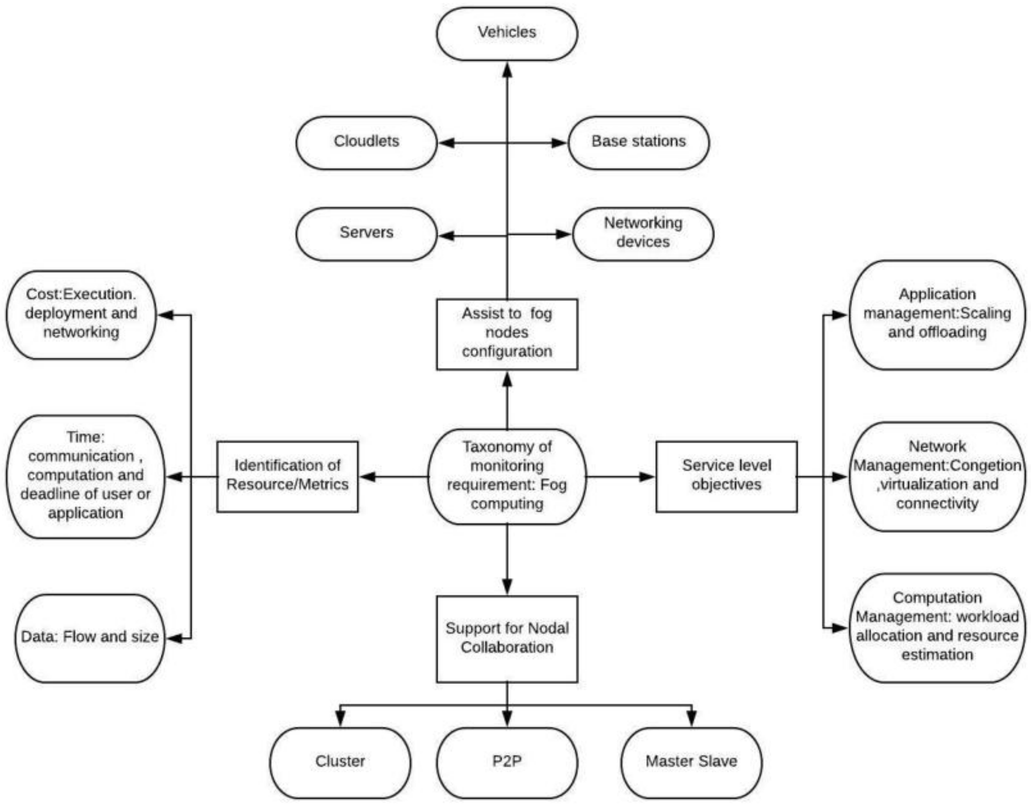

Vehicles
Cloudlets
Base stations
Servers
Networking devices
Cost:Execution. deployment and networking
Assist to fog nodes configuration
Application management:Scaling and offloading
Time: communication , computation and deadline of user or application
Identification of Resource/Metrics
Taxonomy of monitoring requirement: Fog computing
Service level objectives
Network Management:Congetion ,virtualization and connectivity
Data: Flow and size
Support for Nodal Collaboration
Computation Management: workload allocation and resource estimation
Cluster
P2P
Master Slave

FIG. 3.2. *Fog Computing and Monitoring Needs*

highly dynamic in nature and monitoring these environs plays an important role for maintaining QoS.

- Service Level objectives:The service level objectives [18] can be reached by developing fog computing as an transitional layer between end devices and cloud data centers and consists of
  - Application Management like offloading and scaling.
  - Management of network: connectivity,virtualization and congestion
  - Computation Management: resource estimation and workload allocation.

  Most of the SLOs are administration concerned and cover the issues as latency,resources,cost, power , application and data.
- Support for nodal collaboration:The techniques that are available for nodal collaboration [19] in fog computing has been specified and they are as follow:
  - Cluster
  - Peer to peer
  - Master-slave

  The cluster-based association is found to be better in certain cases where the advantage of several Fog nodes work simultaneously.The network overhead plays an important role here.In case of peer to peer(P2P),The access and reliability related issues are prime with respect to P2P nodal association. In master-slave partnership, the master fog node controls processing loads, resource management, control functionalities of the corresponding slave nodes.
- Identification of resources or metrics: Metrics or operational policies can be in terms of
  - Cost Execution, deployment and networking.
  - Time:communication time,computational time, deadline time etc.

Table 4.2
Cloud Monitoring aspects and functional requirements

Tool	Open Source	Monitoring of the container	Quality Monitoring of end to end link	Monitoring of the application	Methods for data storage	GUI facility
Tower 4Clouds	Yes	No	No	Yes	Graphite, Influx DB	Yes
Jcatascopia	Yes	No	No	Yes	Apache Cassandra, MySQL	Yes
Lattice	Yes	Yes	No	Yes	Distributed hash table	No
DARGOS	Yes	No	No	Yes	Neutron DBs and Nova	Yes
PCMONS	Yes	No	No	No	MySQL and Flat file	Yes
OpenNebula	Yes	No	No	No	Apache Cassandra, MySQL SQlite	Yes
Nagios	Yes	No	Yes	No	MySQL , Flat file	Yes
Zabbix	Yes	Yes	Yes	Yes	SQLite, MySQL, Oracle	Yes
Ganglia	Yes	No	No	Yes	RRDtool	Yes
Zenoss	Yes	Yes	Yes	Yes	ZODB, Hbase, MySQL	Yes

– Data flow time
  In Fog computing model, time is well-thought-out as one of the vital features for service provisioning and effectual resource.

There are various open source monitoring tools that are available for cloud computing and the same can be used to monitor the edge and fog computing environment [20], the same has been depicted in Table 4.2 where these tools are analyzed with the functional requirement (container monitoring, application monitoring, quality monitoring of end to end, methods for data storage and GUI facility) which has been discussed earlier.

5. Suggestions for developing new monitoring approaches . New monitoring approaches can be established in the track of refining the prevailing methods. The new directions for the researchers to develop new methods to the monitoring of edge/fog computing are as described below:

- Management of movement:- The application QoS can decrease and increase rapidly as per the movement of the client device and due to the changing network parameters of the connection between the edge node and the end-users such as bandwidth, delay, and jitter etc. There should be some enhancements to the dynamic service migration methods so that it will be easy to predict the future services(in advance). The nodes should be able to house the growing demand for the service distribution and growing network traffic volume. The edge node should ensure the availability of the services.
- Monitoring data management: For large scale environs, the monitoring investigations will generate massive amount to data collection. Which has to be aggregated, processed and stored afterward.A certain mechanism has to built to reduce these data as they will also consume the bandwidth of the network.
- Decentralization Co-ordination: -The biggest challenge for building decentralized systems is to guar-

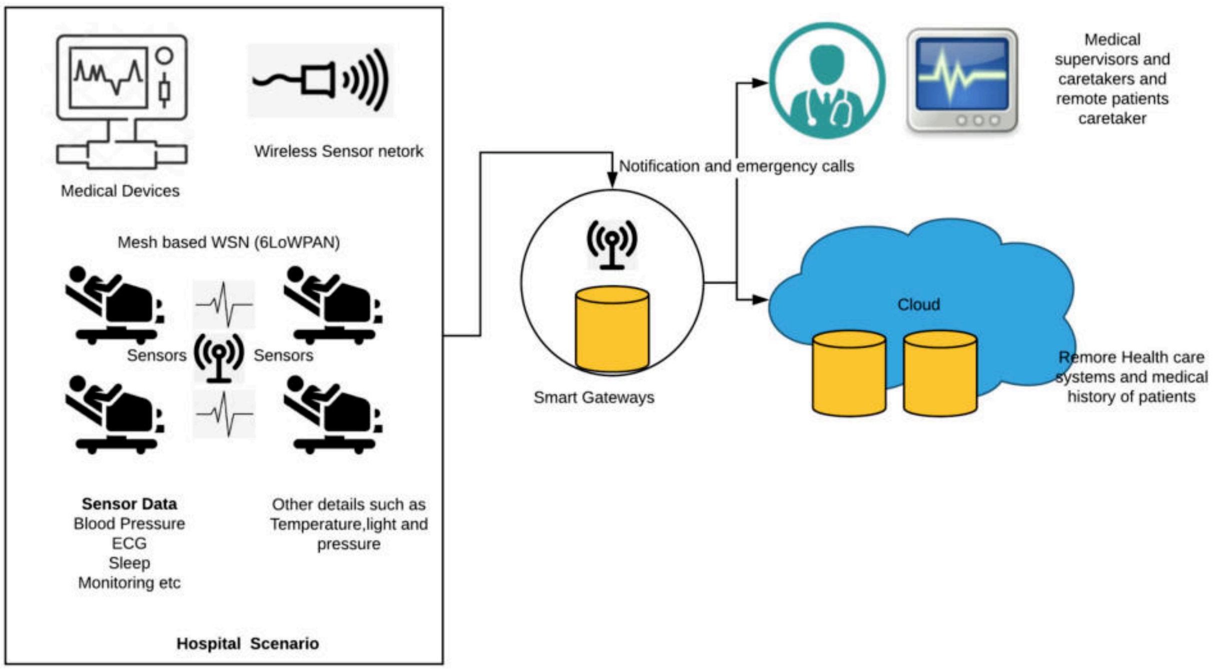

Medical Devices
Wireless Sensor netork
Mesh based WSN (6LoWPAN)
Sensors
Sensors
Sensor Data
Blood Pressure
ECG
Sleep
Monitoring etc
Other details such as Temperature,light and pressure
Hospital Scenario
Notification and emergency calls
Smart Gateways
Medical supervisors and caretakers and remote patients caretaker
Cloud
Remore Health care systems and medical history of patients

Fig. 6.1. *Fog and edge based health monitoring system*

antee that dissimilar application components jointly move the whole systems towards to a common approach/goal.

- Peak resource scheduling: the scheduling approach should be intelligent enough to identify the responses upon the uncertainties of the runtime environs as in the case of varying workload, users mobility and network conditions.
- Peak resource scheduling: the scheduling approach should be intelligent enough to identify the responses upon the uncertainties of the runtime environs as in the case of varying workload, users mobility and network conditions.
- Service Replication: With respect the edge/fog computing, the services can be replicated in cloud infrastructure in multi-zone/geographical locations. Replication of the machines has issues such as inconsistencies which need to be handled carefully and when to do replication is an important factor which needs to identify to save time and performance.

The other non functional aspects for monitoring of edge computing are Robustness,interoperability, non-intrusiveness, scalability and live migration suport, used for handling the applications which are running into the corresponding infrastructure of the edge.

6. Case study: Fog and edge computing for IoT based health monitoring systems. Internet of Things is accepted for any organization who wants to make their systems intelligent in real time scenario and its implementation in medical healthcare [21] is proved to be worthy. The architecture for the same has been depicted in Figure 6.1. The gateway is situated at the edge network and this performs the basic necessity function like translating between the protocols. This normally acts as a hub between the cloud service and the ubiquitous devices such as sensor stratum and proved to be performing better in medical healthcare systems [22]. Where this can perform certain important functions such as identifying the location and mobility of the end users. The valuable features can be reinforced in the edge or gateway such as involving intelligence, empowering networking and connect with the adequate power supply. Fog computing in design and architecture of healthcare

The architecture consists of four components and has been discussed below

- Sensor data related to the medical field and action network: The ubiquitous devices sense and perform the corresponding action by communicating signals from patients body to the control room via the certain usages of the protocols such as 6LowPAN [23], wifi [24], and Zigbee [25].
- Smart e-health network gateway [26]: This results from the different geographically distributed gateways of intelligent gateways, and forming the fog. This receives the data from different networks, usages protocols and provide other higher level of detailed services like reduction in the dimensions, data aggregation, and filtering.
- Systems backend: This includes the platform for cloud computing which implements Dataware housing and analytics of the data collected. These data later can be used for taking important decisions and for medical research epidemiology.
- Features and properties for smart e-health at fog level [27]: In this case, the local pre-processing of the sensors data will be done to do interpretations with respect to medical healthcare and leads to smart e-health gateway.The other important and supportive featues aspects related to the fog are:
  - Processing of the local information
  - Filtering of the information
  - Compression of the information
  - Information fusion
  - Information Analysis
  - Adaptability
  - Local storage
  - Actuation at the local level
  - Security
  - Reconfigurability and technical interoperability
  - Syntactic and semantic interoperability

7. Conclusion. To simplify the process of self-adaptive and decision-making mechanism in the edge/fog computing environs, monitoring clarifications gather information from diverse levels. In addition to monitoring the virtual resources such as disk, memory, and CPU, it is also important to consider the supplementary levels of monitoring together with the container, network quality, and tasks requirement. The functional and non-functional monitoring requirements need to be analyzed for adaptive application for edge/fog computing.The paper also discussed the open challenges for monitoring in edge and fog. The paper ended with the case study of monitoring fog and edge computing with respect to e-health systems and shows that monitoring resources will manage the tasks associated to the health-care data properly or optimally as this includes the various data flow from the processes,data acquisition of sensor nodes and the cloud with the end users.

REFERENCES

[1] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[2] Lu Tan and Neng Wang. Future internet: The internet of things. In *2010 3rd international conference on advanced computer theory and engineering (ICACTE)*, volume 5, pages V5–376. IEEE, 2010.

[3] KP Saharan and Anuj Kumar. Fog in comparison to cloud: A survey. *International Journal of Computer Applications*, 122(3), 2015.

[4] KP Saharan and Anuj Kumar. Fog in comparison to cloud: A survey. *International Journal of Computer Applications*, 122(3), 2015.

[5] Manas Kumar Sanyal, Biswajit Biswas, Subhranshu Roy, and Sajal Bhadra. A proposed model to integrate business intelligence system in cloud environment to improve business function. In *Information Systems Design and Intelligent Applications*, pages 282–292. Springer, 2018.

[6] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 78:680–698, 2018.

[7] Yang Xu, Marco Cello, I-Chih Wang, Anwar Walid, Gordon Wilfong, Charles H-P Wen, Mario Marchese, and H Jonathan Chao. Dynamic switch migration in distributed software-defined networks to achieve controller load balance. *IEEE Journal on Selected Areas in Communications*, 37(3):515–529, 2019.

[8] Bo Yi, Xingwei Wang, Keqin Li, Min Huang, et al. A comprehensive survey of network function virtualization. *Computer Networks*, 133:212–262, 2018.

[9] Demetris Trihinas, George Pallis, and Marios D Dikaiakos. Admin: Adaptive monitoring dissemination for the internet of things. In *IEEE INFOCOM 2017-IEEE conference on computer communications*, pages 1–9. IEEE, 2017.

[10] Adrien Lebre, Jonathan Pastor, Discovery Consortium, et al. *The DISCOVERY Initiative-Overcoming Major Limitations of Traditional Server-Centric Clouds by Operating Massively Distributed IaaS Facilities*. PhD thesis, Inria, 2015.

[11] Ion-Dorinel Filip, Florin Pop, Cristina Serbanescu, and Chang Choi. Microservices scheduling model over heterogeneous cloud-edge environments as support for iot applications. *IEEE Internet of Things Journal*, 5(4):2672–2681, 2018.

[12] Seong-Min Kim, Hoan-Suk Choi, and Woo-Seop Rhee. Iot home gateway for auto-configuration and management of mqtt devices. In *2015 IEEE Conference on Wireless Sensors (ICWiSe)*, pages 12–17. IEEE, 2015.

[13] Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data*, pages 37–42. ACM, 2015.

[14] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 78:680–698, 2018.

[15] Salman Taherizadeh, Andrew C Jones, Ian Taylor, Zhiming Zhao, and Vlado Stankovski. Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review. *Journal of Systems and Software*, 136:19–38, 2018.

[16] Hakiri Akram and Aniruddha Gokhale. Rethinking the design of lr-wpan iot systems with software-defined networking. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 238–243. IEEE, 2016.

[17] Dazhong Wu, Shaopeng Liu, Li Zhang, Janis Terpenny, Robert X Gao, Thomas Kurfess, and Judith A Guzzo. A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing. *Journal of Manufacturing Systems*, 43:25–34, 2017.

[18] HAN Kui-kui, XIE Zai-peng, and LV Xin. Fog computing task scheduling strategy based on improved genetic algorithm. *Computer Science*, (4):22, 2018.

[19] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. Fog computing: A taxonomy, survey and future directions. In *Internet of everything*, pages 103–130. Springer, 2018.

[20] Salman Taherizadeh, Andrew C Jones, Ian Taylor, Zhiming Zhao, and Vlado Stankovski. Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review. *Journal of Systems and Software*, 136:19–38, 2018.

[21] Amir M Rahmani, Tuan Nguyen Gia, Behailu Negash, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, and Pasi Liljeberg. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, 78:641–658, 2018.

[22] Ammar Awad Mutlag, Mohd Khanapi Abd Ghani, N Arunkumar, Mazin Abed Mohamed, and Othman Mohd. Enabling technologies for fog computing in healthcare iot systems. *Future Generation Computer Systems*, 90:62–78, 2019.

[23] Tuan Nguyen Gia, Imed Ben Dhaou, Mai Ali, Amir M Rahmani, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen. Energy efficient fog-assisted iot system for monitoring diabetic patients with cardiovascular disease. *Future Generation Computer Systems*, 93:198–211, 2019.

[24] Chien-Ming Li, Yueh-Ren Ho, Wei-Ling Chen, Chia-Hung Lin, Ming-Yu Chen, and Yong-Zhi Chen. Nasogastric tube dislodgment detection in rehabilitation patients based on fog computing with warning sensors and fuzzy petri net. *Sensors and Materials*, 31(1):117–130, 2019.

[25] Tuan Nguyen Gia, Imed Ben Dhaou, Mai Ali, Amir M Rahmani, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen. Energy efficient fog-assisted iot system for monitoring diabetic patients with cardiovascular disease. *Future Generation Computer Systems*, 93:198–211, 2019.

[26] Amir-Mohammad Rahmani, Nanda Kumar Thanigaivelan, Tuan Nguyen Gia, Jose Granados, Behailu Negash, Pasi Liljeberg, and Hannu Tenhunen. Smart e-health gateway: Bringing intelligence to internet-of-things based ubiquitous healthcare systems. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 826–834. IEEE, 2015.

[27] Vladimir Stantchev, Ahmed Barnawi, Sarfaraz Ghulam, Johannes Schubert, and Gerrit Tamm. Smart items, fog and cloud computing as enablers of servitization in healthcare. *Sensors & Transducers*, 185(2):121, 2015.

Edited by: Anand Nayyar
Received: Mar 16, 2019
Accepted: Apr 25, 2019

DOI 10.12694/scpe.v20i2.1530
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 377–398. http://www.scpe.org © 2019 SCPE

DATA PLACEMENT IN ERA OF CLOUD COMPUTING: A SURVEY, TAXONOMY AND OPEN RESEARCH ISSUES

AVINASH KAUR [*], POOJA GUPTA [†], MANPREET SINGH [‡] AND ANAND NAYYAR [§]

Abstract. In cloud computing, data placement is a critical operation performed as part of workflow management and aims to find the best physical machine to place the data. It has direct impact on performance, cost and execution time of workflows. Number of data placement algorithms is designed in cloud computing environment that aimed to improve various factors affecting the workflows and their execution including the movement of data among data centers. This paper provides a complete survey and analyses of existing data placement schemes proposed in literature for cloud computing. Further, it classifies data placement schemes based on their assess capabilities and objectives. Further objectives and properties of data placement schemes are compared. Finally future research directions are provided with concluding remarks.

Key words: Data placement, Cloud computing, workflow, replication

AMS subject classifications. 68M14, 68M20

1. Introduction.

With the increase in involvement of data in scientific workflows , the need for high end computing and large amount of storage increses. The processing of data is a very complex task.For a scientific workflow relatively tremendous data is to be stored in different data centers [1]. Data centers are facing unpredictable amount of visitors with the generation of data and development of Internet.This leads to the explosive increase in storage capacity [2]. The large amount of scientific data is accumulated by various fields of research such as meteorology, astronomy and bio-informatics. Processing of large scale data and obtaining valuable scientific discoveries is a tedious job.This lead to the rising need of high performance computing. [3]. This high performance computing,also known as cloud computing entails the need of mass storage resources.

The prime process in a cloud computing environment is during the implementation and execution of data intensive applications for meeting the demand of large volumes of data and storing the data at appropriate data center .This case arises the need of data placement strategy [4]. Data placement includes all activities related to movement such as replication,staging,transfer, space de-allocation and space allocation, unregistering and registering meta data, retrieving and locating the data.

There arises a question how the data placement is performed on cloud computing as in Figure 1.1.

The clients work on a data, evaluates it and passes it to the server.The server processes the request of the client.This is to and fro operation and at the end final output is to be stored on the storage device.This storage of output onto a storage device is decided by the data placement strategy.

1.1. Data Placement in cloud.

In the cloud computing era ,storage of data achieves terabytes magnitude scale ,diverse and complex data structures, high requirement of type and level of service have induced great pressure to data management.Cloud systems work on two kinds of applications ,data-intensive and compute intensive .These are handled concurrently in the systems and generates massive ammount of output/intermediate data.The cloud architecure that handles data and workflows is as in Figure 1.1.

The cloud architecture is classified into four layer architecture as shown in Figure 1.1.

- **Application layer:** In this layer the workflows are constructed and requirements are submitted by graphical and textual user interfaces to the system. The re-usability and flexibility of cloud service components and local resource component is provided by modular programming. The workflows are then changed to predefined mathematical models.These models are used in the next layer. This layer makes available many visualizations and presentation functions for calculating the results of applications.
- Service Layer: The scientific workflows are executed in this layer. For proper functioning of workflow management system , it deals with the fault tolerance and monitoring of workflows. The requirements

[*]Lovely Professional University, Phagwara, India (avinash.14557@lpu.co.in).
[†]Lovely Professional University, Phagwara, India (pooja.19580@lpu.co.in).
[‡]Guru Nanak Dev Engineering College, Ludhiana, India (pmpreet78@gmail.com).
[§]Graduate School, Duy Tan University, Da Nang, Viet Nam (anandnayyar@duytan.edu.vn).

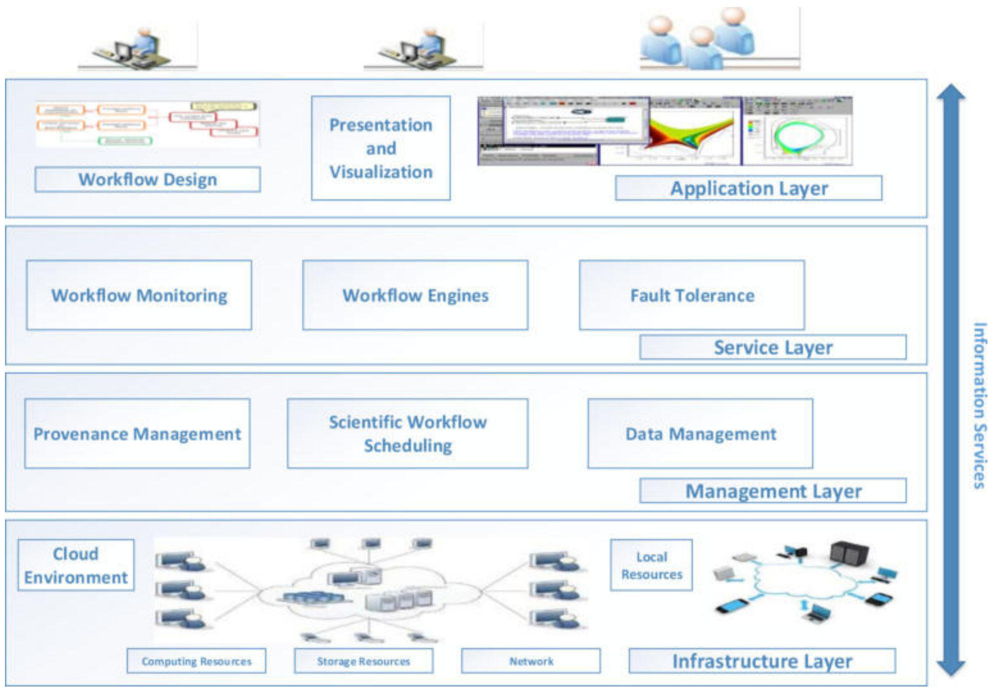


FIG. 1.1. *Workflow management system.*

are received from application layer.Then, the scheduling information is obtained from management layer. The separation of execution of the workflow from the task scheduling and task acquisition leads to the improvement in scalability and stability of the management system.

- Management Layer: It is the bridge of workflow execution and physical resources. The major part of this layer is module of scheduling the scientific workflows that determines which scheduling algorithm to use and it also provided different strategies for scheduling. It is responsible in optimizing the task procedures and parallelizing the processing of data. Before planning the scheduling it obtains the resources and application information about the available services of system. The movement of data is also involved during the scheduling process which is then needed to be handled by efficient data management module.
- Infrastructure Layer: It is responsible for expansion of forgoing services platform which is designed by adding local resources based on the cloud environment. If the local resources are used more ,then it saves the costs. Each computing environment includes network,storage, computing, etc. [5]

1.2. Need of Data Placement.

- Management of budget for storage: Some scientists perform their task without considering the budget whereas there are some that consider budget as one of the QoS parameter.So, it depends upon vendor to vendor.These kinds of vendors prefer the cloud service providers that adhere to the QoS parameters provided by them [6, 7].
- Effective distribution of data onto a storage device: There is a large variety of storge devices available with different parameters like storage capacity,transmission speed etc. So making the data evenly and considerably distributed among these storage devices is also an important task [4].
- Data placement of data intensive jobs is major challenge in cloud environment. Improper data placement policy increases data movement in the data centre which leads to increased cost and delayed services.
- Placement of runtime datasets is still a challenge in complex scientific workflow. Storage and computation capacity is bottleneck at some situations while transferring the intermediate datasets.

- At best of my knowledge and as per the literature review done, no placement policy considers the cached dataset (e.g. fixed position datasets) which affects the performance at significant level.

1.3. Related surveys. The two researchers [8] and [9] have done an inventive review of literature in the field of data placement. Still in the field of data placement research is persistently growing. For the integration and evaluation of existing research present in the data placement field there lies a need of a methodical survey of literature. This paper presents a survey in methodical form for discovering and evaluating research challenges on the basis of existing research in the field of data placement.

2. Background. With the drastic increase in data , requirements for both commercial and scientific applications are expected to reach several million terabytes scales. The matter of concern is about the increasing I/O needs and also about number of users accessing the same dataset.In various fields like genomics and biomedical more people will be accessing more datasets.The movement of large amount of data for replication and processing becomes the necessity.This brings a problem of reliable and efficient data placement.There is requirement of locating the data,moving the data to the place where it is required ,replicating and staging the data.Then allocating and de-allocating the storage and at the end cleaning up everything is required. As the scheduling and managing of computational and network resources is an important task ,similarly scheduling the data placement activities is also crucial.

2.1. Introduction to Cloud Computing. In 1960 , John McCarthy imagined that computation in certain time would be given as a utility . The acknowledgement of this is Cloud computing.The path of a cloud computing environment is being laid by technologies such as Web Services,Hardware Virtualization,Service Oriented Architecture and Mashups [31].

Cloud computing has emerged from various computing paradigms of distributed computing over the growth of both hardware and internet technology. Seamlessly using the power of virtualization and distribution, cloud today provides the use of applications and services over the internet as if they are being used on the local machine. A cloud computing stage is made out of broadly disseminated set of equipment stage, which is arranged and running assorted software services [32].

Most of us are unaware of the use of cloud, the use of e-mails as in gmail, social networking sites as when a picture is uploaded on facebook we use the services offered by or hosted on a cloud. Users do not need to have any information about the background services and can make a communication with many servers at the same time and also there is communication among servers themselves also. Cloud computing aims in providing services to the users that hosts documents on the internet, to outsource the IT infrastructure.

2.2. Workflow. The Workflow Management Coalition (WfMC) defined workflow as : The automation of a business process, in whole or part during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [33]. By the definition given by WfMC, work processes is a progression of organized exercises and calculations of a business procedure which is a unique depiction of the undertakings to be executed in that business process. They are utilized to join a few diverse computational procedures into a solitary lucid procedure. The business applications can now be seen as perplexing work processes, which comprise of different changes performed on the information fancied in accomplishing the goal.Workflows offer awesome points of interest in isolating capacities from applications and in this manner offering data framework to be segmented based by first arranging them and then incorporating them.

WfMC introduces its reference model in recognizing the interfaces inside of this structure which empower items to operate interactively at an assortment of levels. It characterizes a work management framework and the most critical interfaces of system (see Figure 2.1).

- **Workflow Engine:** A software service that gives the run-time environment with a specific end term goal to make, oversee and execute work process cases.
- **Process Definition:** Specifies the information about the process and the workflows related to it.
- **Workflow Interoperability:** This interface makes interoperability possible between different processes of workflow.
- **Invoked Applications:** Interfaces to bolster cooperation with an assortment of IT applications.
- **Workflow Client Applications:** Interfaces to bolster connection with the client interface.

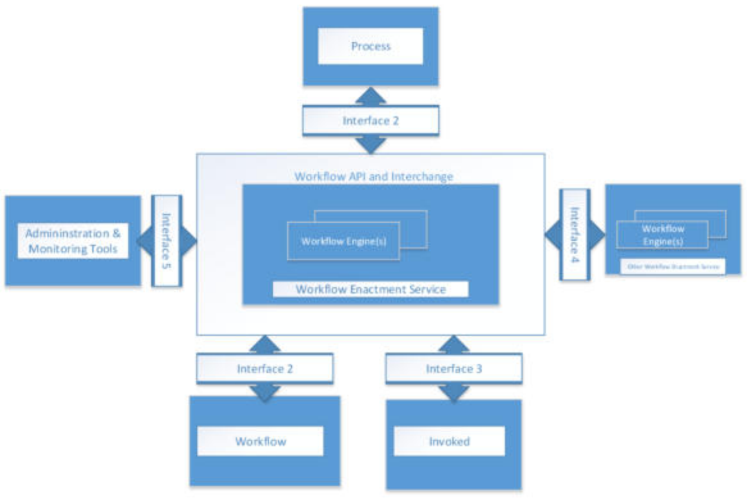


Fig. 2.1. *WfMC reference model.*

- **Administration and Monitoring:** Interfaces to give a framework observing and metric capacities to encourage the administration of composite work process application situations.

Work process has developed from the idea appropriated programming and with a specific end goal to suite different programming styles and complex shell scripts. Numerous work process frameworks have been created for and productively sent on distributed computing infrastructures that offer high performances such as peer to peer computing,cluster computing and grid computing. Because of the a lot of calculation and information included, the force of the disseminated figuring is required to execute these work processes effectively.

Workflows have found their use in scientific applications,analysis of complex applications and optimization of the execution time for similar applications. In specific conditions researcher might do not have the earlier learning to execute and run the applications, or they do not have the required asset frameworks. The accessibility of the assets and their heterogeneity makes it perplexing and hard to be taken care of by an experimental application.Such application workflows require high performance computing infrastructures for executing data and compute intensive activities. To accommodate such requests large, expensive computing infrastructure are bought and installed. Their integration and maintenance involves huge costs, still the scalability of the resources is a bigger problem as the resources might not be available during the peak hours and they cannot be utilized by others during the off peak periods.

Such issues are not covered by cluster and grid computing paradigms, but is handled effectively by Clouds. Cloud offers a resource pool which can dynamically scaled and managed for delivering high computing power and performance. The services are provided on demand of the users over the internet depending on their deployed applications.

2.3. Workflow lifecycle. The phases in the life cycle of workflow are design, partitioning or clustering ,mapping and enactmentas shown in Figure 2.2.

In the design phase of workflow scientific or business workflows are designed by executable languages. In the initial stage of creation of workflow with the definition of goals, high level workflow model is designed. In a decentralized architecture model is deployed and process of clustering or partitioning is defined. The method of partitioning creates fragments of workflow by grouping elements such as control flows,data flows,activities etc. In clustering, the different tasks are clustered together at the same level, horizontal level or vertical level. In the mapping phase, these clustered or partitioned tasks are deployed in virtual machines or the engines of workflow. The enactment phase of workflow us responsible for implementation of clustered tasks or partitions. In this research, the focus is on mapping phase of workflow.

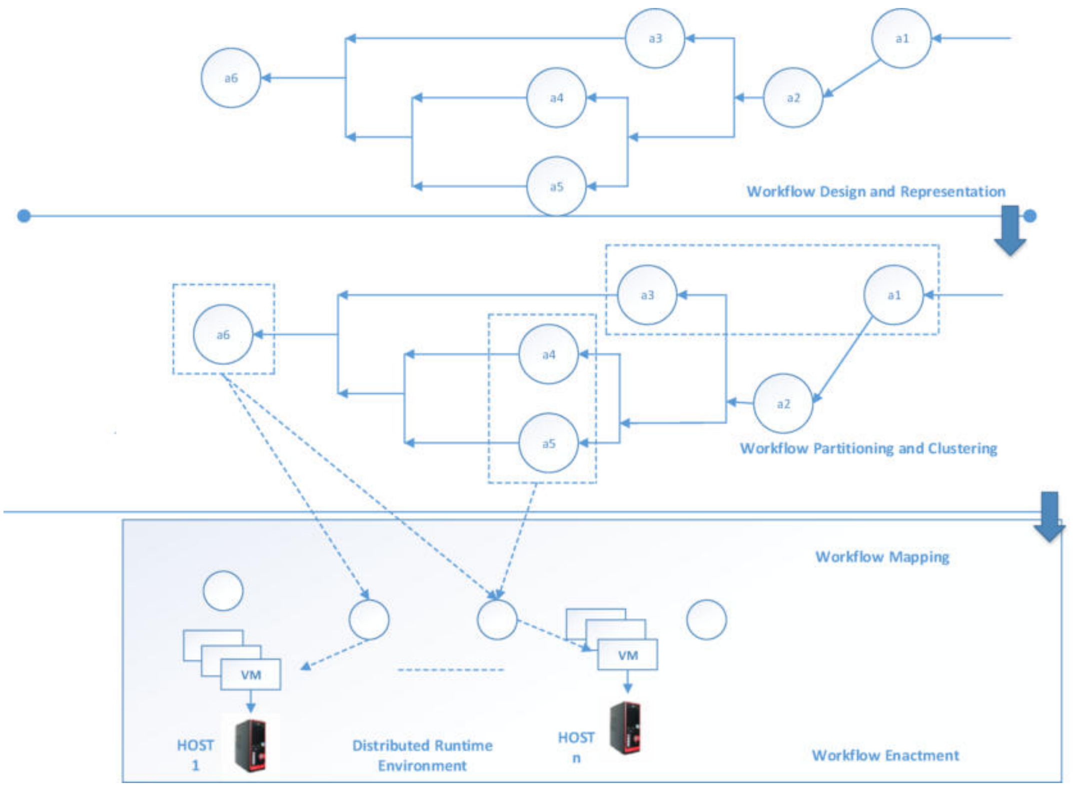

a1
a2
a3
a4
a5
a6
Workflow Design and Representation
Workflow Partitioning and Clustering
Workflow Mapping
VM
VM
HOST 1
Distributed Runtime Environment
HOST n
Workflow Enactment

FIG. 2.2. *Workflow lifecycle.*

2.3.1. Workflow design phase. It is one of the important phase of the lifecycle of workflow. It is divided into Workflow model composition ,application type and interdependency model as in Figure 2.3.

Workflow Model composition. This is responsible to enable user of workflow provide the dependencies and steps in the workflow. The tasks or jobs in the workflow are defined as different steps and the order in which these steps can be executed is defined as dependency. The model composition of workflow can be accomplished using either graph based or language based modeling [55]. Extensible markup language(XML) is used to express the workflow in language based modeling. The control flow is defined by using block structures similar as in C language, such as while and if.So memorizing the syntax of language is required.It becomes tedious task for the users to express complex or large workflow manually using language based modeling.

The control flow of process in graph based modeling is defined using explicit control links between activities. In this workflow is defined graphically using some graph elements.As the graphical representation is easier ,so it is mostly used by the users.The graph based modeling can be further categorized in non-directed acyclic graph(NDAG) or directed acyclic graph(DAG) [56]. In a DAG, the workflow is specified as

$$W = (Ta, E)\ where\ Ta = ta_1, ta_2, ..., ta_n \tag{2.1}$$

is a finite set of tasks specified by the vertices's in DAG and E specifies its existing edges. The dependency of data and communication is represented by an edge [57].In a DAG based workflow ,control patterns are complemented by non_DAG iteration structure that initializes cycle or loop. The scientific application comprises the iteration structure and it requires cyclic execution of tasks. There are different workflow models [55].

Workflow Application Types.

1. Data/Computation intensive workflows: Scientific workflows are responsible for the visualization of complex scientific computations.These are commonly used in e-science technology and applications.

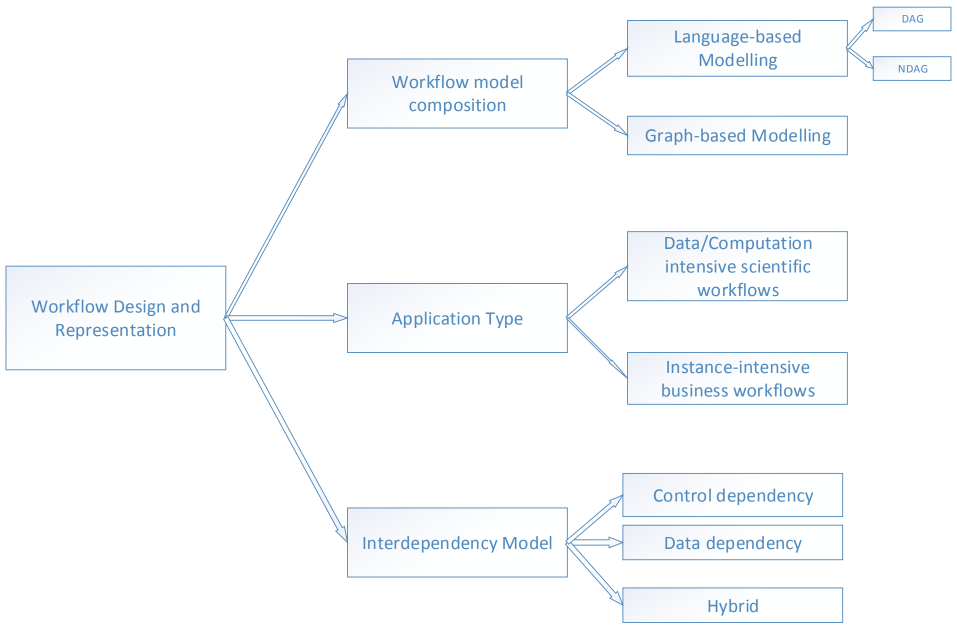

Workflow Design and Representation
Workflow model composition
Language-based Modelling
DAG
NDAG
Graph-based Modelling
Application Type
Data/Computation intensive scientific workflows
Instance-intensive business workflows
Interdependency Model
Control dependency
Data dependency
Hybrid

Fig. 2.3. *Workflow design phase.*

Various computing resources are involved in scientific workflows in de-centralized environments that is used in various areas of bioinformatics,astronomy etc. [58]

2. Instance-Intensive Business Workflow: The models of workflow are applied in business also. Business workflow composes of documented processes taking place in the routine business of an organization. A new workflow is introduced upon further detailing e-business and e-science application ,that is instance intensive workflow. For example, e-business is one of the example of instance intensive workflows [59].The instance of workflow requires parallel execution [61]. The various examples of e-business are insurance claim, bank check etc. [60]
   The basis for instance intensive workflow is data or computation intensive workflow.For example, scientific workflow for weather forecast can be used as commercial weather forecasting product .

Interdependency Model. The type of dependencies existing in the partitions of workflows is defined as interdependency. Data dependency and control dependency are both used for defining different types of dependencies.The control transfer between the tasks or partitions is defined by a link that is applicable to control structures such as conditionals, loops and sequences. Data flows or dependencies supports data-intensive applications. The data transfer between tasks is specified by data dependency. Hybrid models combines both the data and control dependencies for better management of models of workflow [62].

2.3.2. Workflow partitioning. After the workflow is designed in the life cycle of workflow ,then the workflow is partitioned or clustered into number of partitons. These clustered or partiotioned tasks are further sent to the mapping phase of the workflow.

2.3.3. Workflow Mapping. Workflow mapping is responsible for creation of executable instance of workflow that uses suitable resources as in Figure 2.4. Resource in a workflow is defined as agent/service/computer/ person that can execute task or fragment of workflow.

The mapping of workflow is performed either by the user manually or by a workflow system automatically.In the mapping done by the user each partition is assigned to one execution engine.Kepler and Taverna workflow systems are examples of these kinds of mapping The other kind of mapping is assigning the available resources to partitions/tasks using the internal schedulers. The workflow systems employing this type of mapping are

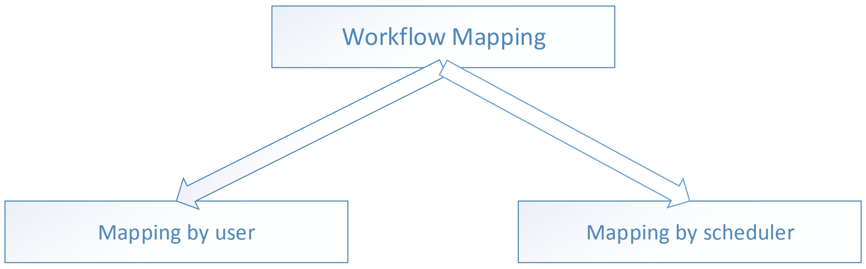


Fig. 2.4. *Workflow Mapping Phase.*

Triana ,Karajan and Pegasus [64]. The grid application toolkit (GAT) is used by Triana for choosing resources at runtime [65]. This research focuses on data placement so it described further in complete detail.

2.3.4. Workflow Enactment. In this phase the fragments or partitions are enacted in Grid or Cloud environment as on figure 6.It is described as a service responsible for executing and managing workflows ,also maintaining coordination between resources and management tools required for execution of workflow [33].The workflow engine is kind of software service that provide executable environment for instance of workflow.Workflow enactment is classified into further two categories Business workflow engines and scientific workflow engines as depicted in Figure 2.5.

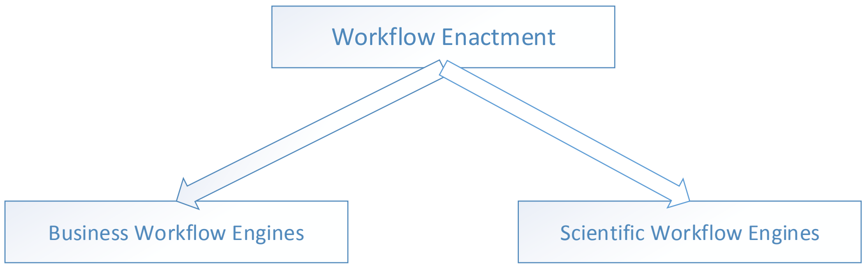


Fig. 2.5. *Workflow Enactment Phase.*

The emergence of Cloud computing introduced economy specific pay-per-use model . The Cloud is the biggest pool of virtual resources such as platform, services and hardware , making workflow management cheaper,easier and faster.There exists many kinds of workflow enactment engines. The enactment engines used in business workflow domain are NINOS [66] , SwinDeW-C(Swinburne decentralized workflow for cloud) [61], etc. and the workflow engines for scientific workflows are Kepler, Taverna etc.

2.4. Data Placement Process. In the computing systems,data placement process is defined as movement of input data of data intensive applications from remote site to the execution site, then a movement of output data from execution to another remote site or same site.To prevent the risk of disk full at the execution site, there is a requirement to allocate space and deallocating the space when job is done.

The data placement steps are presented as DAG (Directed Acyclic Graphs) and dependencies between them are represented by arc. The steps of data placement are as shown in Figure 2.6.

2.4.1. Data Placement Stages.

- Stage-in : This data placement stage is also known as build-time data placement.This involves getting the provenance information of the dataset and pre-clustering the similar data items before uploading the data to the appropriate data center. [13]

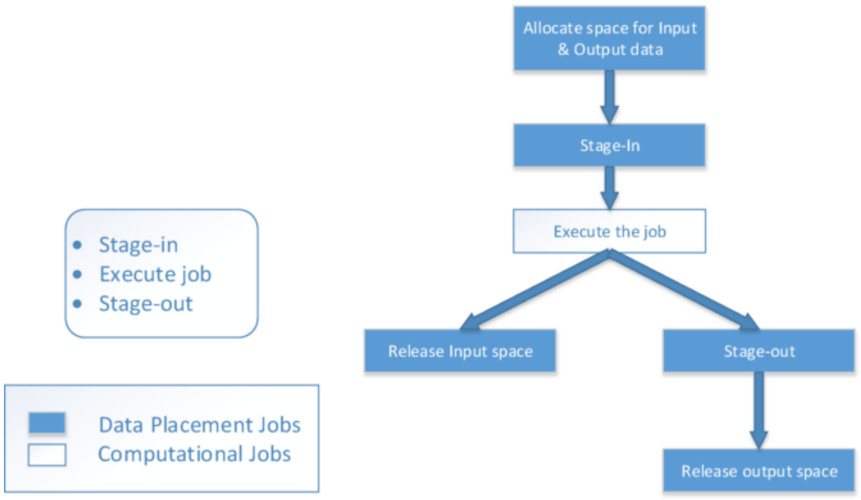


Fig. 2.6. *Data Placement Process.*

- Stage-out: This data placement stage is also known as runtime data placement.During the execution of workflows intermediate data is generated .This data may be the input for the subsequent tasks,so placing and managing this kind of data is again a complicated task. [13]

3. Data Placement Algorithms. The data placement algorithms are classified into different categories of algorithms as shown in Figure 3.1.

3.1. Coorelation Based data placement. A Data placement scheduler ,Stork provides an ability to queue,schedule,manage and monitor the data placement jobs.It also applies the technique of checkpointing the jobs.This approach provides a complete automation for processing the data. The proposed system automate the data transfer fully automatically between the hetrogenous systems.It possesses the ability to recover from network,software and storage system failures without human intervention.It performs dynamic adaption of data placement jobs at the execution time to system environment [34]

The author presents a filecule grouping technique for managing data in science Grids while saving the time for locating the data and grouping the file based on size.Also LRU-bundle an algorithm for file staging is proposed [35]. [36] introduced a BitDew programming interface for data management operations as replication,placement and fault tolerance.This architecture relies on independent services to transfer ,store and schedule the data.

In [29] the investigator proposed an algorithm for run time as well as for build time stage.In the initial stage dependency between all the data sets is calculated and dependency matrix is built.For the partitioning and clustering of data sets BEA(Bond energy algorithm) is used.These partitions are shared among different data centers.These data centers are partitioned using k means algorithm.After the generation of intermediate data ,the newly proposed clustering algorithm deals with new datasets.The dependencies for each data center are judged and then accordingly data is moved.The factors of data movement and gathering of data at one point is covered up.Two algorithms proposed are as follows:

1. Build-Time Stage Algorithm:In this stage dependencies between the tasks is calculated and dependency matrix is built.For the transformation of dependency matrix to clustered dependency matrix ,BEA is used.Global measure(GM) for clustered dependency matrix is defined

$$GM = \sum_{i=1}^{n}\sum_{j=1}^{n} DM_{ij}(DM_{i,j-1} + DM_{i,j+1}) \tag{3.1}$$

Then further partiotined datasets are mapped to data center using binary partiotining algorithm as per

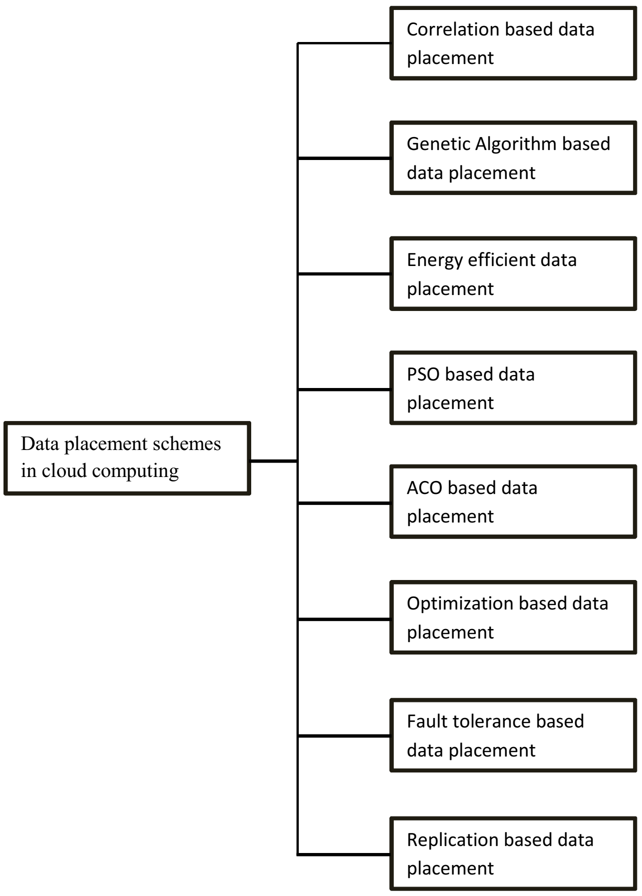

Data placement schemes in cloud computing
Correlation based data placement
Genetic Algorithm based data placement
Energy efficient data placement
PSO based data placement
ACO based data placement
Optimization based data placement
Fault tolerance based data placement
Replication based data placement

FIG. 3.1. *Data Placement Algorithms.*

equation

$$PM = \sum_{i=1}^{p}\sum_{j=1}^{p} CM_{ij} * \sum_{i=p+1}^{n}\sum_{j=p+1}^{n} CM_{ij} - \left(\sum_{i=1}^{p}\sum_{j=p+1}^{n} CM_{ij}\right)^2 \tag{3.2}$$

This process continues recursively.

2. Run-Time Stage Algorithm: In this stage , newly generated datasets are clustered to the data center on the basis of dependency between them using K means algorithm.The dependency for all datasets with all data centers is calculated.At last the dataset placement is performed depending on the maximum dependency on a data center.

In [9] author proposed DCCP(Dynamic Computation correlation placement) as a data placement technique based on dynamic computation correlation.The data sets with highly dynamic correlation are placed at same data center.The factors considered are capacity load and Input/Output load of data centers.During the execution of computations ,data sets processed are stored on local data centers thus leading to the reduction of execution time.This leads to Input/Output and statistical load balancing.

3.2. Genetic Algorithm based data placement. In [30] author proposed algorithm for reduction of movement of data among data centers leading to load balancing in data centers.The heuristic and genetic are combined together for improving the ability to local search and reducing the search time.The heuristic idea is implemented in gene operations and initial population selection.The integer encoding rules are applied and

Table 3.1
Objectives of correlation based data placement schemes

Schemes	Energy aware	Cost aware	Resource aware	Application aware	Factors
[35]	-	✓	-	-	Data dependency,File Size
[36]	-	✓	✓	✓	Data Transfer
[34]	-	✓	-	✓	Data dependency
[29]	-	✓	-	-	Data dependency ,Data Movemement, Execution Time
[9]	-	✓	-	-	Load balance ,Execution Time

Table 3.2
Properties of coorelation based data placement schemes

Schemes	SLA support	Security	Cloud/ Grid	Algorithm	Clustering Algorithm	Tool/Programming Language
[35]	✓	-	Grid	Greedy Request Value	-	Java
[36]	✓	-	Grid	Information Dispersal	-	Java
[34]	✓	-	Grid	-	-	Stock Server
[29]	✓	-	Cloud	Bond Energy	K means	SwinDeW-C
[9]	-	-	Cloud	DCCP	-	cloud computing application platform (CApp)

placement process is represented by a gene.The ineffective fragments of genes that cannot be coded at data centers is determined by the encoding rule.Fitness function depicts the advantages and disadvantages of genetic individuals.It depicts degree of dependency between data sets and load balance of centers.D_dc is the total degree of data dependency in a data center d_c, which is defined as:

$$D_{d_c} = \sum_{ds_i, ds_j in d_c} Dij \tag{3.3}$$

The fitness function is defined as

$$fitness = \frac{1}{|DC_u|}\left(\sum_{dc \epsilon DC_u} D_{dc}\right) \tag{3.4}$$

where DC_u is data centers set which consist of at least one data set.Larger the value depicts better gene. The overloading of data centers are adjusted by non normal chromosome.If the gene is found to be invalid ,it refers to overloading of data centers.So this gene is to be neglected.If the invalidity is depicted at initial stage then the new gene should be generated.If this occurs at cross stage then genes to be re-crossed and if it occurs at mutation stage then genes to be re-mutated.Optimization of genes is performed for reducing movement of data.If the data set is fixed location data set then calculate the dependency of each data set with data center as per formula .If the size of data set is more then the storage capacity of data center then data set is moved to next data center.

In [28] investigator proposed a mathematical data placement technique on the basis of genetic algorithm.The crossover rate(P_c), size of population (G) and mutation rate(P_m) is determined.After the generation of initial population BG , fitness value for of each individual is obtained. Fitness value of is denoted by $F = 1/\Gamma(B_t)$. At last individuals are selected using roulette wheel selection [25].Crossover and mutation are performed on the selected matrix. [26, 27]. The individuals not adhering to the requirements of storage capacity are abandoned.

[37] also introduced technique of data placement based on a genetic algorithm for reducing the time as well as the cost of data scheduling between centers.

Authors considered the factors of dependency among slices of data and distributed cost of transaction for the placement of data using genetic algorithm.It minimizes the transaction cost while balancing the load.Author

TABLE 3.3
Objectives of Genetic Algorithm energy based data placement schemes

Schemes	**Energy aware**	**Cost aware**	**Resource aware**	**Application aware**	**Factors**
[30]	-	✓	✓	✓	Data movement,Load balance
[28]	-	✓	✓	-	Storage Capacity,Data Transfer
[37]	-	✓	✓	-	Data dependency,Data movement,Cost

TABLE 3.4
Properties of Genetic algorithm energy based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/ Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/Programming Language**
[30]	✓	-	Cloud	Heuristic genetic	-	Cloudsim
[28]	-	-	Cloud	Optimal genetic	-	-
[37]	-	-	Cloud	Hashing	K-Means	-

used roulette wheel for realization model.The fitness function is represented as

$$Fitness = \frac{\sum_{DCT_n} C_{ij}}{|DCT|} \tag{3.5}$$

Where $|DCT|$ are number of data center, DCT_n the data center numbered n and C_{ji} is the total cooperation cost between data sets j and i

3.3. Energy Efficient data placement. In [38] [39] [40] [41] [42] many authors proposed energy saving methods but at hardware level such as processor speed adjustment,voltage settings,enlarging memory, etc.But these methods are unable to reach maximum energy optimization as saving of energy by these methods is comparitively less then turning the computer off.These methods are only limited to a single PC or computer.

In [21]author proposes heuristic algorithm for data placement and two node scheduling techniques in order to save consumption of energy during execution of tasks.In the proposed algorithm data blocks are kept rational for finding minimum set of nodes containing the collection of blocks of data.The goal of energy saving is achieved by turning on minimum nodes required covering the maximum data blocks.Greedy algorithm is employed for covering data block with computing node.The author addresses two goals in it .First is the power consumption upper bound is known,accordingly execution time of task requests by node scheduling is minimized.Second ,the execution time of tasks is known, accordingly execution time of task requests by node scheduling is minimized.This optimization of batch scheduling achieves energy saving effect by providing a solution for time constrained and power restrained issues in the platform of cloud.Each node $s \epsilon S$,in node s, q is the number of data blocks , p is the number of data blocks which meet the job requests , so node cover rate

$$\gamma(s) = \frac{p}{q} \tag{3.6}$$

If node s and node s_t have the same data block replica, then node s and node s_t are data-exchangeable. The formula used to compute resource utilization of nodes is

$$U = e.U_{cpu} + (1 - e).U_{disk} \tag{3.7}$$

where U_{cpu} stands for utilization of cpu ,u_{disk} stands for utilization of disk and e is scale factor. [21]

In [43] investigator proposes Popular Data Concentration (PDC) algorithm .It dynamically migrated mostly used data on the disk to different subset of disks in an array.The purpose is to transfer the load to few disks so that other disks can be sent to mode of energy saving.In [44] author addresses conservation fo energy for cluster of nodes that execute Map Reduce jobs.Reconfiguration of clusters is performed on the basis of current workload.When the average utilization rises, clusters are turned on or off.

Table 3.5
Objectives of Energy efficient based data placement schemes

Schemes	**Energy aware**	**Cost aware**	**Resource aware**	**Application aware**	**Factors**
[21]	✓	-	✓	-	power consumption,Execution time
[43]	✓	✓	-	-	File characterstics,workfload characterstics
[44]	✓	-	-	-	power consumption,cost
[45]	✓	✓	-	✓	Execution cost,power consumption

Table 3.6
Properties of Energy efficient based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/ Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/Programming Language**
[21]	✓	-	Cloud	Dynamic Algorithm	-	Cloudsim
[43]	-	-	File server	Popular data concentration	-	Execution Driven simulator
[44]	-	-	Grid	Cluster reconfiguration	-	Gridsim
[45]	-	-	Cloud	EnCloud algorithm	-	iVIC with Phython

In [44] author proposed energy saving algorithms that switch off some number of nodes for saving energy.In [45] investigator proposed EnaClous approach that enables dynamic live application placement considering the energy efficieny.In this approach ,virtual machine is used for encapsuation of an application that perform scheduling of an application and live migration of an application for saving energy.Bin packing algorithm is used for apllication placement.The author proposes energy aware heuristic algorithm.For dealing with the varying resource demands ,investigator presented an over provision technique.

In this [46] author proposed an architectural framework based on resource allocation and principles for energy efficient Green cloud computing.It introducesa virtual node placement scheme and decides to turn on some physical nodes of the active virtual nodes, but data placement problem is not considered.

3.4. PSO based data placement. In [**?**] near optimal assignment of tasks in reasonable time is obtained using proposed hybrid particle swarm optimization approach.PSO based algorithm is presented for conquering TAP(Task Assignment Problem).PSO iteration is embedded with hill climbing heuristic for convergence.The factors considered are execution and communication costs.

In [7] author proposed an algorithm based on particle swarm optimization that leads to reduction of cost and execution time of transferring.The investigator works on on-demand method as standard for computing.This refers to paying per on per hour basis with no long term commitments.According to charging standard of Amazon.P_{out_k} is defined as pricing of data center from DC_k. p_{in_t} as pricing of data transfer to DC_l. P_k is the processing pricing of standard on-demand and T_p is the execution time of tasks.The i represents a task executing on data center k.Cost of data processing is given by

$$C_p = T_p \times P_k \tag{3.8}$$

Cost of data transfer is represented as

$$C_t = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{l \neq k} x_{ik} \times x_{jl} \times (DT_{ij} \times P_{out_k} + DT_{ij} \times P_{in_l}) \tag{3.9}$$

Fitness function is sum of cost of data transfer and processing represented as

$$C = C_p + C_t \tag{3.10}$$

According to the best fitness value ,data is placed at the data center.

Table 3.7
Objectives of PSO based data placement schemes

Schemes	**Energy aware**	**Cost aware**	**Resource aware**	**Application aware**	**Factors**
[?]	-	✓	-	-	Execution and Communication cost
[7]	-	✓	-	-	processing time, transferring time, processing cost, transferring cost
[22]	-	✓	✓	-	Data dependency ,Data Transmission Cost
[17]	-	✓	✓	-	Execution time,Transferring time,Cost
[48]	-	✓	-	-	Execution cost,Communication cost
[8]	-	✓	-	-	Data dependency ,Data Movement

Table 3.8
Properties of PSO based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/ Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/Programming Language**
[?]	✓	-	-	Hybrid-PSO	-	-
[7]	✓	-	Cloud	PSO	-	Matlab
[22]	✓	-	Cloud	PSO	-	-
[17]	✓	-	Cloud	PSO-SPV	-	Cloudsim
[48]	✓	-	Cloud	PSO	-	-
[8]	-	-	Cloud	PSO	Bond Energy(BEA)	Visual Studio 2010

In [22] author proposed cost effective data placement scheme for multi data centers.During the execution of workflows , the data sets get transferred from one data center to another.So mapping of these data sets to an appropriate data center is the issue addressed by the author. The focus is on reducing the cost of data transfer by considering multiple workflows.Let $DC = \cup_{i=1,2,...|DC|} dc_i$be data center set. G_i is a single workflow. Datasets of multiple workflow(MWS) are represented as $DS = \cup_{i=1,2,...n} ds_i$ Data placement of multiple workflow $MWS = \cup_{i=1,2,..n} G_i$ is represented by

$$M_{multi} = \cup_{i,j=1,2,....,|DS|} d_i \rightarrow dc_j \tag{3.11}$$

where $dc_j \epsilon DC \forall d_i \epsilon DS$,$d_i$ is stored in unique data center in M_multi.

In [17] investigator introduced Particle swarm optimization based algorithm embed in SPV(Small Position value) [47].The aim is to optimize task scheduling in cloud computing for minimization of processing cost.The algorithm is based on a small position value.The focus is on reducing execution and transferring time.The fitness function for an algorithm is taken as combination of cost and time $CT = T + C$ [7].

In [48] article investigator presents particle swarm optimization(PSO) based heuristic for scheduling applications to resources of cloud.The factors of data transmission cost and computation cost is considered.The mapping of tasks to resources is performed by varying the computation and communication costs.

In [8] investigator clustered datasets on the basis of dependency using proposed hierarchical clustering based technique.A new factor as size of dataset is introduced.This leads to reduction in data movement ,In the partitioning matrix improved dichotomy algorithm is used.For mapping between data groups and servers PSO based algorithm is used.

3.5. ACO based data placement. In [67], author presents data placement scheme for intermediate data placement by considering the factor of security.The security of data is based on aspects of data integrity,authentication access and data confidentiality. A model of security quantitatively measures services of security provided by data center.Ant Colony Optimization (ACO) based technique is used for dynamic selection of data center.In this the datasets are divided into flexible location and fixed location data sets according to location of stored data.For the evaluation of security services, DDSD (Degree of data security deficiency) is proposed.

Table 3.9
Objectives of ACO based data placement schemes

Schemes	**Energy aware**	**Cost aware**	**Resource aware**	**Application aware**	**Factors**
[67]	-	-	✓	-	data security,data transfer time
[22]	-	✓	✓	-	Data dependency ,Data Transmission Cost
[17]	-	✓	✓	-	Execution time,Transferring time,Cost
[48]	-	✓	-	-	Execution cost,Communication cost

Table 3.10
Properties of ACO based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/ Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/Programming Language**
[67]	✓	✓	Cloud	ACO	-	Cloudsim
[22]	✓	-	Cloud	PSO	-	-
[17]	✓	-	Cloud	PSO-SPV	-	Cloudsim
[48]	✓	-	Cloud	PSO	-	-

3.6. Optimization based data placement. In the [49] proposed system considers the factors of WAN bandwidth ,data center capacity limit,data interdependency,user mobility and application changes.The proposed approach analyzes the logs submitted by the data centers.It uses iterative optimization algorithm based on access patters of data ,locations of clients.Further ,it outputs recommendations of migration back to cloud service.For scaling large volume of data of data logs ,this technique works in SCOPE [14]. This algorithm works in three phases.Mapping of each client to a set of geographical candidates is performed using weighted spherical mean calculation as per equation

$$wsm((w_i, \vec{x}_i)_{i=1}^{N}) = interp\left(\frac{w_n}{\sum w_i}, \vec{x}_N, wsm(w_i, \vec{x}_i)_{i=1}^{N-1}\right) \tag{3.12}$$

It handles the complexities of data inter-dependencies and shared data.In the second phase ,improvement in the placement is done by applying proposed approach .In the third phase, mapping of data to data center is performed by considering the factor of storage capacity of the data center.

In [68] author aims to place data files into and assigning tasks to the sites of execution for reducing the cost while considering weights.To accomplish this , workflow is modelled as a hypergraph.A heuristic based on hyper-graph partitioning is proposed for generating appropriate placement of data and assignment of task . According to the proposed technique, computational and storage loads are distributed evenly according to some pre-decided ratios.The hypergraph partitioner is implemented by modifying PaToH [15].

In multilevel framework,net costs are incorporated to PaToH in three phases. In the first phase of coarsening of net costs if performed.In the phase of initial partitioning the algorithm modification of GHGP [15] is performed to be used with target weights and net costs.In the last phase of refinement ,heuristic FM [16] is modified for accurate calculation of the vertex move gains and cutsize.

In [29] investigator proposed an algorithm for each build time and run time stage.In the initial stage, dependency between all the data sets is calculated and dependency matrix is built.For the partitioning and clustering of data sets BEA(Bond Energy Algorithm) is used.These partitions are distributed among different data centers.The data centers are partitioned using k means algorithm.After the generation of intermediate data ,the newly proposed clustering algorithm deals with new datasets.The dependencies for each data center are judged and then accordingly data is moved.The factors of data movement and gathering of data at one point is covered up.This strategy allocate application data among data centers automatically and reduces the movement of data.

In [50] the investigator propose non linear programming model(NLP) for minimization of retrieval of data and execution cost of workflows.The proposed technique aims at minimizing the computation cost and data transfer cost on the compute resource.For case study,intrusion detection application is considered.While using

Table 3.11
Objectives of optimization based data placement schemes

Schemes	Energy aware	Cost aware	Resource aware	Application aware	Factors
[49]	-	-	✓	-	storage capacity, data interdependency, WAN bandwidth, IP address
[68]	-	✓	✓	-	Communication cost
[29]	-	✓	-	-	Data dependency,Cost
[48]	-	✓	-	-	Execution cost,Communication cost
[50]	-	✓	-	-	execution cost
[?]	-	✓	✓	-	execution cost, execution time, data dependency

Table 3.12
Properties of Optimization based data placement schemes

Schemes	SLA support	Security	Cloud/ Grid	Algorithm	Clustering Algorithm	Tool/Programming Language
[49]	✓	-	Cloud	iterative optimization	Cloud	-
[68]	-	-	Cloud	combinatorial algorithm	-	C programming language
[29]	✓	-	Cloud	Bond Energy	K-Means	SwinDeW-C
[48]	✓	-	Cloud	PSO	-	-
[50]	✓	-	Cloud	NLP	-	Amazon CloudFront

the storage and compute resources ,NLP model is applied on intrusion detection application.

3.7. Fault tolerance based data placement. In [19] investigator proposed the strategy based on clustering algorithm of consistent hashing and minimum distance.The data is clustered efficiently by the clustering algorithm by placing item based and user based data .For addressing effect of outliers and noises ,cluster centers and threshold is updated.CBR strategy [18] and item based algorithm CF is used to fill sparse matrix of users.Consistent hashing algorithm is used for improving fault toleration and scalability.

A Data placement scheduler ,Stork provide an ability to queue,schedule,manage and monitor the data placement jobs.It also applies the technique of checkpointing the jobs.This approach provides a complete automation for processing the data. The proposed system automate the data transfer fully automatically between the heterogenous systems.It possessess the ability to recover from network,software and stoarge system failures without human intervention.It performs dynamic adaption of data placement jobs at the execution time to system environment [34]

3.8. Replication based data placement. In [51] investigator present a heuristic based on ordering jobs cleaned up in workflow and reduce time taken for execution.Also genetic algorithm based approach vary the amount of storage and number of processors for generating schedules with low cost.Tightly coupled data staging approach is used.As per the model introduced DPS and workflow manager work together for fulfilling the requirements of data management.

In [53] investigator proposed CDRM(Cost effective dynamic replication management) that captures the relations between replica number and availability.It aim to increase availability of data for storage system of cloud for balancing load and improving performance while considering failures.CDRM instincts proposed model for maintaining minimum number of replicas according to requirement availability.Placement of replica is on the basis blocking capacity and probbiility of data nodes.CDRM dynamically redistributes according to changing workload and capacity of node.CDRM is implemented in Hadoop Distributed File System(HDFS).

In [52] author proposed two-level DHT (TDHT) approach for applying Lazy update and minimize communication cost.At first level, trade off on security and availability between TDHT and conventional pure data partitioning approach is performed.This approach is integrated with DHT and is called GDHT(Global DHT) .

[6] The investigator used the replication strategy for achieving availability,reliability and appropriate utilization of network bandwidth.PC cluster system is used for implementation of cloud storage system.

Table 3.13
Objectives of Fault tolerant based data placement schemes

Schemes	**Energy aware**	**Cost aware**	**Resource aware**	**Application aware**	**Factors**
[19]	-	✓	✓	-	Fault tolerance,Execution Time
[34]	-	✓	-	✓	Data dependency,Fault tolerance

Table 3.14
Properties of Fault tolerant based data placement schemes

Schemes	**SLA support**	**Security**	**Cloud/ Grid**	**Algorithm**	**Clustering Algorithm**	**Tool/Programming Language**
[19]	✓	-	Cloud	Consistent Hashing	K means	VMware
[34]	✓	-	Grid	-	-	Stock Server

Author proposed new interconnection network MyHeawood for cost effective data placement.It is based on different hashing functions.It is a hierarchical network.It is composed of small switch and dual NIC server.Data placement strategy is based on the hashing function composed of different hash functions. Hash functions are composed of hash key for computing server address for master replica. After that, on the basis of address of master replica ,remaining replicas are allocated in different layer.Three replicas are used for solving overhead of storage caused by multiple replicas.This solves the issue of reliability in cloud. [24]

Further work on replication is improved by [54].

In [23] author introduced a Balanced and file Reuse-Replication Scheduling (BaRRS) algorithm for cloud computing.This algorithm divides the scientific workflow into multiple workflows for balance utilization through parallelization. It deals with reuse of data and different replication techniques for optimization of transferable data.It considers execution time of task,patterns of dependency among tasks and size of files for adapting to current techniques of replication and data reuse.At last it selects the optimal solution on basis of monetary cost and execution time.

4. Results. The main motivation of this research work is to find all the available research in data placement of scientific workflows . In all, 64 research articles are published in prominent journals, symposiums, workshops and in foremost conferences on cloud computing. Most of the research articles on data placement algorithms are published in comprehensive variety of journals and conference proceedings.We perceived that conferences like IEEE/ACM , Future Generation Computer systems, Proceedings of the 4th generation workshop on workflows in support of large-scalescience, Encyclopedia of parallel computing: Springer , Proceedings of fourth international workshop on data intensive distributed computing, Proceedings of VLDB Endowment, Parallel and Distributed Computing international symposium, Cluster Computing and Grid 8th IEEE international symposium, Proceedings of 17th international symposium on high performance distributed computing, Grid computing environments workshop, International Conference on cloud computing 2009, Future Information Technology : Springer, International Journal of High Performance Computing Applications, Procedia Environmental Sciences, The Computer Journal, Cloud Computing and Distributed System Laboratory, International Conference on Super Computing, Computing in science and Engineering, Journal of Grid Computing, Computer standardsand interfaces, Journal of Electrical and Computer Engineering, Chinese Journal of Computers contribute significantly to our review area. Figure 8 shows the percentage of research paper discussing different data placement algorithms (Coorelation, genetic, Energy Efficient, PSO based, ACO based, Optimization Based, Fault Tolerance based, Replication Based) from year 2008 to 2017. Summary table for comparison of data placement algorithms is shown in Table 4.1.

Figure 4.2 depicts the maximum research in Replication based(17%) and PSO based(17%) data placement while very few research in fault tolerance based data placement(6%) and genetic based data placement(9%). Coorelation based and Optimization based contributes 14% each while Energy Efficient, ACO based and Genetic based contributes 12%, 11% and 9% respectively.

Table 3.15
Objectives of Replication based data placement schemes

Schemes	Energy aware	Cost aware	Resource aware	Application aware	Factors
[51]	-	✓	-	-	Makespan,execution cost,storage limit
[34]	-	✓	-	✓	Data dependency, Fault tolerance
[53]	-	✓	-	-	CPU power, memory capacity, network bandwidth, access latency, load balance
[52]	-	✓	-	-	Response Latency,Cost
[24]	-	✓	-	✓	Cost
[23]	-	✓	-	✓	Execution time,Data dependency,Task Size

Table 3.16
Properties of Replication based data placement schemes

Schemes	SLA support	Security	Cloud/ Grid	Algorithm	Clustering Algorithm	Tool/Programming Language
[51]	-	-	Grid	Coarse and fine grained genetic algorithm	-	Cluster viz16
[34]	✓	-	Grid	-	-	Stock Server
[53]	✓	-	Cloud	CDRM	-	-
[52]	✓	✓	cloud	uTLA	-	PlanetLab (Plab) platforms
[24]	-	-	Cloud	Hashing	-	Cloudsim
[23]	-	-	Cloud	BaaRS	-	VMware-ESXi-based (version)

4.1. Data Placement implication platforms. For the implementation of the data placement algorithms as in figure 4.2 , 79% of the algorithms are implemented on cloud while 17% on Grid and 4% on file servers. For the simulation of cloud environment, the tools used are in Figure 4.2.

5. Conclusion and Future Directions. Cloud computing is an emerging computing platform that presents different schemes for the large amount of data arising from scientific workflow application.Data placement is one of the necessary schemes that make cloud computing possible.Recenetly data placement problem has become a hot topic in the area of cloud computing while considering the scientific workflow applications because it can greatly reduce the execution time of workflows and increase the efficiency of data centers.

This paper presents a comprehensive survey and analysis of the data placement schemes proposed for cloud computing.For this purpose, first the definition of data placement schemes are provided, and then the classification of data placement schemes is presented based on the type of placement.Then various proposed data placement schemes are described and factors of data placement in each scheme are investigated to illuminate the advantages, limitations and properties of each placement scheme in detail. Moreover, complete comparative comparisons of the data placement schemes are presented which highlight the items that should be considered in future studies and researches. As specified in these comparisons, most data placement schemes are aimed to improve the performance and execution related issues in data centers.There is less focus on security related issues in data placement.But with the ever increasing security attacks on services of cloud,providing security in the cloud environment has become a critical issue. As a result, security is one of the crucial issue which should be considered in the future data placement.

As per the literature survey there is scope of further improvement in data placement in cloud.Following section describes further improvements in this field.

1. For improving the inter data center traffic ,identification of nearby sites can be pursued that leads to improved latency and costs.The location of data center can be provided as an input.In future more placement factors can be considered such as computation capacity of each server and load balancing.Replicating the used data can also be used for performance in terms of system response time and

Table 4.1
Comparison of different data placement algorithms on placement criteria

Schemes	Security	Storage Capacity	Network Bandwidth	Task Characterstics	Data dependency	Fault Tolerance	Transfer Time	Transfer Cost	Execution Time	Data Movement
Coorelation Based Data placement										
[35]	-	-	-	✓	✓	-	-	-	-	-
[36]	-	-	-	-	-	-	-	✓	-	-
[34]	-	-	-	-	✓	-	-	-	-	-
[29]	-	-	-	-	✓	-	✓	-	✓	✓
[9]	-	-	-	✓	✓	-	-	-	✓	-
Genetic Algorithm Based Data placement										
[30]	-	-	-	-	-	-	-	-	-	✓
[28]	-	✓	-	-	-	-	-	-	-	✓
[37]	-	-	-	-	✓	-	-	✓	-	✓
Energy Efficient based Data placement										
[21]	-	-	-	-	-	-	-	-	✓	-
[43]	-	-	-	✓	-	-	-	-	-	-
[44]	-	-	-	-	-	-	-	✓	-	-
[45]	-	-	-	-	-	-	✓	✓	-	-
PSO Based Data placement										
[?]	-	-	-	-	-	-	-	✓	✓	-
[7]	-	-	-	-	-	-	✓	✓	✓	✓
[22]	-	-	-	-	✓	-	-	✓	-	-
[17]	-	-	-	-	-	-	✓	✓	✓	-
[48]	-	-	-	-	✓	-	-	✓	✓	-
[8]	-	-	-	-	✓	-	-	-	-	✓
ACO Based Data placement										
[67]	✓	-	-	-	-	-	✓	-	-	-
[22]	-	-	-	-	✓	-	-	✓	-	-
[17]	-	-	-	-	-	-	-	✓	✓	✓
[48]	-	-	-	-	-	-	-	✓	✓	-
Optimization Based Data placement										
[49]	-	✓	✓	-	✓	-	-	-	-	-
[68]	-	-	-	-	-	-	-	✓	-	-
[29]	-	-	-	-	✓	-	-	✓	-	-
[48]	-	-	-	-	-	-	-	✓	✓	-
[50]	-	-	-	-	-	-	-	-	✓	✓
[?]	-	-	-	-	✓	-	-	-	✓	-
Fault Tolerance Based Data placement										
[19]	-	-	-	-	-	✓	-	-	✓	-
[34]	-	-	-	-	✓	✓	-	-	-	-
Replication Based Data placement										
[51]	-	✓	-	-	-	-	-	✓	✓	-
[34]	-	-	-	-	✓	✓	-	-	-	-
[53]	-	✓	✓	-	-	-	-	-	-	-
[52]	-	-	-	-	-	-	-	✓	✓	-
[24]	-	-	-	-	-	-	-	✓	-	-
[23]	-	-	-	✓	✓	-	-	-	✓	-

reliability.

2. The data sets are of two types fixed location and variable location data sets. The fixed location datasets are the one that are located in a predetermined locations in the cloud environment and are non transferable whereas the variable location datasets can be moved from one place to another. There are works listing the use of variable location datasets, but for enhancement fixed location datasets can

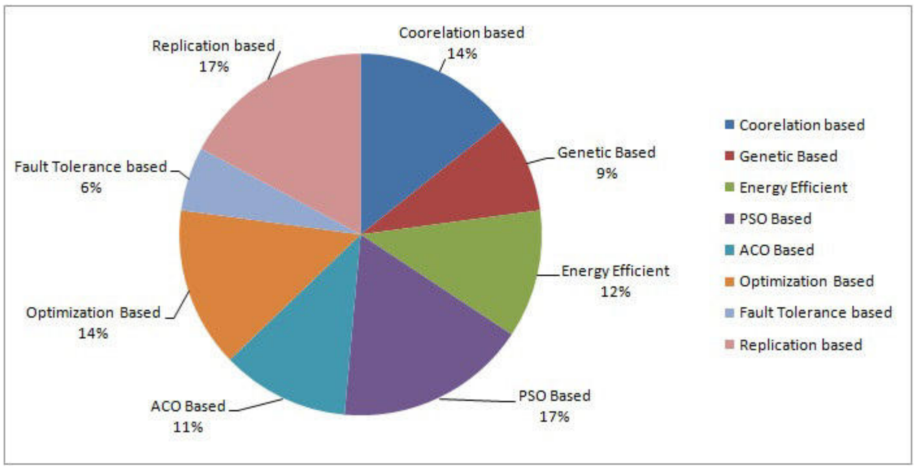


Fig. 4.1. *Data Placement Algorithms in Cloud.*

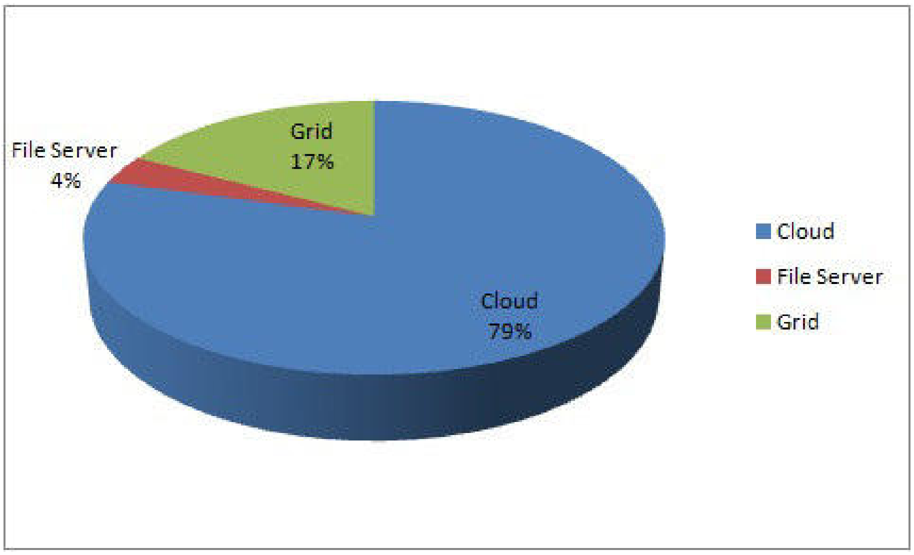


Fig. 4.2. *Platform for Algorithms.*

also be considered for further improvement of cost,price etc.

3. The system performance of cloud can also be improved further by detailing the characteristics of tasks that can further be used for improvement of scheduling mechanism in a heterogeneous environment.Further policies can be implemented on the basis of characteristics of tasks and system with different policy schemes can be implemented.
4. The cost of running the workflow and the energy consumption can be analyzed .The further investigation is required on the issue for the minimization of power consumption that will lead to improvement of performance in scientific workflows.The data placement can be explored by considering the energy/power efficiency and power/performance trade-offs.
5. In future more placement factors can be considered such as computation capacity of each server and load balancing.Replicating the used data can also be used for better performance in terms of system reliability and response time.
6. Different methods of genetic selection affect the performance of the algorithm which requires further study.Fuzzy strategies are not outlined.The granularization approach most suitable for the entire work-

load and optimal storage location of each of the resulting granular,sub workloads can be determined.

REFERENCES

[1] Ewa Deelman and Ann Chervenak. Data management challenges of data-intensive scientific workflows. In *Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on*, pages 687–692. IEEE, 2008.

[2] Alexandros Labrinidis and Hosagrahar V Jagadish. Challenges and opportunities with big data. *Proceedings of the VLDB Endowment*, 5(12):2032–2033, 2012.

[3] John J Rehr, Fernando D Vila, Jeffrey P Gardner, Lucas Svec, and Micah Prange. Scientific computing in the cloud. *Computing in science & Engineering*, 12(3):34–43, 2010.

[4] Han Ning Wang, Wei Xiang Xu, and Chao Long Jia. A high-speed railway data placement strategy based on cloud computing. In *Applied Mechanics and Materials*, volume 135, pages 43–49. Trans Tech Publ, 2012.

[5] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[6] Julia Myint and Thinn Thu Naing. A data placement algorithm with binary weighted tree on pc cluster-based cloud storage system. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 315–320. IEEE, 2011.

[7] Lizheng Guo, Zongyao He, Shuguang Zhao, Na Zhang, Junhao Wang, and Changyun Jiang. Multi-objective optimization for data placement strategy in cloud computing. In *International Conference on Information Computing and Applications*, pages 119–126. Springer, 2012.

[8] Qing Zhao, Congcong Xiong, and Peng Wang. Heuristic data placement for data-intensive applications in heterogeneous cloud. *Journal of Electrical and Computer Engineering*, 2016, 2016.

[9] Tao Wang, Shihong Yao, Zhengquan Xu, and Shan Jia. Dccp: an effective data placement strategy for data-intensive computations in distributed cloud computing systems. *The Journal of Supercomputing*, 72(7):2537–2564, 2016.

[10] Ewa Deelman, James Blythe, Yolanda Gil, and Carl Kesselman. Pegasus: Planning for execution in grids. *GriPhyN*, 20:2002, 2002.

[11] Ian Foster, Jens Vockler, Michael Wilde, and Yong Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *Scientific and Statistical Database Management, 2002. Proceedings. 14th International Conference on*, pages 37–46. IEEE, 2002.

[12] Fran Berman, Geoffrey Fox, and Anthony JG Hey. *Grid computing: making the global infrastructure a reality*, volume 2. John Wiley and sons, 2003.

[13] Xin Liu and Anwitaman Datta. Towards intelligent data placement for scientific workflows in collaborative cloud environment. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 1052–1061. IEEE, 2011.

[14] Ronnie Chaiken, Bob Jenkins, Per-Åke Larson, Bill Ramsey, Darren Shakib, Simon Weaver, and Jingren Zhou. Scope: easy and efficient parallel processing of massive data sets. *Proceedings of the VLDB Endowment*, 1(2):1265–1276, 2008.

[15] Ümit Çatalyürek and Cevdet Aykanat. Patoh (partitioning tool for hypergraphs). In *Encyclopedia of Parallel Computing*, pages 1479–1487. Springer, 2011.

[16] Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *Papers on Twenty-five years of electronic design automation*, pages 241–247. ACM, 1988.

[17] Lizheng Guo, Shuguang Zhao, Shigen Shen, and Changyuan Jiang. Task scheduling optimization in cloud computing based on heuristic algorithm. *JNW*, 7(3):547–553, 2012.

[18] Zeina Chedrawy and Syed Sibte Raza Abidi. An intelligent knowledge sharing strategy featuring item-based collaborative filtering and case based reasoning. In *Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on*, pages 67–72. IEEE, 2005.

[19] Qiang Li, Kun Wang, Suwei Wei, Xuefeng Han, Lili Xu, and Min Gao. A data placement strategy based on clustering and consistent hashing algorithm in cloud computing. In *Communications and Networking in China (CHINACOM), 2014 9th International Conference on*, pages 478–483. IEEE, 2014.

[20] Mingjun Wang, Jinghui Zhang, Fang Dong, and Junzhou Luo. Data placement and task scheduling optimization for data intensive scientific workflow in multiple data centers environment. In *Advanced Cloud and Big Data (CBD), 2014 Second International Conference on*, pages 77–84. IEEE, 2014.

[21] Yanwen Xiao, Jinbao Wang, Yaping Li, and Hong Gao. An energy-efficient data placement algorithm and node scheduling strategies in cloud computing systems. In *Proc. of the 2nd Intl Conf. on Advances in Computer Science and Engineering (CSE 2013). Paris: Atlantis Press*, volume 63, 2013.

[22] Xuejun Li, Yang Wu, Fei Ma, Erzhou Zhu, Futian Wang, Lei Wu, and Yun Yang. A new particle swarm optimization-based strategy for cost-effective data placement in scientific cloud workflows. In *Future Information Technology*, pages 115–120. Springer, 2014.

[23] Israel Casas, Javid Taheri, Rajiv Ranjan, Lizhe Wang, and Albert Y Zomaya. A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems. *Future Generation Computer Systems*, 2016.

[24] Xin Huang, Yu Xing Peng, and Peng Fei You. Data placement and query for cloud computing based on myheawood network. In *Applied Mechanics and Materials*, volume 543, pages 3100–3104. Trans Tech Publ, 2014.

[25] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[26] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[27] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic

algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[28] Wei Guo and Xinjun Wang. A data placement strategy based on genetic algorithm in cloud computing platform. In *Web Information System and Application Conference (WISA), 2013 10th*, pages 369–372. IEEE, 2013.

[29] Dong Yuan, Yun Yang, Xiao Liu, and Jinjun Chen. A data placement strategy in scientific cloud workflows. *Future Generation Computer Systems*, 26(8):1200–1214, 2010.

[30] Zhao Er-Dun, Qi Yong-Qiang, Xiang Xing-Xing, and Chen Yi. A data placement strategy based on genetic algorithm for scientific workflows. In *Computational Intelligence and Security (CIS), 2012 Eighth International Conference on*, pages 146–149. IEEE, 2012.

[31] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A taxonomy and survey of cloud computing systems. *INC, IMS and IDC*, pages 44–51, 2009.

[32] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee, 2008.

[33] David Hollingsworth and UK Hampshire. Workflow management coalition: The workflow reference model. *Document Number TC00-1003*, 19, 1995.

[34] Tevfik Kosar and Miron Livny. A framework for reliable and efficient data placement in distributed computing systems. *Journal of Parallel and Distributed Computing*, 65(10):1146–1157, 2005.

[35] Shyamala Doraimani and Adriana Iamnitchi. File grouping for scientific data management: lessons from experimenting with real traces. In *Proceedings of the 17th international symposium on High performance distributed computing*, pages 153–164. ACM, 2008.

[36] Gilles Fedak, Haiwu He, and Franck Cappello. Bitdew: a programmable environment for large-scale data management and distribution. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–12. IEEE, 2008.

[37] Pai Zheng, Li-Zhen Cui, Hai-Yang Wang, and Meng Xu. A data placement strategy for data-intensive applications in cloud. *Jisuanji Xuebao(Chinese Journal of Computers)*, 33(8):1472–1480, 2010.

[38] Meikel Poess and Raghunath Othayoth Nambiar. Tuning servers, storage and database for energy efficient data warehouses. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 1006–1017. IEEE, 2010.

[39] Andreas Beckmann, Ulrich Meyer, Peter Sanders, and Johannes Singler. Energy-efficient sorting using solid state disks. *Sustainable Computing: Informatics and Systems*, 1(2):151–163, 2011.

[40] Lei Rao, Xue Liu, Le Xie, and Wenyu Liu. Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[41] Willis Lang and Jignesh Patel. Towards eco-friendly database management systems. *arXiv preprint arXiv:0909.1767*, 2009.

[42] EN Mootaz Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy-efficient server clusters. In *International Workshop on Power-Aware Computer Systems*, pages 179–197. Springer, 2002.

[43] Eduardo Pinheiro and Ricardo Bianchini. Energy conservation techniques for disk array-based servers. In *ACM International Conference on Supercomputing 25th Anniversary Volume*, pages 369–379. ACM, 2014.

[44] Nitesh Maheshwari, Radheshyam Nanduri, and Vasudeva Varma. Dynamic energy efficient data placement and cluster reconfiguration algorithm for mapreduce framework. *Future Generation Computer Systems*, 28(1):119–127, 2012.

[45] Bo Li, Jianxin Li, Jinpeng Huai, Tianyu Wo, Qin Li, and Liang Zhong. Enacloud: An energy-saving application live placement approach for cloud computing environments. In *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*, pages 17–24. IEEE, 2009.

[46] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.

[47] Jason M Cope, Nick Trebon, Henry M Tufo, and Pete Beckman. Robust data placement in urgent computing environments. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–13. IEEE, 2009.

[48] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, and Rajkumar Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on*, pages 400–407. IEEE, 2010.

[49] Sharad Agarwal, John Dunagan, Navendu Jain, Stefan Saroiu, Alec Wolman, and Harbinder Bhogan. Volley: Automated data placement for geo-distributed cloud services. In *NSDI*, volume 10, pages 28–0, 2010.

[50] Suraj Pandey, Kapil Kumar Gupta, Adam Barker, and Rajkumar Buyya. Minimizing cost when using globally distributed cloud services: A case study in analysis of intrusion detection workflow application. *Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, Melbourne, Australia, Tech. Rep*, 2009.

[51] Shishir Bharathi and Ann Chervenak. Scheduling data-intensive workflows on storage constrained resources. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, page 3. ACM, 2009.

[52] Yunqi Ye, Liangliang Xiao, I-Ling Yen, and Farokh Bastani. Cloud storage design based on hybrid of replication and data partitioning. In *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*, pages 415–422. IEEE, 2010.

[53] Qingsong Wei, Bharadwaj Veeravalli, Bozhao Gong, Lingfang Zeng, and Dan Feng. Cdrm: A cost-effective dynamic replication management scheme for cloud storage cluster. In *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*, pages 188–196. IEEE, 2010.

[54] Suraj Pandey and Rajkumar Buyya. Scheduling workflow applications based on multi-source parallel data retrieval in distributed computing networks. *The Computer Journal*, 55(11):1288–1308, 2012.

[55] Jia Yu, Rajkumar Buyya, and Chen Khong Tham. Cost-based scheduling of scientific workflow applications on utility

grids. In *e-Science and Grid Computing, 2005. First International Conference on*, pages 8–pp. Ieee, 2005.

[56] Hua Huang, Yi Lai Zhang, and Min Zhang. A survey of cloud workflow. In *Advanced Materials Research*, volume 765, pages 1343–1348. Trans Tech Publ, 2013.

[57] Sucha Smanchat and Kanchana Viriyapant. Taxonomies of workflow scheduling problem and techniques in the cloud. *Future Generation Computer Systems*, 52:1–12, 2015.

[58] Jan Hidders, Paolo Missier, and Jacek Sroka. Recent advances in scalable workflow enactment engines and technologies. *Future Generation Comp. Syst.*, 46:1–2, 2015.

[59] Bang Ouyang, Farong Zhong, and Huan Liu. An eca-based control-rule formalism for the bpel process modularization. *Procedia Environmental Sciences*, 11:511–517, 2011.

[60] Rongbin Xu, Xiao Liu, Ying Xie, Futian Wang, Cheng Zhang, and Yun Yang. Logistics scheduling based on cloud business workflows. In *Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on*, pages 29–34. IEEE, 2014.

[61] Ke Liu, Hai Jin, Jinjun Chen, Xiao Liu, Dong Yuan, and Yun Yang. A compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on cloud computing platform. *International Journal of High Performance Computing Applications*, 2010.

[62] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.

[63] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R Pocock, Anil Wipat, et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[64] Gregor von Laszewski and Mike Hategan. Java cog kit karajan/gridant workflow guide. Technical report, Technical Report, Argonne National Laboratory, Argonne, IL, USA, 2005.

[65] Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. Visual grid workflow in triana. *Journal of Grid Computing*, 3(3-4):153–169, 2005.

[66] Guoli Li, Vinod Muthusamy, and Hans-Arno Jacobsen. A distributed service-oriented architecture for business process execution. *ACM Transactions on the Web (TWEB)*, 4(1):2, 2010.

[67] Wei Liu, Su Peng, Wei Du, Wei Wang, and Guo Sun Zeng. Security-aware intermediate data placement strategy in scientific cloud workflows. *Knowledge and information systems*, 41(2):423–447, 2014.

[68] Ümit V Çatalyürek, Kamer Kaya, and Bora Uçar. Integrated data placement and task assignment for scientific workflows in clouds. In *Proceedings of the fourth international workshop on Data-intensive distributed computing*, pages 45–54. ACM, 2011.

[69] Weiwei Chen, Rafael Ferreira, Ewa Deelman, and Rizos Sakellariou. Balanced Task Clustering in Scientific Workflows. pages 1–8.

Edited by: Pijush Kanti Dutta Pramanik
Received: Mar 16, 2019
Accepted: Apr 2, 2019

DOI 10.12694/scpe.v20i2.1537
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 399–431. http://www.scpe.org © 2019 SCPE

RESEARCH ON AUTO-SCALING OF WEB APPLICATIONS IN CLOUD: SURVEY, TRENDS AND FUTURE DIRECTIONS

PARMINDER SINGH [*], POOJA GUPTA [†], KIRAN JYOTI [‡] AND ANAND NAYYAR [§]

Abstract. Cloud computing emerging environment attracts many applications providers to deploy web applications on cloud data centers. The primary area of attraction is elasticity, which allows to auto-scale the resources on-demand. However, web applications usually have dynamic workload and hard to predict. Cloud service providers and researchers are working to reduce the cost while maintaining the Quality of Service (QoS). One of the key challenges for web application in cloud computing is auto-scaling. The auto-scaling in cloud computing is still in infancy and required detail investigation of taxonomy, approach and types of resources mapped to the current research. In this article, we presented the literature survey for auto-scaling techniques of web applications in cloud computing. This survey supports the research community to find the requirements in auto-scaling techniques. We present a taxonomy of reviewed articles with parameters such as auto-scaling techniques, approach, resources, monitoring tool, experiment, workload, and metric, etc. Based on the analysis, we proposed the new areas of research in this direction.

Key words: Cloud computing, resource provisioning, web applications, auto scaling, resource estimation

AMS subject classifications. 68M14, 91C15

1. Introduction.

Cloud computing is emerging technology, which provides processing, bandwidth, and storage as a service. Elasticity is the main characteristic of cloud computing. This technique allocated and de-allocated the resources from the processes as per increment or decrement in requirement. The resource pools are seeming unlimited for the users and can acquire or release the resources anytime [5, 96]. Regular monitoring of giving services is required to ensure the Quality of Service (QoS) and need to fulfill the Service Level Agreements (SLAs). SLA violation leads to the penalty to cloud providers. It is a big challenge for the cloud service providers to provide services within budget and raise profit from datacenters. QoS assures that the behavior of cloud services towards reliability, availability, elasticity, cost, time, etc. [75]. The infrastructure providers offer different pricing policies, companies such as Amazon [131] provides resources for a fixed price per hour. Thus, the application providers should decide the resources for processes, while maintaining the SLAs. In auto-scaling, decision-making techniques to allocate the number of resources to different processes are categorized as reactive and proactive. The reactive technique regularly watches events such as CPU, workload, queue, etc., and performs the elastic operation for resources as per threshold. Proactive forecasting methods used to predict the traffic from the past workload. So far none of the technique is splendid in all the cases [84]. The reasons for ambiguity and diffusion in resource allocation for cloud environment are heterogeneity of resources, dynamic application requirements, and failures. As for now, none of the technique can tackle previously mentioned issues. Autonomic cloud computing can contribute the self-optimization, self-protecting, self-healing, and self-configuration scaling [68].

1.1. Motivation for Research.

- Auto-scaling techniques for cloud application applied to estimate the required resources to process the input requests. Web applications workload is dynamic with sudden burst due to flash workload. This study focused on various auto-scaling techniques for web applications in cloud computing.
- We have recognized the need for detailed survey specifically for the web applications. A methodological survey has been carried out for auto-scaling techniques for web applications. Hence, we summarized the present research challenges and future scope in this area.

1.2. Contribution of the Study.

- A detail investigation has done to study various existing auto-scaling techniques in cloud computing.
- The mentioned techniques classification has been done as per the common characteristics.

[*]Lovely Professional University, India (parminder.16479@lpu.co.in).
[†]Lovely Professional University, India (pooja.19580@lpu.co.in).
[‡]Guru Nanak Dev Engineering College, India (kiranjyotibains@yahoo.com).
[§]Duy Tan University, Da Nang, Vietnam (anandnayyar@duytan.edu.vn).

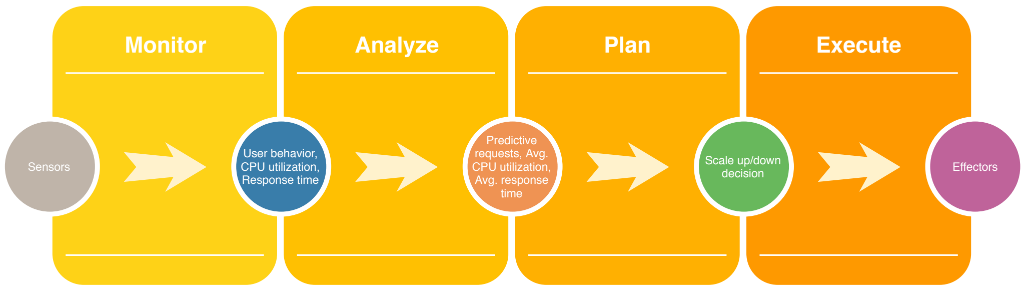

Monitor
Analyze
Plan
Execute
Sensors
User behavior, CPU utilization, Response time
Predictive requests, Avg. CPU utilization, Avg. response time
Scale up/down decision
Effectors

Fig. 2.1. *MAPE Control Lopp System*

- Future research direction in the area of auto-scaling is presented.

1.3. Related Surveys. Earlier surveys have been conducted by the authors [49, 88, 85, 54, 28, 107, 23, 109], but the regular updates in cloud infrastructure and persistent research yielding the new research areas. There is a need to explore the present challenges and new research area in this field. This survey augments the existing studies and recent research articles to describe the research challenges for web applications in cloud computing.

2. Auto Scaling. Auto-scaling is a technique to dynamically adjusts the resources allocated to elastic applications as per the incoming workloads. Auto-scaler in the cloud environment is generic while some are application specific to meet the SLA, QoS and minimizing the cost of scaling. The auto-scaling challenge for the web applications is to dynamically grow or shrink the resources to meet fluctuated workload requirement. Autonomous scaling techniques work without human intervention. Autonomic systems are self-(configuring-optimizing-protecting-healing) [67]. The auto-scaling following the MAPE loop (Figure 2.1): Monitoring (M), Analysis (A), Planning (P) and Execution (E) [92].

- **Monitoring**: The monitoring system collects the information from a cloud environment about the compliance of user expectations, resource status, and SLA violation. It provides the state of infrastructure to the cloud provider, and users get to know about application status with expected SLA. Auto-scaling protocols are decided on the basis of performance metrics for web applications. The author suggested parameters such as resize numbers, operating interval, decision duration, decision threshold, refractory period and instance bounds [45]. Generally metrics provided by cloud providers are related to VM management; otherwise, it will be taken from the operating system. The proxy metrics are used to reduce the complexity of metrics such as hypervisor level and application level (e.g., CPU utilization, workload).
- **Analysis**: The collected information is further processed in the analysis phase. It gathers all information from metrics and current system utilization and prediction information of future workload. Some auto-scaler are working on a reactive approach. The decision is taken after analyzing the current system state. The threshold values are fixed to scale in/out decisions, while others are using a reactive approach or both. Reactive is a sophisticated approach because it's always a delay between the settings of resources for scaling decision. The VM startup time varies from 350 to 400 seconds [90]. Flash crowd and events are still a challenge with the reactive approach.
- **Planning**: Analysis phase evaluates the present state, now the planning phase has to decide to scale up/down or scale in/out to compliance with SLA and profit trade-off.
- **Execution** : Execution phase is already decided in the planning phase. Cloud providers API is responsible for the execution of planning. The client is unaware of the issues in the execution phase. VMs are available to users for a certain period, the startup time of VM takes some time, and these delays have been already discussed with the user in resource SLA.

Auto-scaling techniques have the following research challenges:

Under-provisioning: The application has not sufficient resources to serve all incoming requests from the servers. It may happen due to flash crowd or events, or poor auto-scaling algorithm. This situation leads to

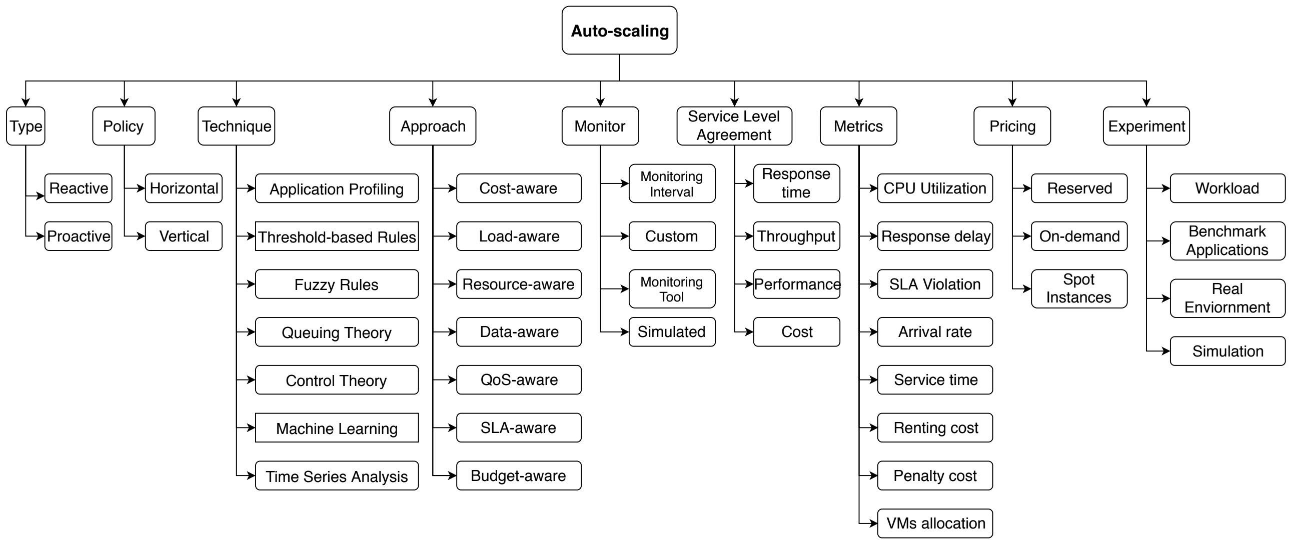


FIG. 3.1. *The taxonomy of web application in cloud computing*

SLAs violation and providers have to pay the penalty to the users, and reliability of the providers also effects. The servers take some time to be back in the normal state.

Over-provisioning: In this situation, the number of resources to process the application is more than the required resources. Service provider's reliability is increased in this case. SLAs violation is minimum in over provisioning and up to a certain level beneficial to handle the fluctuated workload. It affects the profit of the service provider, and on-demand services become costly for the clients. No perfect solution exists either in automatic or autonomous scaling.

Oscillation: It is a pack of both unwanted situations. Rapid resources scaling is taking place without considering the effect on application performance. The condition can be avoided using static and dynamic threshold value fixed for the VMs scaling. A cooling down period is another approach used to handle the oscillation [71].

3. Taxonomy of Auto-scaling. The proposed taxonomy for web application in cloud environment is shown in Figure 3.1. The existing research classified based on the parameters mention in the taxonomy. The taxonomy covered the following points:

- **Type:** Auto-scaler is a crucial component in cloud computing. Auto-scalers are grouped into two categories: Reactive and Proactive. The reactive approaches took the scaling decision by analyzing the current state of the system. Proactive technique analysis the historical data and take scaling decision. Many article formed the new techniques using hybridization of these methods.
- **Policy:** Cloud service providers widely use horizontal scaling as elasticity feature. Cloud providers are providing a fix and customizable resources where the user can configure VMs by specifying memory, cores, bandwidth, etc. Some articles developed the auto-scaling techniques using vertical scaling, where the user can re-configure the VMs resources such as CPU, memory and network bandwidth as per the requirement change. The Centurylink service provider gives the service to scale the CPU cores without vertical downtime.
- **Auto-scaling techniques:** The researcher in the cloud computing used various techniques for analysis and planning phase in MAPE loop to automate the scaling process. In literature, widely used techniques classified in 7 major techniques such as Threshold rules, fuzzy rules, application profiling,machine learning, queuing theory, control theory and time series analysis.
- **Approach:** The newly added feature in the taxonomy is the approach of the investigator on above of the techniques to auto-scale. In the literature, the investigator vision to improve one or more factors such as cost, resource optimization, QoS, SLA violation, etc. Further, the researcher get clear idea

about the articles which improve the specific objectives.

- **Monitoring Tools:** It act as performance indicators. It helps to determine the scaling decisions. Monitoring interval defines the performance of auto-scaler. Balanced performance can be achieved by selecting the right monitoring interval according to the application. Amazon cloudwatch monitoring used by many articles whether some are using custom monitoring tool.
- **Service Level Agreement (SLA):** The service providers need to know the expectations of the customers. It is the master service agreement between the customer and service providers related to various performance outage issues. This document describes the responsibilities of the customer and service providers. It helps the customer to compare the performance among different service providers. The commonly consider performance metrics in SLA related to auto-scaling are response time, budget, cost, throughput, etc.
- **Performance metrics:** It is an essential tool to enhance the reliability of the users moving their applications to the cloud. Difference articles are using various metrics to check the performance of their model such as response time, CPU usage, input request rate, etc.
- **Pricing Model:** Cloud providers are categories the cloud resources in three different pricing models: on-demand, reserved and spot-instances. On-demand resources give performance guarantee to the target application. The number of resources grows or shrink as per the workload change. The reserved resources are a fixed number of resources. Amazon provides the spot instances relatively cheaper than the on-demand resources. Spare capacity instances sell through auction mechanism and user acquire the resources by submitting the bid. Single cloud and multi-cloud resource estimation challenges are different for web applications.
- **Experiment:** The experiment part describes the experiment evaluation in the various articles. The researcher used real time workload or synthetic workload to input the user requests. The benchmarking application are also widely used in the research article for the multi-tier web application. Experiment characteristics considered for the implementation of the model has been reviewed in the taxonomy of auto-scaling. Synthetic and real workload used by an article for the analysis of the model. Cloudsim simulator used by many articles, while some manuscripts used real testbed for the study of the model.

4. Elastic Applications. The elastic applications are dynamic in nature towards change in workload and variables. The load balancer is responsible for the management of elastic applications. Cloud computing applications are managed on VMs, and VMs are managed by the servers. Auto-scaler is a decision maker for the scaling of resources. The system is said to be autonomous because human intervention is not required. Many cloud applications (e.g., Video streaming, web applications) are elastic in nature [85]. Most of the papers about elastic applications considered the web applications as compared to other elastic applications. Web applications consist of three tiers: Presentation, Application, and Database. Most of the articles focused on the application or business tier scaling. The major issue with auto-scaling is to scale the small running jobs, while long-running jobs are coming under the scheduling problem.

4.1. Web Applications Architectures. Earlier single tier architecture has used the dedicated server for each tier such as a web server (load balancer), application server and database server. The load balancer transfers the load among other instances of single layer architecture. This architecture is not suitable according to the elasticity and scalability features of cloud computing.

Most of the companies are now using multi-tier architecture (e.g., Amazon), where each tier in architecture serves a specific purpose as shown in Figure 4.1. The most important benefit of multi-tier architecture is to manage the scalability and elasticity features. Resource management of multi-tier applications is still a challenge due to higher interdependencies between the different layers [31]. Web tier, application tier, and database tier are deployed on the web server, application server, and database server. The responsibilities of the servers are:

- Web server: It accepts or rejects the incoming client request. Another important work of web server is to serve the static content. It also passes the request to the application server. The response is delivered back to the user.
- Application Server: It receives the request from the web server. The literature survey is biased towards the scaling of VMs for the application server. Business logic is processed by this tier. It requests the database for the required data. Optimization of queries can be done at this level also. After processing

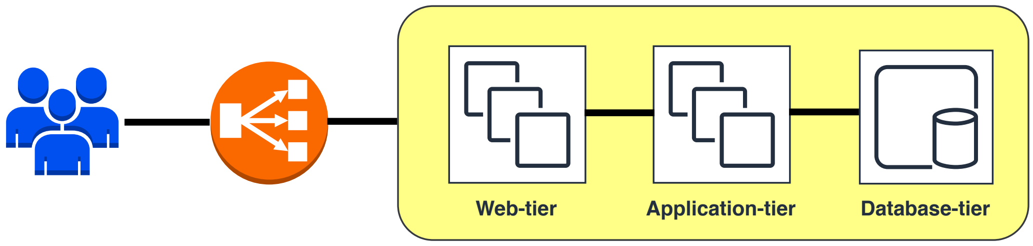


FIG. 4.1. *3-tier Web Architecture [31].*

data again send back to the web servers in the desired format.

- Database Server: Database management system is working at this level. Single tenant and multi-tenant databases compatible with the cloud architecture (e.g. Oracle 12c). Structured and unstructured relational database management system is used to store the data and pass the required data to application servers.

Another type of applications is service-based web applications such as Facebook and e-commerce website of Amazon. Each service represented as the node is connected with another service through directed edges and whole system abstracted as a directed graph. The application further classified in Micro-services and Service-Oriented Architecture (SOA). Our article is mainly focusing on multi-tier web applications.

5. Survey on Auto-scaling Techniques. This literature survey is focusing on auto-scaling techniques specifically for web applications in cloud computing. To the best of our knowledge, there is no survey specifically focuses on web applications auto-scaling for analysis and planning phase. Auto-scaling of cloud computing has a large number of articles published. We put the papers in the associated category by identifying the approach, algorithms used in the research paper for a better understanding. We again revised the models, metrics, monitoring tools, etc., from the articles and create tables according to the classification of techniques. The symbol '-' in tables represent the lack of information in the article.

Author [49] have done a survey of Internet applications deployed on dedicated or shared data centers. It focuses on the web server's admission control issue. This service is related to the SaaS layer of the cloud architecture. The elastic nature of web applications is more relevant with auto-scaling and can be done in two ways: horizontal or vertical. Identification has done on the basis of approach reactive or proactive. The metric used by the technique is identified in the survey. Workload used in the analysis of the approach is also discussed along with the experimental setups. SLA parameter has to check the validity of the model, so SLA parameters are collected from the papers.

Resources are the major factor in auto-scaling. To the best of our knowledge, no existing survey has identified all types of resources used in a cloud environment. Existing surveys considered only on-demand resources in cloud infrastructure. In this paper, the authors have identified other type of resources (e.g. Spot-instances, reserve and on-demand), for application tier elastic layer.

Another important survey done by the authors [88] on resource management in the cloud, but little attention has given to auto-scaling. Author [85] focused on the elastic application including web application. This article classifies the technique and very useful literature survey, but the survey did not focus on the type of resources such as on-demand, reserve and spot instances. Many new emerging fields such as fog computing, edge computing, dew computing takes the support from the cloud computing for elastic services [124].

We have carried out a very diverse survey by including these surveys with specifically to web applications auto-scaling techniques. A survey has done on the basis of auto-scaling technique. Author [85] has used the classification technique and we have added more approaches along with mentioned former groups. This section covers the literature survey on auto-scaling techniques. Each article is reviewed on the basis of auto-scaling taxonomy. The classification of auto-scaling techniques are:

1. Application Profiling (AP)
2. Threshold-based Rules (TR)
3. Fuzzy Rules (FR)

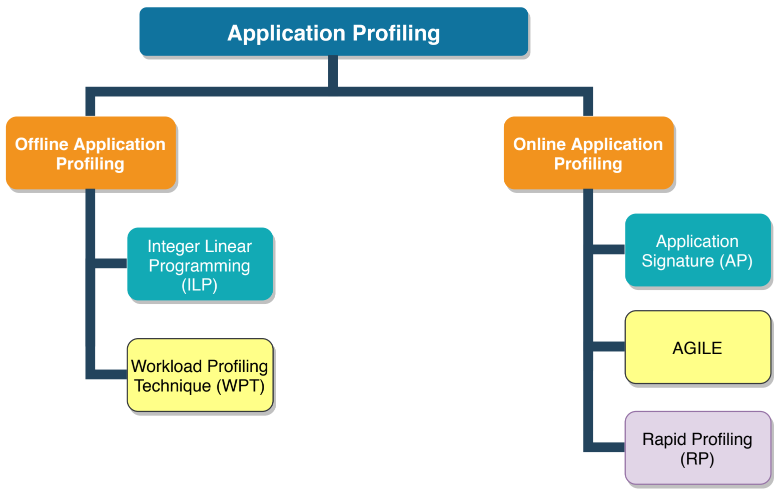

Application Profiling
Offline Application Profiling
Online Application Profiling
Integer Linear Programming (ILP)
Workload Profiling Technique (WPT)
Application Signature (AP)
AGILE
Rapid Profiling (RP)

Fig. 5.1. *Methods of Application Profiling*

4. Control Theory (CTH)
5. Queuing Theory (QTH)
6. Machine Learning (ML)
7. Time Series Analysis (TSA)

5.1. Application Profiling. Application profiling is a process of finding the peak point of resource utilization while running an application workload. The workload used for profiling can be real or synthetic. The test can be both online and offline for application profiling. It is one of the simplest ways to find the desired resources at a different point of time.

Profiling of job through an offline process gives resource requirements at different levels of workload. The produced resources requirements help auto-scaler to monitor the resource provisioning task in a precise manner. Author [119] applied the integer linear programming (ILP). Authors [39, 43, 108] used workload profiling technique. The profiling cons are to reproduce requirements manually after every update in applications.

To overcome the offline profiling issues, online profiling comes in rescue. The fine grain tasks need the VMs immediately to cater to the flash workload. Author [132] devised the technique to profile the application workload with the further classification of application signatures. The classification has been carried out on the basis of a number of machines required at different time stamps of application workload. The trained decision tree update every time when the new application profiled according to its characteristics. Author [98] applied the online technique to estimate the resources of each tier while estimating specific tier other tiers gained ample resources. This process continues for each tier and gets the model for every tier. Author [30] designed a rapid profiling approach for multi-tier web applications. Analysis of resources requirement correlation with each tier of the web application inhomogeneous environment and profile the VM for a specific tier. This approach can estimate the performance of VM without running on each tier. So it helps to put the newly added VM rapidly into the service. Author [128] presented the architecture for the self management of the application in cloud infrastructure. In this article, the author performed the application profiling using micro services. [83] designed the approach using application profiling.

The method to find the critical point of utilization of resources while running of application workload (real or synthetic). It is further classified into two categories (Figure 5.1):

- Off-line Application Profiling: This method of profiling is performed in a detailed manner in resource provisioning of tasks at different stages of workload. The advantage of offline profiling is to perform profiling manually after the applications are updated. The different methods of offline application profiling are:
  - Integer Linear Programming (ILP): It is a mathematical optimization problem. In this partial or

all variables are integer. The constraints and objective function is linear [119].
  – Workload Profiling Technique (WPT): This technique gather the workload information. The workload modeling performed to estimate the performance under the stressed conditions. The workload profiling done with various types of testing techniques such as limit finding, soak testing, etc. [39, 43, 108].
- Online Application Profiling: This method of profiling is dynamic in nature and fulfill the needs of the fine grained tasks that immediately require working virtual machines with different workloads.The different techniques of online application profiling are:
  – Application Signatures (AS): This technique identifies a small set of classes of workload and classifies them according to workloads. The resource allocation of workload is cached at runtime [132, 120].
  – Elastic distributed resource scaling (AGILE): In this technique, wavelets are used for fulfilling the resource demand prediction with much large time for the applications servers to start up before falling short of performance [98].
  – Rapid Profiling (RP): In this technique, the individual performance of the virtual machine instance is profiled after it is obtained from the cloud. This technique helps to judge the best tier for the profiled instance of virtual machine [30].

The cloud is providing resources on-demand. Spot instances can be used with fault tolerance for cloud applications. Maximum techniques are considering the on-demand resources in the auto-scaling decision. Author [108] used the heterogeneous spot instances with on-demand resources which could minimize the cost to a great extent. Spot instances are up to 90% cheaper than on-demand and reserve resources. Spot instances have an out-of-bid issue, but can be useful for flash crowd and event challenges. It will provide the cost benefit to cloud providers.

Table 5.1 shows the taxonomy of reviewed articles in this section.

5.2. Threshold-based Rules. Threshold based techniques are simple to implement and very popular. For deciding the threshold value requires a deep understanding of all the parameters of the corresponding environment and workload. Amazon EC2 [4] and RightScale [113] are using threshold-based technique. The threshold based rule technique is purely related to planning. The number of resources allocated to the application in the form of VM according to a set of rules. There are two types of scaling decision: Scaling up/down vertically increase or decrease the capacity of the working node. Scaling in/out refers to horizontally increasing or decrease the VM capacity. The scaling up/out and scaling down/in is working on the principle in Algorithm 12.

Algorithm 12 The algorithm of threshold rule based auto-scaling

```
1: if $x_1 > tU_1$ and/or $x_2 > tU_2$ and/or ... then          ▷ $tU_i$ is upper threshold
2:     if $Clock$ % dU == 0 then                                ▷ dU is upper duration
3:         $n = n + r$                                          ▷ Scale-out or Scale-up
4:     else if $Clock$ % iU == 0 then                           ▷ iU is upper cool down period
5:         $do - nothing$
6:     end if
7: end if
8: if $x_1 < tL_1$ and/or $x_2 < tL_2$ and/or ... then          ▷ $tL_i$ is upper threshold
9:     if $Clock$ % dL == 0 then                                ▷ dL is lower duration
10:        $n = n - r$                                          ▷ Scale-in or Scale-down
11:    else if $Clock$ % iL == 0  then                          ▷ iL is lower cool down period
12:        $do - nothing$
13:    end if
14: end if
```

There are two parts of Algorithm 12: condition and action. The *Clock* is a system clock, $x1$ and $x2$ are performance metrics (e.g. Number of requests, response time, budget, CPU load, etc.), tU and tL are the upper and lower threshold values of the performance metrics. Threshold conditions checked with upper duration dU

Table 5.1
Taxonomy on application profiling based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[119]	Reactive	Horizontal	AP and ILP	Cost-aware	-	Arrival rate	Cost and response time	Reserved	Custom testbed
[30]	Proactive	Vertical	AP and ANN	Load/ Capacity aware	Custom tool	Response time	CPU and memory usage	On-demand	Custom testbed. SpecWeb , TPC-W, TPC-C applications on Xen
[43]	Reactive and Proactive	Horizontal	AP	Resource-aware	Custom tool	No. of servers and arrival rate	Response time	on-demand	Custom testbed (Intel Xeon 38 servers)
[132]	Proactive	Vertical	AP and CMAC	Workload-aware	Custom tool (Script)	Response time	CPU, I/O, memory, swap	On-demand	Custom testbed. Applications on Xen (SpecWeb , TPC-W, TPC-C)
[98]	Proactive	Horizontal	AP	Cost and resource aware	-	Cost and response time	No. of VMs	On-demand	Custom decision agent (VirtRL) and olio application
[39]	Proactive	Horizontal	AP and TSA	User-behavior	Custom Tool.	Throughput	CPU usage and arrival time	On-demand	Custom testbed. MediaWiki Application
[108]	Reactive	Horizontal	AP	Fault tolerant	Simulated	Availability and response time	CPU load	On-demand	Custom Simulation
[128]	Mixed	Horizontal	AP	Resource aware	Real time	Response time	Request rate and avg. response time	on-demand	Amazon AWS
[82]	Reactive	Horizontal	AP	Data-aware	Simulated	Response latency	Throughput, No. of resources	On-demand	Custom testbed. IBM X3500 M4 machine with Xen
[120]	Reactive	Horizontal	AP	Resource aware	Simulated	Response time and success ratio	CPU usage, response time	On-demand	Custom Simulation
[83]	Proactive	Horizontal	AP	Workload aware	Real testbed	Throughput and response time	Input rate, response time	Reserved and on demand	6 Cassandra 3.0 nodes with Linux Ubuntu 14.04 Server x86 64. Synthetic workload.

and lower duration dL, which considered in the seconds. If certain conditions are met, the action will be taken. The manager should decide resources r acquired or released during the action performed from the total resources n. During horizontal scaling r is the number of VMs and during vertical scaling r is CPU or RAM. The lower and upper cool down period is set as iL and iU, during this time auto-scaler do nothing.

It is easy to deploy a threshold-based technique in a cloud environment. Defining rules are a difficult task, it requires input from the client on different QoS metrics. QoS parameters have performance trade-off. Application manager has to give threshold value of performance metrics parameters. The experiment carried out by the authors [70], they define the performance benefits of the application-oriented metrics than system specific metrics. The threshold needs to be carefully decided otherwise it can cause the oscillation problem [33]. To handle the oscillation issue, certain techniques such as cool down, calm and inertia periods are set, so no scaling decision can take place in these time slots.

Most of the techniques are using one or maximum two performance metrics. Commonly used performance metrics are input request rate, CPU load time and average response time of application. Some of the authors are taking application response time as a performance metric [33, 51] while few focused on the system level metrics such as storage, network, and CPU [52]. Author [87] considered the scaling overhead and SLA violation. Due to the introduction of a new type of resources [108] evaluate the availability metrics along with response time.

The number of threshold values also varies with different techniques. Most of the providers are using two threshold values: Upper and the lower threshold value. Author [52] used four threshold values. In this technique along with upper threshold ($ThrU$), the author defines $ThrbU$, which is marginally lower than $ThrU$ and $ThroL$ is pretty above than the lower threshold ($ThrL$). The significance of setting such a threshold is to determine the trends. Scaling action could be taken as per the trends.

Metrics are fetched from the monitoring tool. Collecting the run-time data from the monitor and taking action as per the rules is a reactive approach.

RightScale [113] voting system is combined with the reactive approach. If most of VMs agree to scale up/out or down/in decisions, then scaling action will be performed. Threshold needs to set by the application manager. Several authors are using the RightScale technique [123, 71, 45, 25]. Author [25] working on the scalability of web applications by defining the set of rules for the active sessions, further extend his work [24] using RightScale. The rules are decided as per the upper and lower threshold value of the sessions. If the sessions are going upper or lower scaling decision will be taken.

RightScale is working on the voting system, but it is highly dependent on the threshold values set by the application-manager and the nature of the application workload. Author [71] compare the RightScale with another technique and concluded the disadvantages of RightScale. Researcher [123] attempted to overcome this issue using the strategy tree. The regression-based model has been used for three types of scaling policies. Strategy tree follows the parent as per the workload trend.

The threshold rule-based technique is popular due to its simplicity and the client can easily understand. Reactive nature of TR technique is a major challenge. Another issue in the TR technique is to set appropriate performance metrics. In a cloud environment, the approximate startup time of the VM is 5 to 10 minutes, so to ready, the machine in a reactive manner put a delay in service, which leads to performance degradation. To resolve this issue to some extent, an author [86] suggested the dynamic threshold values. The initial values of the threshold are static and later dynamic threshold values set as per the SLAs violation.

Multi-cloud scaling is another issue in the auto-scaling. Auto-scaling for three-tier application has been developed [48]. The rule-based technique has been devised to take scaling decision for multiple data center.

Along with the on-demand resources, spot-instances are used for web application provisioning. The spot instances are 90% cheaper than on-demand resources. So effective scaling strategy can provide significant benefit to the clients as well as a service provider to reduce the SLA violation. Author [108] provided a reliable solution to use spot instances for web application provisioning. Spot-instances have a potential for fault tolerance and tackling of web application's flash crowds or flash events workload.

Smart kill [20] is an important idea used to reduce the cost. Most of the service provider charge hourly basis. It can further improve the system performance to avoid VM starting and shutdown time. It can reduce the SLAs violation.

It is easy to implement the rule-based auto-scaling of the particular application. If the workload and nature of application can forecast easily than rule-based technique can be used. Application with an unpredictable pattern should choose other suitable scaling strategies.

Table 5.2 shows the taxonomy of reviewed articles in this section.

5.3. Fuzzy Rules. One of the rule-based technique for an auto-scaling approach is fuzzy rules. In this technique, rules are defined using if-else conditions. The major advantage of fuzzy based auto-scaler over the threshold based rule-based approach is linguistic terms e.g. low, medium, high. It uses the control system as

Table 5.2
Taxonomy on threshold rules based reviewed literature

Ref	**Type**	**Policy**	**Technique**	**Approach**	**Monitor**	**SLA**	**Metric**	**Pricing**	**Experiment**
[71]	Hybrid	Horizontal	RightScale and TSA (LR + AR(1))	User behavior	Simulated	-	CPU load, Request rate	on-demand	Custom simulator with real experiment
[123]	Reactive	Horizontal	RightScale, Strategy tree	Policy Based	Custom tool(4-minutes)	-	CPU idle time, number of sessions	on-demand	Real provider. Amazon EC2 + RightScale (PaaS) + a simple web application
[70]	Reactive	Horizontal	TR	QoS based	Custom tool	CPU load	Average waiting time in queue	on-demand	Public cloud: FutureGrid, Eucalyptus India cluster
[24]	Reactive	Horizontal	RightScale + MA to performance metric	Performance Based	Custom tool	-	Number of active sessions	on-demand	Custom testbed. Xen + custom collaborative web application
[45]	Reactive	Horizontal	RightScale	Qos/ Performance based	Amazon CloudWatch	-	CPU load	on-demand	Real provider. Amazon EC2 + RightScale (PaaS) + a simple web application
[91]	Reactive	Vertical	TR	SLA based	Simulated	-	CPU load, memory, bandwidth, storage	on-demand	Custom simulator, plus Java rule engine Drools
[52]	Reactive	Both	TR	Storage/ Network aware	-	-	CPU load, response time, network link, load, jitter and delay.	on-demand	-
[51]	Reactive	Both	TR	Cost Aware	Custom tool. 1 minutes	Response time	CPU, memory, I/O	on-demand	Custom testbed (called IC Cloud) + TPC
[20]	Proactive	Horizontal	TR + QTH	QoS Aware	Amazon CloudWatch. 1-5 minutes	Response time	Request rate	on-demand	Real provider. Amazon EC2 + Httperf + MediaWiki
[86]	Reactive	Vertical	TR	Cost/SLA Aware	Simulated. 1 minute	Response time	CPU load	on-demand	Custom simulator
[48]	Reactive	Horizontal	TR	Cost/ Performance aware	Custom tool and manual.	Response time, Sessions	CPU load	on-demand	Multi-cloud Amazon EC2 (4 data centers)
[87]	Reactive	Horizontal	TR + Learning Automata	SLA aware	Simulated. 5-minutes	-	Scaling Overhead, SLA violation	on-demand	CloudSim
[79]	Reactive	Horizontal	TR	SLA aware	Custom	Turnaround time	Turnaround time, No. of requests	On-demand	Real. Google compute engine.
[35]	Reactive	Horizontal	TR	Performance aware	Amazon CloudWatch	Response time	CPU usage	On-demand	Real. Amazon EC2

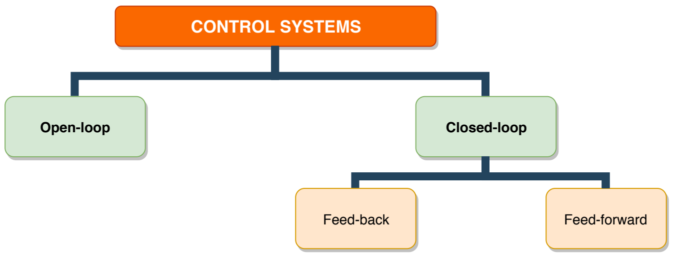


Fig. 5.2. *Categories of control system*

a performance model for estimating the required resources (output variable) for the workload (input variable). Fuzzy sets are formed from the input and output variables, the membership function defines this process between the $(0, 1)$ interval. Fuzzification is the process of transforming input variables into fuzzy sets. The inverse transformation of single numeric values to best inferred fuzzy value is known as defuzzification.

The fuzzy rule-based model used to construct the auto-scaler. This auto-scaler is not able to handle the dynamic workload because most the components of the fuzzy controller are fixed at the design time (e.g. Rule set, membership function). Regular update in the fuzzy controller with on-line monitoring can make the system of adaptive nature, can better handle the dynamic workload [136, 140]. Author [140] proposed a model to apply the fuzzy-controller at the application tier, and forecast the resources required input workload. Author [136] use the same method for the database tier. Predict the required resources r_{t+1} for the time step $t+1$, considering no flash workload during that time step. Author [47] developed a dynamic approach for horizontal scale-out for the dynamic workload. Author proposed a fitness function: $\mathcal{F}_k = \sum_i^N a_i \frac{m_{i,k}}{\mathcal{R}_i}$. Here, metric observation represented by $m_{i,k}$ and $\mathcal{R}_i$ at k sample time $a_i \epsilon [0, 1]$. The model works well under the failure of VMs, and robust for different workloads.

Author [13] designed a fuzzy controller for scaling of scientific applications. The vertical scaling policy developed based on threads in the application and load on the VM. In future, proactive policy can be combined with this technique to reduce the waiting time in VM setup.

Many research articles extend the work with the neural fuzzy controller. Four-layer neural network uses to represent the fuzzy model [74]. The first layer belongs to the input node. The second layer represents each input variable membership to the fuzzy set. Layer three determines the precondition part of fuzzy rules. Final layer act as a defuzzifier, which used to convert the layer 3 in numeric output. The rules and membership of nodes are formed on-line with the help of parameters and structure learning.

Table 5.3 shows the taxonomy of reviewed articles in this section.

5.4. Control Theory. This method is used in two phases analysis and planning of the MAPE loop. It automates the processing systems such as data centers, storage systems, and web server systems. The main goal is to automate the scaling process. The controller maintains the controlled variable y and manipulated variable y_{ref} matches to the desired level. Manipulated variable act as an input to the target system, the output is measured by the sensor.

The different categories of control systems are (Figure 5.2):

- Open-loop: It is also called the non-feedback method. In this method, feedback is not considered for validating the desired goal. The output is calculated using the present state of the system.
- Closed-loop: The closed loop systems are further categorized into two categories:
  - Feed-back: The monitoring of the output and also a deviation from the goal is monitored by the feedback controller.
  - feed-forward: The anticipation of any kind of error in output is performed by the feed forward method. The action is performed in it after considering the type of error in the system.

Feedback controller is mostly used in the reviewed articles (Figure 5.3). It further divided into several categories [104] (Figure 5.4):

- **Fixed gain controllers**: The simplest of all the available controllers. The different types of fixed gain

Table 5.3
Taxonomy on fuzzy rules based reviewed literature

Ref	**Type**	**Policy**	**Technique**	**Approach**	**Monitor**	**SLA**	**Metric**	**Pricing**	**Experiment**
[140]	Reactive	Horizontal	FR	Workload aware/ SLA aware	Custom tool	Throughput	CPU allocation, Arrival rate	Reserved containers	Custom testbed + applications (Java Pet Store)
[73]	Proactive	Horizontal	FR	QoS based	-	End-to-end delay	Number of servers per tier	on-demand	Simulation
[136]	Proactive	Vertical	FR	Qos Based	Xentop. 10 seconds	Response time, through-put	CPU load, No. of queries, disk I/O bandwidth	on-demand	Custom setup. Xen + (RUBiS and TPC-H)
[74]	Proactive	Vertical	FR + ANN	Workload aware	Custom tool. 3 minutes	End-to-end delay	Number of requests, resource usage	on-demand	Simulation
[40]	Reactive	Horizontal	FR	SLA based	Custom tool	Response time	Arrival rate	on-demand	Custom simulator
[59]	Proactive	Horizontal	FR	Load aware	Custom tool	Number of hits	RMSE	On-demand	Microsoft Azure Cloud
[47]	Proactive	Horizontal	FR + PID controller	QoS based	Amazon cloudwatch	CPU usage, NETIN, NETOUT	CPU load and network usage	on-demand	Real setup: Amazon EC2
[80]	Reactive	Horizontal	FR	Resource aware	Amazon Cloudwatch	Response time. 200 ms	Arrival rate, VM allocation	On-demand	Amazon EC2
[13]	Proactive	Vertical	Fuzzy rules + CPU controller	Parallel applications aware	Custom tool	Response time	Thread in application and load on VM	on demand	Simulation with CloudSim.

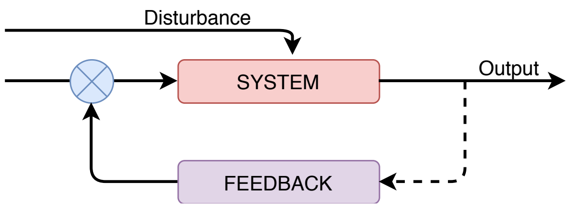


Fig. 5.3. *Working of Feedback control system*

controllers are PID(Proportional-Integral-Derivative), PI(Proportional-Integral) and I(Integral). In it, Proportional Integral Derivate (PID) is most common controller, with following control algorithm:

$$u_k = K_p e_k + K_i \sum_{j=1}^{k} e_j + K_d \left(e_k - e_{k-1}\right) \tag{5.1}$$

Manipulated variable is represented by u_k (e.g. New number of VM); e_k difference between set point y_{ref} and y_k output; k_i, k_p and k_d are the integral, proportional and derivative gain parameters, which further adjust as per the target system.
The authors of [77, 76] use the I controller to manage the required VMs on average CPU utilization. Park and the author [103] deploy PI controller to adjust the resources for batch jobs and used their execution process. k_p and k_i (gain factors) adjust manually according to trail-and-error [77] basis or target application model. Authors [142] adjusted a PID controller derivative term using the RIL agent. The agent learns to reduce the sum of squared error of control variables without affecting budget and

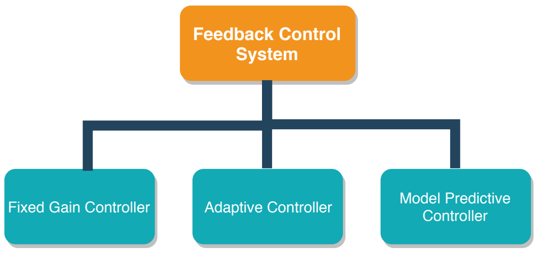


FIG. 5.4. *Categories of Feedback control system*

time constraints over time.

- **Adaptive controllers**: These controllers are the adaptive controllers that work as per the online data made available by target systems. It is incapable of handling the flash load. It is further classified into three categories
  - Self-Tuning PID controller(SPID)
  - Self-tuning regulator(STR)
  - Gain scheduling(GS)

  The adaptive controller is also used in the literature. For example, the author [2] used the two models, adaptive and proactive controllers for scaling-down, based on the input workload and dynamic gain parameters. Scaling-up is done using a reactive approach. Author also devised proportional controller using proactive technique [1]. Author [102] introduced an adaptive PID controller for MIMO and used second-order ARMA for non-linear relationships between performance and resource allocation. The controller can fit the disk I/O usage and CPU. Author [14] used smoothing splines with gain. A gain-scheduling adaptive controller applied to estimate the number of servers for input workload. Author [62] combine the Kalman filter with controllers (SISO and MIMO) for CPU allocation to VMs. Author [41] predicted job completion using kriging model. The master node enqueued all the incoming request.
- **Model predictive controllers (MPC)**: It is proactive in nature that considers the present output and also this model forecasts the future behavior of the system. One of its categories is Look-ahead controller [114]. An optimization problem is solved by considering the cost function. These types of controllers come under the proactive approach. Author [114] devised workload forecasting model using the ARMA model and look-ahead controller. Fuzzy Model Predictive Controller developed using the fuzzy model [136].

  Author [38] devised the auto-scaling technique for vertical memory. The application performance and resource utilization considered in developing the hybrid controller. The objective is to consume less memory for the task, and achieve the highest memory utilization. The author achieved 83% memory utilization. This work can further extend to increase memory utilization and considered the cache memory in vertical scaling.

  The auto-scaling task is highly dependent on the design of the controller and the target application. The controller main objective is to add and remove the resources in the auto-scaling process is very effective. The problem still persists, when the workload is dynamic and non-linear. General auto-scaling model is required, that can be an adaptive controller with MPCs.

As discussed earlier, the controller has to tune the input variable (number of VMs) to calculate the output variable (CPU load). In order to achieve this goal, a model has been devised to represent the formal relationship, which determines how input parameters affect the output variables. This relationship is known as a transfer function in control theory. PID controller represents with a linear equation, there is also the possibility to define with a non-linear equation. Simple PID controller Single-Input-Single-Output(SISO) is used, but there is also a provision to use Multiple-Input-Multiple-Output(MIMO). Following performance models have been used in literature:

- ARMA(X) [102]: ARMA (Auto-regressive Moving Average) model is used to define the characteristics of time series and draw future predictions. ARMAX (ARMA with eXogenous input) devised the

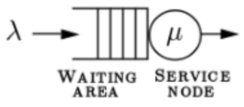


FIG. 5.5. *A simple queuing model with one server [26].*

relationship between two-time series.

- Kalman filter [62]: It is used to make a prediction of time series. It is a recursive algorithm.
- Smoothing splines [14]: It is a polynomial function used to smooth the curves to avoid the noisy observations.
- Kriging model or Gaussian Process Regression [41]: Statistical framework combined with linear regression to predict future values. Unsampled data have been used to perform the forecasting with a confidence measure.
- Fuzzy model [74, 136, 140]: Fuzzy models are used with fuzzy rules. Set of membership elements assigned a degree of value between 0 and 1 (Boolean logic).

Author [101] devised a RIL based controller. This system does not assure the performance in all type of conditions. The work carried specifically for business logic tier. The proposed solution able to handle the controller failures. Author [129] designed a hybrid technique for auto-scaling of cloud applications. SDN controller applied to control the activities of the environment. The proposed model able to reduce the SLA violation upto 0.03% only.

Tables 5.4 and 5.5 show the taxonomy of reviewed articles in this section.

5.5. Queuing Theory. It is one of the widely used for modeling Internet applications. It is used in the analysis phase of the auto-scaling process. It estimates the performance metrics and waiting time for the requests. Queuing theory is a field of applied probability to solve the queuing problem. Queuing issue is quite common in many fields such as telephone exchange, petrol station, supermarket, etc. The structure of the model is shown in Figure 5.5. The requests arrive at the server at mean arrival rate λ and remain in queue till the processing. One or more servers are available to process the requests at the mean process rate μ.

Author Kendall [65] represents the standard notation for queuing model known as kendall's notation. It describe the queue as $\mathcal{A}/\mathcal{B}/\mathcal{C}/\mathcal{K}/\mathcal{N}/\mathcal{D}$.

- $\mathcal{A}$: Arrival process.
- $\mathcal{B}$: Service time distribution.
- $\mathcal{C}$: Number of servers.
- $\mathcal{K}$: Capacity of system. The number of places in the system.
- $\mathcal{N}$: Calling population. Size of users from where the request comes. Open population has an infinite number of customers and closed model is a finite number of customers.
- $\mathcal{D}$: Queue's discipline. It represents the priority of jobs.

$\mathcal{K}, \mathcal{N}, \mathcal{D}$ elements are optional, by default variables are considered as $\mathcal{K} = \infty$, $\mathcal{N} = \infty$ and $\mathcal{D} = FIFO$. FIFO (First In First Out) is mostly used, which served the request as they come. Another important one is PS (Process sharing). M (Markovian) refers to a Poisson process characterized by λ, which indicates the number of arrival request per time unit. D stands for deterministic also refers constantly. G is also used commonly known as general distribution.

Multi-tier applications are complex in nature and the queuing network can be useful. The load balancer is represented by a single queue and distribute the coming requests to the VMs.

Queuing theory is useful for stationary nature systems. It can work with both proactive and reactive kind of environment. The main objective of the cloud-based system is to develop a model on the basis of some known parameters (e.g. Arrival rate λ). Performance metrics measured as the mean response time, and the average waiting time in the queue. The web application workload is dynamic, it requires the timely recalculation of queuing model and metrics.

The queuing model can be used in two ways: simulation and analytical method. The analytical approach can be used with simple models. $M/M/1$ and $G/G/1$ are the well-known methods used to define the arrival

Table 5.4
Taxonomy on control theory based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[103]	Reactive	Vertical	PI controller	Cost-aware	Sensor library	Job's deadline	Job progress	On-demand	Custom setup. BLAST , OpenLB, ADCIRC, WRF, and Montage applications on HyperV
[77]	Reactive	Horizontal	PI controller (Proportional thresholding) and ES	-	(Xen) Hyperic HQ	-	CPU load, request rate	On-demand	Custom setup. Simple web service + Xen + ORCA
[102]	Proactive	Vertical	MIMO adaptive controller	SLO based	Xen + custom tool. 20 seconds	Response time	CPU usage, disk I/O, response time	On- demand	Custom testbed. Xen + 3 applications (RUBiS, TPC-W, media server) + ARMA (performance model)
[14]	Proactive	Horizontal	CTH: Linear Regression + Gain-scheduler (adaptive) + Smoothing splines + (performance model)	Resource-aware	20 seconds	Response time	No. of servers and requests, response time	on-demand	Real provider. Amazon EC2 + CloudStone benchmark
[62]	Proactive	Vertical	CTH: Adaptive SISO and MIMO controllers + Kalman filter	Resource-aware	Custom tool. 5-10 seconds	Response time	CPU load	on-demand	Custom testbed. Xen + RUBiS application
[76]	Reactive	Horizontal	CTH: PI controller (Proportional thresholding)	SLO based/ Workload based	Hyperic SIGAR. 10 second	-	CPU load, request rate	on-demand	Custom Setup. Xen + Modified cloudStone (Hadoop)
[2]	Hybrid	Horizontal	CTH: QTH + Adaptive controllers	SLA Based	Simulated	No. of request not handled	Service rate, No. of requests	on-demand	Custom simulator in Python
[1]	Hybrid	Horizontal	CTH: Adaptive, Proportional controller + QTH	Load Aware	1 minute simulated.	-	Service rate, Total no. of requests and pending in buffer	on-demand	Custom simulator (using Python)
[41]	Proactive	Horizontal	CTH: Self-adaptive controller + Kriging model (performance model)	QoS aware	-	Execution time	Number of incoming and enqueued requests, number of VMs	on-demand	Custom setup. Private cloud + Sun Grid Engine (SGE)
[53]	Proactive	Horizontal	CTH: fuzzy controller + TS	Workload Aware	Custom tool (Fuzzy controller)	Response time	Number of hits, RMSE	on-demand	Custom testbed. Azure Small VM
[36]	Proactive	Horizontal	CTH: Learning automata	SLA Aware	Custom tool. 5 minutes	Response time	Scaling overhead	on-demand	Cloudsim

Table 5.5
Taxonomy on control theory based reviewed literature (continuation)

Ref	**Type**	**Policy**	**Technique**	**Approach**	**Monitor**	**SLA**	**Metric**	**Pricing**	**Experiment**
[38]	Proactive	Vertical	CTH: Hybrid Controller + TS	Application and Resource aware	Control interval using /proc/ meminfo	Response time	CPU and memory utilization	on-demand	Custom testbed. Xen hypervisor (RUBBoS application)
[105]	Proactive	Horizontal	CTH: PID controller + Fuzzy logic	Resource aware	Amazon Cloudwatch	CPU utilization, sensitivity analysis	Scheduling gain, VM failure	on-demand	Custom tested and Amazon EC2. Wikipedia, FIFA world cup workload
[138]	Hybrid	-	CTH: Software defined network (SDN) controller	Security aware	-	DDoS attack prevention	Arrival rate	Containers	Simulated
[101]	Proactive	Horizontal	CTH: Reinforcement learning	Application aware	6 minutes	Response time	Cost and SLA violation	on-demand	RUBiS applicatoin. Custom testbed.
[129]	Hybrid	Both	Software defined network (SDN) controller	Resource aware	Simulated. 1 minute	Response time	CPU capacity, bandwidth	On-demand VMs or Containers	Simulate with CloudSimSDN

and service process. Simulation can be used for the complex system to obtain the desired metrics.

$M/M/1$ (Poisson-based) is a basic queuing model. Exponential distribution is followed by both the arrival times and the service times. The mean response time R of $M/M/1$ model can be calculated as $R = \dfrac{1}{\mu - \lambda}$. Arrival rate is represented by λ and μ shows the service time respectively. $G/G/1$ is another simple method. Inter arrival time and service time are controlled by general distributions with prior information of mean and variance. $G/G/1$ system is represented by the following equation:

$$\lambda \geq \left[s + \frac{\sigma_a^2 + \sigma_b^2}{2(R - s)} \right]^{-1} \tag{5.2}$$

R is the mean response time and s is average service time. The variance of inter-arrival time is σ_a^2 , σ_b^2.

Little's Law [64] is also used in many queuing scenarios. It states that the average number of requests $E[C]$ in the system is same as the average customer arrival rate λ multiplied by the average time of each customer in the system $E[T] : E[C] = \lambda \times E[T]$.

Simple and complex queuing used in the research articles for analysis of the performance of applications and system.

Author [1, 2] used a $G/G/n$ queue to model a cloud application, and n represents the number of servers. The model used to calculate the required resources to process the hosted application workload λ, the response time according to the configuration of servers.

An elastic cloud application can be represented using the queuing network, considered each VM as a separate queue. Author [130] devised a technique using $G/G/1$ queues. Histogram used to predict the peak workload. The number of servers calculated as per the queuing model and peak workload in specific time step. The reactive approach further used to correct the value. The deficiency of the technique is the under-utilization of resources.

Multi-tier applications can be deployed in a cloud environment and assigned one or more queues for a specific tier. Author [141] proposed a closed system with a network of queues, which handles a limited number of users. Author [50] deploy the multi-tier application as an open system. $G/G/n$ queues have been used and considered one queue per each tier. Author [8] used the queuing model for a three-tier web application. It computes the response time per each tier. Author [10] proposed a hybrid auto-scaling technique. The queuing theory clubbed with the time series analysis technique. The prediction error and prediction interval can be reduce with appropriate selection of the model using classification technique.

The models discussed are considered the analysis part of the MAPE loop. Some techniques are used to implement the planning phase using optimization algorithm and predictive controller [2] in order to maximize the revenue for the datacenters [134]. On-line monitoring captures the number of requests, which further input to the queuing model to estimate the number of VMs required for the processing of application workload [50, 130]. Author [141] proposed a model using regression technique to approximate estimate the number of CPU by calculating the quantity of a client requests at each tier (e.g. Browsing, ordering or shopping). Author [44] proved the importance of parallelization of workload in multi-core system. The model can be improved with the generic design which can implemented in Hadoop, edge and other environments for vertical scaling.

Application of the queuing model is for cloud application and system modeling. It defines the static architecture and required an update in parameters or structure of the model. Simulation and the analytical tool can be used to solve the model. Any change in the number of input request or a number of resources needs to update the model, which is not cost-efficient. Most of the articles used CloudSim simulator [18], and some authors [135] developed a new simulation tool specifically for the auto-scaling. It includes the trace file of widely used workloads.

Queuing model is a component in auto-scaler. The model works quite efficiently with linear parametrized applications, and not fit for the applications with a non-linear workload. The queuing model requires more efforts to work with multi-tier application because of complex nature and non-linear relationships.

Table 5.6 shows the taxonomy of reviewed articles in this section.

5.6. Machine Learning. Machine learning is a technique that is used on online learning for the constructing dynamic approach for the estimation of resources. It is a self-adaptive technique as per the workload pattern available [3]. The machine learning algorithms reclassified into various categories. The review is made for the different categories used in this particular literature.

- **Reinforcement Learning (RIL)**: It is one of the widely used machine learning approaches. The agent performs an action as per the received environmental state. As in a cloud environment, auto-scaler acts as an agent. An agent works on the principle of trial and error method. It gets a response or reward for each action performed [126]. The optimal scaling decision is taken by sensing the current state, performance metric, and type of application. The auto-scalar or agent works to yield more rewards. The main objective of machine learning is to control the data. The purpose of the agent is to fetch appropriate action of each states.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{0}^{\infty} \gamma^k r_{t+k+1} \tag{5.3}$$

At time $t+1$ the rewards gained are R_{t+1}, the γ factor is the discount factor. The value function $Q(s,a)$ known as Q-value function defines the policy. The $Q(s,a)$ values evaluates the cumulative rewards for each state s by execution action a.

$$\mathcal{Q}(s,a) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1|S_t} = s, a_t = a \right\} \tag{5.4}$$

This proactive learning method makes the decision with the state of application about the future reward(e.g. response time). The result is generated after the complete execution of the application on the cloud. The phases analyze and plan of MAPE process are covered by RIL techniques.Firstly, data about the application and rewards are taken from the lookup table (or any other structure) for later use (analyze phase).In the planning phase, the data is used to take the scaling action.

To apply RIL to auto-scaling, some basic elements need to define: first, the action set A, the state space S, and the reward function R. The first two depend upon the type of scaling: horizontal and vertical. Reward function depends upon the cost to acquire the resources (VMs, network bandwidth, etc.), and SLA violation penalty cost. While considering the horizontal scaling, the state defines as input workload and the number of VMs.

Table 5.6
Taxonomy on queuing theory based reviewed literature

Ref	**Type**	**Policy**	**Technique**	**Approach**	**Monitor**	**SLA**	**Metric**	**Pricing**	**Experiment**
[134]	Reactive	Horizontal	QTH	Budget Aware	Simulated	Response time	Arrival rate, service time	on-demand	Custom simulator (Monte-Carlo)
[141]	Proactive	-	QTH + Regression (Predict CPU load)	QoS Based	Custom tool. 1 minute	-	Number and type of transactions (requests), CPU load	on-demand	Custom simulator, based on C++Sim. + Data collected from TPC-W
[130]	Hybrid	Horizontal	QTH + Histogram + Thresholds	Resource Aware	Custom tool. 15 minutes	Response time	Peak workload	on-demand	Custom testbed. Xen + 2 applications (RUBiS and RUBBOS)
[8]	Proactive	Horizontal	QTH + Historical performance model	Prediction /Resource Aware	-	Response time	Arrival rate	on-demand	Custom testbed. Eucalyptus + IBM Performance Benchmark Sample Trade
[50]	Reactive	Horizontal	QTH (model) + Reactive scaling	Cost-Aware	Custom tool. 1 minute	Response time and cost	Arrival rate, service time	on-demand	Custom testbed (called IC Cloud) + TPC-W bench mark
[42]	Proactive	Horizontal	QTH + kalman Fil ter	User Aware	Custom tool. 10 seconds	Response time	Arrival rate	on-demand	Custom TestBed
[133]	Proactive	Horizontal	QTH	QoS based	-	Response time, Bottlenecks	Arrival rate	on-demand	Custom testbed. Open stack
[11]	Proactive	Both	QTH	Resource aware	Custom	Response time	Request arrival rate, No. of VMs per tier	on-demand	Custom testbed. RUBiS benchmarking application
[56]	Reactive	Horizontal	QTH	Resource aware	Custom	Response time	Arrival rate, No. of resource allocation	on-demand	Custom testbed. OpenStack.
[116]	Proactive	Horizontal	QTH + Markov chain	SLO aware	-	Response time	CPU utilization, throughput, Bandwidth, Session loss	on-demand	Simulation and real testbed on Amazon EC2. Synthetic workload using Apache JMeter
[135]	Proactive	Horizontal	QTH (Simulator for autoscaling)	User behavior and resources	Custom tool. 15 minutes	VM utilization, Arrival rate	Arrival rate	on-demand	Custom Simulator for general purpose
[44]	Proactive	Both	QTH + Amdahl's Law+ Kalman Filters	Cost-aware	Custom testbed. 10 seconds	Cost and Response time	Resource allocation, service rate	on-demand	Real testbed using OpenStack. RUBiS benchmarking applications, WITS traces and synthetic workload.
[10]	Hybrid	Horizontal	QTH + threshold rules	SLO aware	Custom agent	Resource Oscillation	No. of VMs	-	BUNGEE experiment controller. FIFA World Cup, BibSonomy, IBM CICS transactions, German-Wikipedia, Retailrocket traces.

Researcher [127] devised using (w, u_{t-1}, u_t), where w is the the total number of user requests observed per time period; u_t and u_{t-1} are the number of VMs allocated to the application in the current time step, and the previous time step, respectively;
Authors [32] followed the definition of the state as (w, u, p), where w is the total number of requests and u is considered as the number of VMs allocated and p as performance in the contract of average response time. Horizontal scaling has been done on the basis of three decisions: append the new VM, deallocate the VM and do nothing.
On the other hand, the resources assigned to each VM (CPU, RAM) for vertical scaling are considered in state definition. Authors Authors [112, 139] devised the state definition as (mem_1 , $time_1$, $vcpu_1$,...., mem_u, $time_u$, $vcpu_u$), where mem_i, $time_i$ and $vcpu_i$ are the i^{th} VM's memory size, scheduler credit and number of virtual CPUs. The possible operations for three variables can be increased, decrease or no-operation. The RIL defines the task as a combination of each variable operation. Rao et al. again proposed a technique, where RIL agent learned per VM [111]. State definition is configured as CPU, bandwidth, memory, and swap.
RIL learning is also very useful for tasks that are very closely related to auto-scaling problems, for example, the configuration of application parameters [16, 139]. Author [139] include the RIL agent with ANN to configure the parameters as per the application and VM performance, such as Session timeout, MinSpareServers, Keepalive timeout, Maxclient.
RIL can also be combined with control theory. Authors [142] combine the RIL agent with a PID controller. Application parameters are guided by the controller to meet the Service Level Objectives (SLO). The resources are dynamically provisioned according to the parameters. The author [95] estimated the future prediction problems with anomaly detection technique. The isolation unsupervised tree is applied for the upgrading the model. The local and global level auto-scaler design is proposed. The author consider the containerized resources in cloud infrastructure. In real environment, there is more complex anomalies are presented. So, more number of anomalies need to detect to design the robust solution.
RIL algorithms are important to gain the best management policy for cloud applications without any initial knowledge. It is an online approach to learn and adapt as per the workload, application or system change. RIL technique could be a better approach to handle the auto-scaling for general applications, but the RIL approach is not mature enough to deal with the real cloud environment. This is an open research field and efforts are required to handle the flash workload and rapidly change of state and actions.

- **Hidden Markov Model (HMM)** : This is modeled with a hidden Markov chain with the hidden states. In the hidden Markov chain technique, the state is known to the observers whereas in HMM it is invisible to observers. For example, two friends Sam and Rick are communicating on the phone. Sam describes his activities (eating, drinking) and Rick estimated the weather condition from his activities. The estimation is performed on the basis of the maximum likelihood estimation technique. The parameters of HMM are x defines the states in the model, y is the total observations under consideration and state transition probability defined with a and b the output probability shown in Figure 5.6.
  An HMM model usually require fewer observations as compare to basic Markov chain models [99, 100]. The system condition needs to observed minutely and input to the model. The author used the Weka data mining tool for implementing the HMM model, 2 states used with 0.01 iteration cutoff and spherical covariance. As per an author, model given 97% accuracy in the auto-scaling decision. Database tier is not considered in this work.
- **Support Vector Machine (SVM)**: It is also used in some articles. It is one of the supervised learning technique that is a combination of two techniques: classification and regression. It can be applied to non-linear workload [97]. The classification is done on the basis of clusters formed and generates the classes for data under specific region. Linear SVM can be used for classifying the web workload.
  Researcher [78] proposed an adaptive technique with SVM and Linear Regression (LR). Characterization has done on the basis of priority of the job and workload pattern. The generic model has been developed

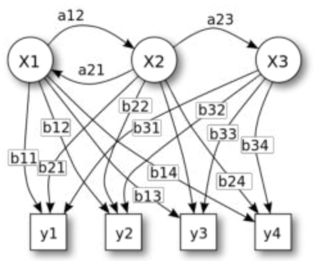


FIG. 5.6. *Hidden Markov Model (HMM) [110].*

for cloud applications. The model can be extended further considering the different QoS parameters according to the SLA and application.

- **Q-learning:** It is the most used algorithm in auto-scaling by extending it with various techniques. Author [127] used the SARSA [126] approach. Some articles [112, 139, 111, 16] did not specify which machine learning approach used in their research, but problem definition and update function resemble the SARSA. There are many problems in Q-learning and SARSA addressed various ways as following [9, 33, 32, 139, 127]:
  - Initial performance is bad and required a large training period. On-line performance is poor before the best solution is found. It requires the exploration of different states and actions.
  - The curse of dimensionality problem in Q-learning. The number of state variables grows the state exponentially. States are stored in lookup tables. As the table grows, it takes time to search for the possible state from the lookup table.
  - The environmental condition has a great impact on the performance of the RIL algorithms. As the conditions change the best optimal solution degrades the performance. RIL need to design to work with application behavior or environmental conditions.

Tables 5.7 and 5.8 show the taxonomy of reviewed articles in this section.

5.7. Time Series Analysis. Time series analysis is a very popular model and has applications in many areas including economics, bioinformatics, finance, engineering, etc., to predict the measurement for future time steps. An example, the number of requests that reach the server for an application at one-minute intervals. The time series is used in the analysis phase to forecast future workload.

Performance metrics (input workload, CPU load) sampled at fixed time intervals (e.g. 5 minutes). The predicted series X gained from the sequence of observation w.

$$X = x_t, x_{t-1}, x_{t-2}, ..., x_{t-w+1} \tag{5.5}$$

Time series applied in auto-scaling to forecast the future workload or resources required. Predefined rules are designed in planning phase [66], and optimize the resource allocation [114].

As discussed earlier, the main objective of time series analysis is to predict the future workload, on q observation from historical workload known as a history window or input workload (where $q \leq w$). Time series analysis classified in two categories: direct prediction and identification of a pattern in time series. The first category, direct prediction contains auto-regression, moving average, ARMA, ARIMA, exponential smoothing, and machine learning approaches:

- Moving average (MA): A widely used to smooth a time series to filter noise from random fluctuation and to make predictions. General formula of MA is as follows:

$$\hat{x}_{t+r} = a_1 x_t + a_2 x_{t-1} + ... + a_q x_{t-q+1} \tag{5.6}$$

$\hat{x}_{t+r}$ is a forecast value from last q observations with weighted average. Prediction interval denotes by r, and typically sets to 1. Equal or different weight factors are assigned to the values denoted

Table 5.7
Taxonomy on machine learning based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[127]	Proactive	Horizontal	RIL+ ANN-SARSA + Queuing model	Resource aware	-	Response time	Arrival rate, previous number of VMs	on-demand	Custom testbed (shared data center). Trade3 application (a realistic simulation of an electronic trading platform)
[112]	Proactive	Vertical	RIL+ANN	VM performance	Custom tool	Throughput, response time	CPU and memory usage	on-demand	Custom testbed. Xen + 3 applications (TPC-C, TPC-W, SpecWeb)
[33]	Proactive	Horizontal	TR + RIL	Resource aware	Custom tool. 20 seconds)	Response time	Request rate, response time, number of VMs	on-demand	-
[142]	Proactive	Vertical	CT- PID controller + RIL + ARMAX model + SVM regression	QoS based	-	Application-related benefit function	Application adaptive parameters CPU and memory	on-demand	-
[111]	Proactive	Vertical	RIL + CMAC	SLA based/ Capacity based	Custom tool (Script)	Response time	CPU, I/O, memory, swap	on-demand	Custom testbed. Xen + 3 applications (TPC-C, TPC-W, SpecWeb)
[32]	Proactive	Horizontal	RIL	Resource-aware	-	Response time, cost	Number of user requests, number of VMs, response time	on-demand	Custom testbed. Olio application + Custom decision agent (VirtRL)
[139]	Proactive	Vertical	RIL + ANN	Budget aware/QoS based	Custom tool	Response time	CPU and memory usage	on-demand	Custom testbed. Xen + 3 applications (TPC-C, TPC-W, SpecWeb)
[9]	Proactive	Horizontal	RIL	Workload aware	Simulated	Response time, cost	Number of user requests, number of VMs, response time	on-demand	Custom simulator (Matlab)
[16]	Proactive	Vertical	RIL + Simplex	Resource aware	Custom tool	Response time, throughput	CPU, memory, application parameters	on-demand	-
[99]	Reactive	Horizontal	HMM	SLA aware	-	Cost-Performance	Number of request, MAPE, RMSE	on-demand	-
[78]	Reactive	Horizontal	LR(Linear regression) + SVM	Workload based	Custom tool. (1, 10 and 60 minutes)	Cost-Performance	Prediction time (ms)	on-demand	Simulation
[60]	Reactive	Horizontal	RIL: Fuzzy Q-learning	Workload aware	Custom.	Response time	No. of VMs	On-demand	Simulated and Custom testbed.

Table 5.8
Taxonomy on machine learning based reviewed literature (continuation)

Ref	**Type**	**Policy**	**Technique**	**Approach**	**Monitor**	**SLA**	**Metric**	**Pricing**	**Experiment**
[12]	Proactive	Horizontal	RIL	Resource-aware	Custom	Response time	CPU usage, throughput, Mean Q-value	On-demand	Custom testbed and CloudRL matlab simulation. Benchmarking applications (RUBiS, RUBBoS and Olio)
[137]	Reactive	Horizontal	RIL: Q-learning	Workload aware	Custom. 1 hour.	-	Arrival rate and rewards	reserved, on-demand and spot instances	Custom testbed. Matlab.
[95]	Proactive	Vertical	Unsupervised learning: Isolation based tree	Resource aware	Custom	Response time	CPU utilization, response time, no. of failed sessions	Reserved and on-demand	CloudSim. Custom testbed. RUBiS benchmarking application.
[63]	Proactive	Horizontal	HMM: Multi layered	Data-streaming applications	Custom.	-	CPU usage, Prediction time, RMSE, MAPE	on-demand	Custom testbed. Hadoop for data streaming.

by $a_1, a_2, a_3, ..., a_q$. There are types of MA's are Simple Moving Average (SMA) and the Weighted Moving Average (WMA). Moving average performs in various forms, but the objective of MA is the same. In simple moving average's general formula is considering the arithmetic mean of previous q values. It considers the same weight of all the observations. WMA considers the different weight for each observation. The highest weight assigned to new observation, while less weight is given to previous observations.

- Exponential Smoothing (ES): In contrast to the moving average, the weighted average of previous observation is also calculated, but the ES considers all the observation of time series (w values) from past history. The new parameter α, the smoothing factor has very less influence on the predicted value, because exponentially decreasing weight assigned to the new observations. There are many versions of ES such as simple ES and Brown's double ES [15]. The smoothed value s_t is calculated from the present observation and past smoothed value based on recursive formula: $s_t = \alpha x_t + (1-\alpha)s_{t-1}$. Time series with fewer trends can be forecast using simple ES. Brown's double ES is suitable for linear trend time series. Two smoothing series are required to calculate as:

$$\begin{aligned} s_t^1 &= \alpha x_t + (1-\alpha)s_{t-1}^1 \\ s_t^2 &= \alpha s_t^1 + (1-\alpha)s_{t-1}^2 \end{aligned} \tag{5.7}$$

s_t^2 is calculated from simple ES to s_t^1. The trend d_t and level c_t is obtained from s_t^1 and s_t^2 smoothed values. The forecast value $\hat{x}_{t+r}$ is obtained from following formula:

$$\begin{aligned} \hat{x}_{t+r} &= c_t + r_{dt} \\ c_t &= 2s_t^1 - s_t^1 \\ d_t &= \frac{\alpha}{1-\alpha}s_t^1 - s_t^2 \end{aligned} \tag{5.8}$$

- Auto-regression AR(p): The future value is forecast using the linear weighted sum of previous observation p in the time series:

$$x_{t+1} = b_1 x_t + b_2 x_{t-1} + ... + b_p x_{t-p+1} + \epsilon_t \tag{5.9}$$

p represent the number of observations in the AR equation, which could be different from history window w length. White noise ϵ_t is added in the formula. AR coefficients are calculated using different methods such as maximum likelihood or least squares. The widely used technique in the literature is Yule-Walker equations and auto-correlation coefficient. The formula of auto-correlations as follows for x_t to x_{t-k}, where $k = 1, 2, 3...$:

$$r_k = \frac{covariance(x_t, x_{t-k})}{var(x_t)} = \frac{E[(x_t - \mu)(x_{t-k} - \mu)]}{var(x_t)} \tag{5.10}$$

- Auto-Regressive Moving Average, ARMA(p,q): This model is the hybridization of both AR of order p and MA of order q. ARMA model is described as:

  $$x_t = b_1 x_{t-1} + ... + b_p x_{t-p} + \epsilon_t + a_1 \epsilon_{t-1} + ... + a_q \epsilon_{t-q} \tag{5.11}$$

  ARMA model can be used either purely as AR or MA model using ARMA$(p, 0)$ and ARMA$(0, q)$ respectively. ARMA is best suitable for stationary time series.
  The extension of the ARMA model is Auto-Regressive Integrated Moving Average (ARIMA) model is suitable for non-stationary time series. If ϵ (white noise) shows no pattern, then ARIMA(p, d, q) used where d represents the degree of difference.
- Machine learning techniques: Analysis of time series was carried out by many authors using machine learning-based techniques.
  Regression-based techniques are based on statistical methods to form a polynomial function, that find the nearest points from the history window w. Linear regression is ordered 1 polynomial expression. The distance of points should be as less as possible. Multiple Linear Regression is used when there is more than one variable in the expression.
  Neural Network is a group of interconnected artificial neurons put on multiple layers. Multiple inputs are put in the hidden layer, which further given an output. In time series one output against one input from the history window. Random weights are assigned to the input vector during the training phase. The weights are adapted to the optimized output has not achieved.
- Pattern Recognition: Time series data is a combination of seasonal and non-seasonal data. The time series has patterned with a specific time period (e.g. hours, day, month, year, or season) on the basis of short term and long term workload. It finds the matching pattern in the history that relates to the current pattern. It is very similar to the string matching technique [27].
- Signal Processing Techniques: This technique is based on harmonic analysis to decompose a time series signal into different frequencies. Fast Fourier Transformation and spectral density estimation filter the noise and estimating the signal value.
- Auto-correlation: In the linear regression errors are independent, the auto-correlation function is used to deal with the dependent error pattern. The input workload from the history window is shifted recursively, and further compared with the original time series.

The histogram is used to represent the time series. It splits the time series values into equal size bins, and each bin represents the frequency. In literature, it is used to represent the forecast values, resource distribution, and usage pattern.

5.7.1. Review of Articles. Time series analysis for multi-tier applications is most frequently used in the literature. A simple moving average model yields poor results, so that the MA used to remove time-series noise [76, 103]. The resource prediction model of Author [55] was developed using double exponential smoothing and a simple medium and Weighted Moving Average (WMA) applied. Because of w history records, ES significantly provides better results. Author [94] used the Dual ES of Brown to predict the load of work and obtain good results with a small error in the HTTP workload. Author [72] applied neural network and differential evolution for workload prediction. The back propagation method give adequate accuracy. Furthermore, the accuracy of the model can be improved with the pattern classification technique.

The auto-regression models applied for [21, 22, 46, 71, 114] workload and resource forecast. The author taking three previous observations used the AR model to estimate workload [114]. Estimated additional response time from the forecast values. A resource allocation optimization controller applied, taking SLA violation costs, reconfiguration, and leasing resources into account. In order to forecast requests per second, Author [71] used AR(order 1) and concluded that its performance is very much based on many manager-determined parameters (e.g. history window, adaptation window, size, and monitoring interval). The forecast is based on short and long term trends, depending heavily on the size of the history window. Further, the model extension determines the adjustment window.

The ARMA Model foresees the future workload (number of requests) is simple and efficient. The usage of VMs on the CPU is predicted by Author [37]. In different article [117, 17, 93], the ARIMA model is applied. The historic workload was required in ARIMA. The model's performance depends greatly on its history. ARIMA is ideal for dynamic workload applications such as web applications. The horizontal and vertical scaling was applied to increase the cost-benefit [117]. The classification given is: increasing, stable, seasonally and on/off, used by the authors [89] VMs were repacked (or reconfigured) to provide certain capacities, and workload-based application repacking was performed. The approach then finds the ideal package of VMs. The approach proposed can save 7% to 60% for the use of the resource. With additional QoS parameters, the container based approach can be further optimized. For workload forecasts and evaluate impacts on various QoS-parameters, investigator [17] used ARIMA model. The workload of the web application is dynamic and includes seasonal information. For non-seasonal data, the model provides 91% accuracy but is not suited for a highly non-seasonal workload. This work can be further extended by means of an adaptive approach for working load classification and the heuristic design for ARIMA fit for various classes. As mentioned above, an author [93] presented the GA-based approach to time-series prediction do not fit in for all types of workload. The prediction model has been assessed using traces of real workload. A new metric, describing solution optimization, was introduced in the article called the Elasticity Index (EI). The EI range varies between 0 to 1, with a value of almost 1 the solution is good. Compared to other models, the model gives less error. This approach takes longer to predict the incoming request, even hourly prediction. In addition, the mapping of only few prediction techniques with a particular application pattern and workload pattern can be enlarged. In particular, GA models can design cost, energy, resource sharing, and parameters.

Author [125] performed the workload prediction using classification technique. The historical workload took under consideration of sliding window. The past 5 lags has considered for to fit the model and predict the future request. On the behalf of peak-to-mean-ratio and $l2 - norm$ used to calculate the scale of time series. The model gave sufficient accuracy in terms of RMSE and MAPE for web application workload. There is no existing model which can capture all types of pattern. The error rate is still more than 10%. The model accuracy can be improved with application profiling. Author [69] designed the CloudInsight approach using regression, time series and machine learning techniques. The predictor pool has built and select the prediction model according to the testing phase output.

The precise of neural networks [58, 106] are highly dependent on the size of the input in the history window in several regression equations [14, 106, 71]. Author [58] have used more than one historical value and achieved a better result. The need for a balanced size of the input history window is defined by the author [71]. Regression of different window sizes used to locate forecast values. Another important factor is the r prediction interval. The size of the interval window is examined and found 12 minutes at the right point due to the startup time of VM being between 5 and 15 minutes. The neural network applied to 2 minutes to forecast the gaming load by the author [106]. The neural network in terms of accuracy, however, is better than MA and ES.

The literary authors also focused on horizontal and vertical scaling. It can be taken individually or in a hybrid manner. The researcher [34] investigated that horizontal scaling costs are higher than vertical scaling, but also relatively high throughput. The author prefers to scale horizontally. Model of regression to estimate the future workload. The author [37] concentrated on the flash crowd's horizontal scaling (CPU and memory), while regular changes are made with vertical scaling.

Proactive time series forecasting and a reactive approach can be combined. The author uses a reactive strategy for scale-up and scale-down scale model and developed a model of hybrids that can be self-sized [57]. Polynomial regression is used to determine the number of instances of application and database VMs.

Some authors have also applied a pattern identification method on time series analysis [19, 46, 122]. FFT-based techniques for identifying patterns matching the use of resources (RAM, CPU, I/O, and network) are proposed [46]. The comparison is done using auto-correlation, histogram, and auto-regression. The authors [19] have developed the algorithm with more parameters and poor performance.

The use of application resources in some articles is also estimated by the use of the mean distribution [21] and, the highest frequency with simple histogram [46]. The current usage and historical data [29], the dynamic load balancer is introduced by the Holt's technique. This can be used with web workloads and able to give efficiency in prediction. This can solve problems with load balancing.

Techniques for time series analysis can predict future web applications workload. Moreover, resource requirements can be predicted through this information. This approach is very attractive because the auto-scaler is known to be able to prepare the VMs in advance. The inconvenience of techniques is precision depending on workload, selection of historical windows, measurements, the interval of prevision and the target application. The time series forecasting exists no best solution for all types of pattern. The future scope of the adaptive or GA-based time series forecast for a specific application, taking QoS parameters into account. Tables 5.9 and 5.10 show the taxonomy of reviewed articles in this section.

6. Future Directions.

As per the literature survey, still, there is scope for further improvement in the present auto-scaling solutions for the multi-tier web applications. The following sections describe important future directions in this field.

6.1. Monitoring Tools.

Multi-tier web applications are installed on the servers. Cost effective monitoring tools are required to implement, which explore the hidden parameters such as cache memory, service time and type of request (e.g. Compute intensive and data intensive). Application and workload-specific monitoring interval can further improve the auto-scaler results.

6.2. Pricing Model.

Companies such as Amazon, Google and Microsoft have their different pricing model. Most of the researchers consider auto-scaling techniques on on-demand resources while considering the unlimited resources. Other types of model are also introduced, for example, Spot instances by the Amazon. The cost of such resources is very less as compare to on-demand resources. Very few authors start working on a spot instance, so the area is very immature and have different research challenges. Auto-scaling technique for spot-instances using reactive and proactive techniques give benefit to cloud providers and clients.

6.3. Resource Allocation.

While allocating the resource for input workload, the information of data center resources is also required. Some parameters such as CPU, RAM, Disk are well-known factors considered, but some parameters such as cache memory, network bandwidth, and fault tolerance are least considered in the literature. VMs for input workload for different tiers of the web application is the future scope of many articles. SLA base resource allocation with different QoS parameters need improvements. It can be further improved by extending the queuing network model.

6.4. Horizontal and vertical Scaling.

Horizontal scaling is used in most of the articles. Operating system and cloud architecture support horizontal scaling. The vertical scaling is easier in cloud infrastructure and gives better cost benefit. Hypervisors and operating system support could be enhanced for the vertical scaling. Energy and cost-effective VMs allocation, and consolidated migration are future areas for research.

6.5. Workload Predictor.

The existing workload predictors are considering the historical workload. Flash workload is hard to predict from the past workload. As growth in the online data mining and deep learning techniques, real-time data can be used to avoid the sudden burst condition in the data center. Web applications face this problem due to any hackers attack (DDOS) or flash event. Categorization of application will be more helpful to handle the flash crowd. Spot instances can be more effective in terms of cost optimization to serve flash workload.

6.6. Multi-cloud Auto-scaling.

Multiple cloud providers are supporting multi-tier web architecture. Cloud providers individually proving the reliability of their services. Multi-cloud architecture for web applications needs to work upon. The cloud provider can be selected by considering the application feature as the type of application, input workload, geographical area, etc. The reliability-aware auto-scaling in the multi-cloud environment by factorizing the various parameters.

Table 5.9
Taxonomy on time-series analysis based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[21]	Proactive	Horizontal	TSA: AR(1) and Histogram + QTH	Resource Aware	Simulated. 1, 5, 10 and 20 minutes	Response time	Request rate and service demand	On- demand	Custom simulator + algorithms in Matlab
[22]	Proactive	Horizontal	TSA:	AR Energy Aware	Simulated	Service not available (login), energy consumption	Login rate, number of active connections, CPU load	On- demand	Simulator
[106]	Proactive	Horizontal	TSA: ML Neural Network (compared to MA, last value and simple ES)	QoS based	2 minutes	Prediction accuracy	Number of entities (players)	On- demand	Simulator of a MMORPG game
[46]	Proactive	Vertical	TSA: FFT and Discrete Markov Chains. Compared with auto-regression, auto-correlation, histogram, max and min.	SLO based	Libxenstat library. 1 minute	Response time	CPU load	On- demand	Custom testbed. Xen + RUBiS + part of Google Cluster Data trace for CPU usage
[94]	Proactive	-	TSA: Brown's double ES	Performance based	10 minutes	-	Number of requests per VM	On- demand	Custom testbed. TPC-W
[66]	Proactive	Both	TSA: AR + TR	Resource Aware	Zabbix	-	Number of requests	On- demand	Hybrid: Amazon EC2 + Custom testbed (Xen + Eucalyptus + PhpCollab application)
[19]	Proactive	Horizontal	TSA: Pattern matching	Workload Aware	100 seconds	Number of serviced requests, cost	Total number of CPUs	On- demand	Analytical models
[122]	Hybrid	Vertical	TSA: FFT and Discrete-time Markov Chain	SLO Aware	Libxenstat library. 1 second	Response time, job progress	CPU load, memory usage	On- demand	-
[114]	Proactive	Horizontal	TSA: AR	QOS Aware	-	Response time, VM cost, application reconfiguration cost	Number of users in the system	On- demand	No experimentation on systems
[57]	Hybrid	Horizontal	TR (scale out) + TSA, polynomial regression (scale in)	SLA based	1 minute	Response time	CPU load (scale out), number of requests, number of VMs (scale in)	On- demand	-
[37]	Proactive	Both	TSA: ARMA	SLA based	-	Prediction accuracy	Number of requests, CPU load	On- demand	Custom testbed. Xen and KVM
[55]	Proactive	-	TSA: Brown's double ES. Compared with WMA	Cost-Aware	Simulated	Prediction accuracy	CPU load, memory usage	On- demand	Custom testbed. TPC-W

Table 5.10
Taxonomy on time-series analysis based reviewed literature (continuation)

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[58]	Proactive	Horizontal	TSA: ML Neural Network and (Multiple) LR + Sliding window	Resource Aware	Amazon CloudWatch. 1 minute	Prediction accuracy	CPU load (aggregated value for all VMs)	On- demand	Real provider. Amazon EC2 and TPC-W application to generate the dataset and R-Project.
[34]	Proactive	Both	TSA: Polynomial regression	Resource/ Cost Aware	Custom tool. 1 minute	Response time, cost for VM, application license and reconfiguration	Number of requests	On- demand	Custom testbed. KVM + Olio
[117]	Hybrid	Both	TSA: ARIMA + Repacking	Cost-Aware	4 and 20 minutes	Response time	Number of request, cost	On- demand	Custom simulator
[39]	Hybrid	Horizontal	TSA + Profiler	QoS Aware	5 minutes	Response time	Number of request, CPU usage and throughput	On- demand	-
[17]	Proactive	Horizontal	TSA: ARIMA	QoS aware	simulated	Response time	Request rate, RMSE	On- demand	CloudSim toolkit
[93]	Proactive	Horizontal	TSA: ARIMA + GA	Resource Aware	simulated	Response time, Cost	Request rate, Bootstrap, Mean Elasticity Index (MEI), RMSE, MSE	On- demand	Custom testbed in cloud
[29]	Proactive	Horizontal	TSA: Holt model + Reverse trend (RT) + GA + Fuzzy logic	Load Balance aware	controlled environment	Execution time, Number of migrations	Average and standard error and Breakdown	On- demand	Custom testbed using two clusters with Globus toolkit
[6]	Proactive	Horizontal	TSA: Double Exponential Smoothing (DES)	Cost aware	Custom	Resources and SLA	Number of requests	On- demand	CloudSim
[7]	Hybrid	Horizontal	TSA: Weight moving average (WMA)	cost aware	Custom	cost	Number of requests + CPU utilization	On- demand	CloudSim
[72]	Proactive	-	TSA: ANN + Differential evolution	Workload aware	Prediction interval (1, 5, 10, 20, 30, 60)	-	Arrival rate, RMSE	On-demand	Sumlation in Matlab. NASA and Saskatchewan traces
[69]	Proactive	-	TSA: CloudInsight	Workload aware	-	-	No. of requests, Prediction overhead, Commutative distribution function (CDF)	-	Python 2.7 on Ubuntu 16.07. Google workload, Wiki, Facebook dataset
[125]	Proactive	Horizontal	TSA: LR, ARIMA and SVR	Resource aware	Custom	Prediction accuracy and resources	Number of request	On- demand	R-tool
[61]	Hybrid	Horizontal	TSA: QTH + Continuous Time Markov Chain (CTMC) + AR	Resource aware	Custom	Response time	Arrival rate, VMs allocated, Resource utilization	On-demand	Real experiments on Amazon EC2. RUBiS, RUBBoS, Cassandra and Olio benchmarking applications

6.7. Energy aware Auto-scaling. The existing work mostly focused on cost optimization and QoS requirement. The data center is also facing environmental issues also. Solar datacenters and renewable energy resources in the data center can reduce the carbon problem. So the carbon and energy-aware auto-scaling provide the preference to the environmental friendly data centers in a single and multi-cloud environment. The weather condition and maximum solar power generating data centers give the highest priority for resource allocation.

6.8. Bin-packing Auto-scaling. The bin packing approach offers potential research topics in auto-scaling of web applications in cloud computing. The size of bins could vary, which makes this approach more robust. Resource provisioning of bins is lightweight and gives cost benefits.

7. Conclusions. Auto-scaling technique provisions the resources as per the incoming workloads. The application providers reap the benefit with this technique in terms of cost while maintaining the QoS. However, the auto-scaling design and development process having number of challenges. In literature, authors proposed the auto-scaling techniques with various characteristics in order to overcome the problems in auto-scaling in cloud.

In this article, a literature survey of the state-of-the-art auto-scaling techniques has carried out and, identified the key challenges in auto-scaling of multi-tier web applications. Moreover, the auto-scaling techniques are classified and taxonomy is presented based on various characteristics. Furthermore, the future research directions have been proposed based on the key challenges.

REFERENCES

[1] Ali-Eldin, A., Kihl, M., Tordsson, J., and Elmroth, E. (2012a). Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In *Proceedings of the 3rd workshop on Scientific Cloud Computing Date*, pages 31–40. ACM.

[2] Ali-Eldin, A., Tordsson, J., and Elmroth, E. (2012b). An adaptive hybrid elasticity controller for cloud infrastructures. In *2012 IEEE Network Operations and Management Symposium*, pages 204–212. IEEE.

[3] Alzubi, J., Nayyar, A., and Kumar, A. (2018). Machine learning from theory to algorithms: an overview. In *Journal of Physics: Conference Series*, volume 1142, page 012012. IOP Publishing.

[4] Amazon (2019). Amazon EC2. `https://aws.amazon.com/ec2/`. [Online; accessed 30-March-2019].

[5] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., et al. (2009). Above the clouds: A berkeley view of cloud computing.

[6] Aslanpour, M. S. and Dashti, S. E. (2017). Proactive auto-scaling algorithm (pasa) for cloud application. *International Journal of Grid and High Performance Computing (IJGHPC)*, 9(3):1–16.

[7] Aslanpour, M. S., Ghobaei-Arani, M., and Toosi, A. N. (2017). Auto-scaling web applications in clouds: a cost-aware approach. *Journal of Network and Computer Applications*, 95:26–41.

[8] Bacigalupo, D. A., van Hemert, J., Usmani, A., Dillenberger, D. N., Wills, G. B., and Jarvis, S. A. (2010). Resource management of enterprise cloud systems using layered queuing and historical performance models. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. IEEE.

[9] Barrett, E., Howley, E., and Duggan, J. (2013). Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, 25(12):1656–1674.

[10] Bauer, A., Herbst, N., Spinner, S., Ali-Eldin, A., and Kounev, S. (2019). Chameleon: A hybrid, proactive auto-scaling mechanism on a level-playing field. *IEEE Transactions on Parallel and Distributed Systems*, 30(4):800–813.

[11] Beltrán, M. (2015). Automatic provisioning of multi-tier applications in cloud computing environments. *The Journal of Supercomputing*, 71(6):2221–2250.

[12] Benifa, J. B. and Dejey, D. (2018). Rlpas: Reinforcement learning-based proactive auto-scaler for resource provisioning in cloud environment. *Mobile Networks and Applications*, pages 1–16.

[13] Bhardwaj, T. and Sharma, S. C. (2018). Fuzzy logic-based elasticity controller for autonomic resource provisioning in parallel scientific applications: a cloud computing perspective. *Computers & Electrical Engineering*, 70:1049–1073.

[14] Bodik, P., Griffith, R., Sutton, C., Fox, A., Jordan, M., and Patterson, D. (2009). Statistical machine learning makes automatic control practical for internet datacenters. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, pages 12–12.

[15] Brown, R. G. and Meyer, R. F. (1961). The fundamental theorem of exponential smoothing. *Operations Research*, 9(5):673–685.

[16] Bu, X., Rao, J., and Xu, C.-Z. (2013). Coordinated self-configuration of virtual machines and appliances using a model-free learning approach. *IEEE transactions on parallel and distributed systems*, 24(4):681–690.

[17] Calheiros, R. N., Masoumi, E., Ranjan, R., and Buyya, R. (2015). Workload prediction using arima model and its impact on cloud applications qos. *IEEE Transactions on Cloud Computing*, 3(4):449–458.

[18] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50.

[19] Caron, E., Desprez, F., and Muresan, A. (2011). Pattern matching based forecast of non-periodic repetitive behavior for cloud clients. *Journal of Grid Computing*, 9(1):49–64.

[20] Casalicchio, E. and Silvestri, L. (2013). Autonomic management of cloud-based systems: the service provider perspective. In *Computer and Information Sciences III*, pages 39–47. Springer.

[21] Chandra, A., Gong, W., and Shenoy, P. (2003). Dynamic resource allocation for shared data centers using online measurements. In *International Workshop on Quality of Service*, pages 381–398. Springer.

[22] Chen, G., He, W., Liu, J., Nath, S., Rigas, L., Xiao, L., and Zhao, F. (2008). Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *NSDI*, volume 8, pages 337–350.

[23] Chen, T., Bahsoon, R., and Yao, X. (2018). A survey and taxonomy of self-aware and self-adaptive cloud autoscaling systems. *ACM Computing Surveys (CSUR)*, 51(3):61.

[24] Chieu, T. C., Mohindra, A., and Karve, A. A. (2011). Scalability and performance of web applications in a compute cloud. In *e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on*, pages 317–323. IEEE.

[25] Chieu, T. C., Mohindra, A., Karve, A. A., and Segal, A. (2009). Dynamic scaling of web applications in a virtualized cloud computing environment. In *e-Business Engineering, 2009. ICEBE'09. IEEE International Conference on*, pages 281–286. IEEE.

[26] Cohen, J. W. (2012). *The single server queue*, volume 8. Elsevier.

[27] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). Introduction to algorithms second edition.

[28] Coutinho, E. F., de Carvalho Sousa, F. R., Rego, P. A. L., Gomes, D. G., and de Souza, J. N. (2015). Elasticity in cloud computing: a survey. *annals of telecommunications-annales des télécommunications*, 70(7-8):289–309.

[29] De Grande, R. E., Boukerche, A., and Alkharboush, R. (2017). Time series-oriented load prediction model and migration policies for distributed simulation systems. *IEEE Transactions on Parallel and Distributed Systems*, 28(1):215–229.

[30] Dejun, J., Pierre, G., and Chi, C.-H. (2011). Resource provisioning of web applications in heterogeneous clouds. In *Proceedings of the 2nd USENIX conference on Web application development*, pages 5–5. USENIX Association.

[31] Douglas, K. B. (2003). Web services and service-oriented architectures: the savvy manager's guide.

[32] Dutreilh, X., Kirgizov, S., Melekhova, O., Malenfant, J., Rivierre, N., and Truck, I. (2011). Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow. In *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*, pages 67–74.

[33] Dutreilh, X., Moreau, A., Malenfant, J., Rivierre, N., and Truck, I. (2010). From data center resource allocation to control theory and back. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 410–417. IEEE.

[34] Dutta, S., Gera, S., Verma, A., and Viswanathan, B. (2012). Smartscale: Automatic application scaling in enterprise clouds. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 221–228. IEEE.

[35] Evangelidis, A., Parker, D., and Bahsoon, R. (2018). Performance modelling and verification of cloud-based auto-scaling policies. *Future Generation Computer Systems*, 87:629–638.

[36] Fallah, M., Arani, M. G., and Maeen, M. (2015). Nasla: Novel auto scaling approach based on learning automata for web application in cloud computing environment. *International Journal of Computer Applications*, 113(2).

[37] Fang, W., Lu, Z., Wu, J., and Cao, Z. (2012). Rpps: a novel resource prediction and provisioning scheme in cloud data center. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pages 609–616. IEEE.

[38] Farokhi, S., Jamshidi, P., Lakew, E. B., Brandic, I., and Elmroth, E. (2016). A hybrid cloud controller for vertical memory elasticity: A control-theoretic approach. *Future Generation Computer Systems*, 65:57–72.

[39] Fernandez, H., Pierre, G., and Kielmann, T. (2014). Autoscaling web applications in heterogeneous cloud infrastructures. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 195–204. IEEE.

[40] Frey, S., Lüthje, C., Reich, C., and Clarke, N. (2014). Cloud qos scaling by fuzzy logic. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 343–348. IEEE.

[41] Gambi, A. and Toffetti, G. (2012). Modeling cloud performance with kriging. In *Proceedings of the 34th International Conference on Software Engineering*, pages 1439–1440. IEEE Press.

[42] Gandhi, A., Dube, P., Karve, A., Kochut, A., and Zhang, L. (2014). Adaptive, model-driven autoscaling for cloud applications. In *11th International Conference on Autonomic Computing (ICAC 14)*, pages 57–64.

[43] Gandhi, A., Harchol-Balter, M., Raghunathan, R., and Kozuch, M. A. (2012). Autoscale: Dynamic, robust capacity management for multi-tier data centers. *ACM Transactions on Computer Systems (TOCS)*, 30(4):14.

[44] Gandhi, A., Dube, P., Karve, A., Kochut, A., and Zhang, L. (2018). Model-driven optimal resource scaling in cloud. *Software & Systems Modeling*, 17(2):509–526.

[45] Ghanbari, H., Simmons, B., Litoiu, M., and Iszlai, G. (2011). Exploring alternative approaches to implement an elasticity policy. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 716–723. IEEE.

[46] Gong, Z., Gu, X., and Wilkes, J. (2010). Press: Predictive elastic resource scaling for cloud systems. In *2010 International Conference on Network and Service Management*, pages 9–16. IEEE.

[47] Grimaldi, D., Pescape, A., Salvi, A., Persico, V., et al. (2017). A fuzzy approach based on heterogeneous metrics for scaling out public clouds. *IEEE Transactions on Parallel and Distributed Systems*.

[48] Grozev, N. and Buyya, R. (2014). Multi-cloud provisioning and load distribution for three-tier applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 9(3):13.

[49] Guitart, J., Torres, J., and Ayguadé, E. (2010). A survey on performance management for internet applications. *Concurrency and Computation: Practice and Experience*, 22(1):68–106.

[50] Han, R., Ghanem, M. M., Guo, L., Guo, Y., and Osmond, M. (2014). Enabling cost-aware and adaptive elasticity of

multi-tier cloud applications. *Future Generation Computer Systems*, 32:82–98.

[51] Han, R., Guo, L., Ghanem, M. M., and Guo, Y. (2012). Lightweight resource scaling for cloud applications. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 644–651. IEEE.

[52] Hasan, M. Z., Magana, E., Clemm, A., Tucker, L., and Gudreddi, S. L. D. (2012). Integrated and autonomic cloud resource scaling. In *2012 IEEE Network Operations and Management Symposium*, pages 1327–1334. IEEE.

[53] Heinze, T., Pappalardo, V., Jerzak, Z., and Fetzer, C. (2014). Auto-scaling techniques for elastic data stream processing. In *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*, pages 296–302. IEEE.

[54] Huang, D., He, B., and Miao, C. (2014). A survey of resource management in multi-tier web applications. *IEEE Communications Surveys & Tutorials*, 16(3):1574–1590.

[55] Huang, J., Li, C., and Yu, J. (2012). Resource prediction based on double exponential smoothing in cloud computing. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 2056–2060. IEEE.

[56] Huang, G., Wang, S., Zhang, M., Li, Y., Qian, Z., Chen, Y., and Zhang, S. (2016). Auto scaling virtual machines for web applications with queueing theory. In *2016 3rd International Conference on Systems and Informatics (ICSAI)*, pages 433–438. IEEE.

[57] Iqbal, W., Dailey, M. N., Carrera, D., and Janecek, P. (2011). Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems*, 27(6):871–879.

[58] Islam, S., Keung, J., Lee, K., and Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155–162.

[59] Jamshidi, P., Pahl, C., and Mendonça, N. C. (2016). Managing uncertainty in autonomic cloud elasticity controllers. *IEEE Cloud Computing*, 3(3):50–60.

[60] Jin, Y., Bouzid, M., Kostadinov, D., and Aghasaryan, A. (2018). Testing a q-learning approach for derivation of scaling policies in cloud-based applications. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 1–3. IEEE.

[61] JV, B. B. and Dharma, D. (2018). Has: Hybrid auto-scaler for resource scaling in cloud environment. *Journal of Parallel and Distributed Computing*, 120:1–15.

[62] Kalyvianaki, E., Charalambous, T., and Hand, S. (2009). Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In *Proceedings of the 6th international conference on Autonomic computing*, pages 117–126. ACM.

[63] Kaur, G., Bala, A., and Chana, I. (2019). An intelligent regressive ensemble approach for predicting resource usage in cloud computing. *Journal of Parallel and Distributed Computing*, 123:1–12.

[64] Keilson, J. and Servi, L. (1988). A distributional form of little's law. *Operations Research Letters*, 7(5):223–227.

[65] Kendall, D. G. (1953). Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354.

[66] Khatua, S., Ghosh, A., and Mukherjee, N. (2010). Optimizing the utilization of virtual resources in cloud environment. In *2010 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, pages 82–87. IEEE.

[67] Kim, H., el Khamra, Y., Jha, S., and Parashar, M. (2009a). An autonomic approach to integrated hpc grid and cloud usage. In *e-Science, 2009. e-Science'09. Fifth IEEE International Conference on*, pages 366–373. IEEE.

[68] Kim, H., Parashar, M., Foran, D. J., and Yang, L. (2009b). Investigating the use of autonomic cloudbursts for high-throughput medical image registration. In *2009 10th IEEE/ACM International Conference on Grid Computing*, pages 34–41. IEEE.

[69] Kim, I. K., Wang, W., Qi, Y., and Humphrey, M. (2018). Cloudinsight: Utilizing a council of experts to predict future cloud application workloads. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 41–48. IEEE.

[70] Koperek, P. and Funika, W. (2011). Dynamic business metrics-driven resource provisioning in cloud environments. In *International Conference on Parallel Processing and Applied Mathematics*, pages 171–180. Springer.

[71] Kupferman, J., Silverman, J., Jara, P., and Browne, J. (2009). Scaling into the cloud. *CS270-advanced operating systems.*

[72] Kumar, J. and Singh, A. K. (2018). Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems*, 81:41–52.

[73] Lama, P. and Zhou, X. (2009). Efficient server provisioning with end-to-end delay guarantee on multi-tier clusters. In *Quality of Service, 2009. IWQoS. 17th International Workshop on*, pages 1–9. IEEE.

[74] Lama, P. and Zhou, X. (2013). Autonomic provisioning with self-adaptive neural fuzzy control for percentile-based delay guarantee. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 8(2):9.

[75] Lango, J. (2014). Toward software-defined slas. *Communications of the ACM*, 57(1):54–60.

[76] Lim, H. C., Babu, S., and Chase, J. S. (2010). Automated control for elastic storage. In *Proceedings of the 7th international conference on Autonomic computing*, pages 1–10. ACM.

[77] Lim, H. C., Babu, S., Chase, J. S., and Parekh, S. S. (2009). Automated control in cloud computing: challenges and opportunities. In *Proceedings of the 1st workshop on Automated control for datacenters and clouds*, pages 13–18. ACM.

[78] Liu, C., Liu, C., Shang, Y., Chen, S., Cheng, B., and Chen, J. (2017). An adaptive prediction approach based on workload pattern discrimination in the cloud. *Journal of Network and Computer Applications*, 80:35–44.

[79] Liu, X., Yuan, S.-M., Luo, G.-H., Huang, H.-Y., and Bellavista, P. (2017). Cloud resource management with turnaround time driven auto-scaling. *IEEE Access*, 5:9831–9841.

[80] Liu, B., Buyya, R., and Toosi, A. N. (2018a). A fuzzy-based auto-scaler for web applications in cloud computing environ-

ments. In *International Conference on Service-Oriented Computing*, pages 797–811. Springer.

[81] Liu, X., Dastjerdi, A. V., Calheiros, R. N., Qu, C., and Buyya, R. (2018b). A stepwise auto-profiling method for performance optimization of streaming applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(4):24.

[82] Liu, X., Dastjerdi, A. V., Calheiros, R. N., Qu, C., and Buyya, R. (2018). A stepwise auto-profiling method for performance optimization of streaming applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(4):24.

[83] Lombardi, F., Muti, A., Aniello, L., Baldoni, R., Bonomi, S., and Querzoni, L. (2019). Pascal: An architecture for proactive auto-scaling of distributed services. *Future Generation Computer Systems*.

[84] Lorido-Botrán, T., Miguel-Alonso, J., and Lozano, J. A. (2012). Auto-scaling techniques for elastic applications in cloud environments. *Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09-12*.

[85] Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, 12(4):559–592.

[86] Lorido-Botrcn, T., Miguel-Alonso, J., and Lozano, J. A. (2013). Comparison of auto-scaling techniques for cloud environments.

[87] Mahallat, I. (2015). Astaw: Auto-scaling threshold-based approach for web application in cloud computing environment.

[88] Manvi, S. S. and Shyam, G. K. (2014). Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *Journal of Network and Computer Applications*, 41:424–440.

[89] Mao, M. and Humphrey, M. (2011). Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 1–12. IEEE.

[90] Mao, M. and Humphrey, M. (2012). A performance study on the vm startup time in the cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 423–430. IEEE.

[91] Maurer, M., Brandic, I., and Sakellariou, R. (2011a). Enacting slas in clouds using rules. In *European Conference on Parallel Processing*, pages 455–466. Springer.

[92] Maurer, M., Breskovic, I., Emeakaroha, V. C., and Brandic, I. (2011b). Revealing the mape loop for the autonomic management of cloud infrastructures. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 147–152. IEEE.

[93] Messias, V. R., Estrella, J. C., Ehlers, R., Santana, M. J., Santana, R. C., and Reiff-Marganiec, S. (2016). Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure. *Neural Computing and Applications*, 27(8):2383–2406.

[94] Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., and Yuan, L. (2010). Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 514–521. IEEE.

[95] Moghaddam, S. K., Buyya, R., and Ramamohanarao, K. (2019). Acas: An anomaly-based cause aware auto-scaling framework for clouds. *Journal of Parallel and Distributed Computing*, 126:107–120.

[96] Nayyar, A. (2011). Private virtual infrastructure (pvi) model for cloud computing. *International Journal of Software Engineering Research and Practices*, 1(1):10–14.

[97] Nayyar, A., Puri, V., et al. (2017). Comprehensive analysis & performance comparison of clustering algorithms for big data. *Review of Computer Engineering Research*, 4(2):54–80.

[98] Nguyen, H., Shen, Z., Gu, X., Subbiah, S., and Wilkes, J. (2013). Agile: Elastic distributed resource scaling for infrastructure-as-a-service. In *ICAC*, volume 13, pages 69–82.

[99] Nikravesh, A. Y., Ajila, S. A., and Lung, C.-H. (2014). Cloud resource auto-scaling system based on hidden markov model (hmm). In *Semantic Computing (ICSC), 2014 IEEE International Conference on*, pages 124–127. IEEE.

[100] Nikravesh, A. Y., Ajila, S. A., and Lung, C.-H. (2017). An autonomic prediction suite for cloud resource provisioning. *Journal of Cloud Computing*, 6(1):3.

[101] Nouri, S. M. R., Li, H., Venugopal, S., Guo, W., He, M., and Tian, W. (2019). Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications. *Future Generation Computer Systems*, 94:765–780.

[102] Padala, P., Hou, K.-Y., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., and Merchant, A. (2009). Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 13–26. ACM.

[103] Park, S.-M. and Humphrey, M. (2009). Self-tuning virtual machines for predictable escience. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 356–363. IEEE Computer Society.

[104] Patikirikorala, T. and Colman, A. (2010). Feedback controllers in the cloud. In *Proceedings of APSEC*.

[105] Persico, V., Grimaldi, D., Pescape, A., Salvi, A., and Santini, S. (2017). A fuzzy approach based on heterogeneous metrics for scaling out public clouds. *IEEE Transactions on Parallel and Distributed Systems*, 28(8):2117–2130.

[106] Prodan, R. and Nae, V. (2009). Prediction-based real-time resource provisioning for massively multiplayer online games. *Future Generation Computer Systems*, 25(7):785–793.

[107] Qu, C., Calheiros, R. N., and Buyya, R. (2016a). Auto-scaling web applications in clouds: a taxonomy and survey. *arXiv preprint arXiv:1609.09224*.

[108] Qu, C., Calheiros, R. N., and Buyya, R. (2016b). A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances. *Journal of Network and Computer Applications*, 65:167–180.

[109] Qu, C., Calheiros, R. N., and Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Computing Surveys (CSUR)*, 51(4):73.

[110] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[111] Rao, J., Bu, X., Xu, C.-Z., and Wang, K. (2011). A distributed self-learning approach for elastic provisioning of virtualized cloud resources. In *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 45–54. IEEE.

[112] Rao, J., Bu, X., Xu, C.-Z., Wang, L., and Yin, G. (2009). Vconf: a reinforcement learning approach to virtual machines auto-configuration. In *Proceedings of the 6th international conference on Autonomic computing*, pages 137–146. ACM.

[113] RightScale (2019). RightScale Universal Cloud Management. `http://www.rightscale.com/`. [Online; accessed 30-March-2019].

[114] Roy, N., Dubey, A., and Gokhale, A. (2011). Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 500–507. IEEE.

[115] Runsewe, O. and Samaan, N. (2018). Cloud resource scaling for time-bounded and unbounded big data streaming applications. *IEEE Transactions on Cloud Computing*.

[116] Salah, K., Elbadawi, K., and Boutaba, R. (2016). An analytical model for estimating cloud resources of elastic services. *Journal of Network and Systems Management*, 24(2):285–308.

[117] Sedaghat, M., Hernandez-Rodriguez, F., and Elmroth, E. (2013). A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, page 6. ACM.

[118] Shah, N. B., Shah, N. D., Bhatia, J., and Trivedi, H. (2019). Profiling-based effective resource utilization in cloud environment using divide and conquer method. In *Information and Communication Technology for Competitive Strategies*, pages 495–508. Springer.

[119] Sharma, U., Shenoy, P., Sahu, S., and Shaikh, A. (2011). A cost-aware elasticity provisioning system for the cloud. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 559–570. IEEE.

[120] Shah, N. B., Shah, N. D., Bhatia, J., and Trivedi, H. (2019). Profiling-based effective resource utilization in cloud environment using divide and conquer method. In *Information and Communication Technology for Competitive Strategies*, pages 495–508. Springer.

[121] Shekhar, S., Barve, Y., and Gokhale, A. (2017). Understanding performance interference benchmarking and application profiling techniques for cloud-hosted latency-sensitive applications. In *Proceedings of the10th International Conference on Utility and Cloud Computing*, pages 187–188. ACM.

[122] Shen, Z., Subbiah, S., Gu, X., and Wilkes, J. (2011). Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 5. ACM.

[123] Simmons, B., Ghanbari, H., Litoiu, M., and Iszlai, G. (2011). Managing a saas application in the cloud using paas policy sets and a strategy-tree. In *Proceedings of the 7th International Conference on Network and Services Management*, pages 343–347. International Federation for Information Processing.

[124] Singh, S. P., Nayyar, A., Kumar, R., and Sharma, A. (2018). Fog computing: from architecture to edge computing and big data processing. *The Journal of Supercomputing*, pages 1–36.

[125] Singh, P., Gupta, P., and Jyoti, K. (2018). Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. *Cluster Computing*, pages 1–15.

[126] Sutton, R. S. and Barto, A. G. (1998). *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge.

[127] Tesauro, G., Jong, N. K., Das, R., and Bennani, M. N. (2006). A hybrid reinforcement learning approach to autonomic resource allocation. In *2006 IEEE International Conference on Autonomic Computing*, pages 65–73. IEEE.

[128] Toffetti, G., Brunner, S., Blöchlinger, M., Spillner, J., and Bohnert, T. M. (2017). Self-managing cloud-native applications: Design, implementation, and experience. *Future Generation Computer Systems*, 72:165–179.

[129] Toosi, A. N., Son, J., Chi, Q., and Buyya, R. (2019). Elasticsfc: Auto-scaling techniques for elastic service function chaining in network functions virtualization-based clouds. *Journal of Systems and Software*.

[130] Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., and Wood, T. (2008). Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1):1.

[131] Varia, J. and Mathew, S. (2014). Overview of amazon web services. *Amazon Web Services*.

[132] Vasić, N., Novaković, D., Miučin, S., Kostić, D., and Bianchini, R. (2012). Dejavu: accelerating resource allocation in virtualized environments. In *ACM SIGARCH computer architecture news*, volume 40, pages 423–436. ACM.

[133] Vilaplana, J., Solsona, F., Teixidó, I., Mateo, J., Abella, F., and Rius, J. (2014). A queuing theory model for cloud computing. *The Journal of Supercomputing*, 69(1):492–507.

[134] Villela, D., Pradhan, P., and Rubenstein, D. (2007). Provisioning servers in the application tier for e-commerce systems. *ACM Transactions on Internet Technology (TOIT)*, 7(1):7.

[135] Vondra, T. and Šedivỳ, J. (2017). Cloud autoscaling simulation based on queueing network model. *Simulation Modelling Practice and Theory*, 70:83–100.

[136] Wang, L., Xu, J., Zhao, M., Tu, Y., and Fortes, J. A. (2011). Fuzzy modeling based resource management for virtualized database systems. In *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 32–42. IEEE.

[137] Wei, Y., Kudenko, D., Liu, S., Pan, L., Wu, L., and Meng, X. (2019). A reinforcement learning based auto-scaling approach for saas providers in dynamic cloud environment. *Mathematical Problems in Engineering*, 2019.

[138] Xing, J., Zhou, H., Shen, J., Zhu, K., Wangt, Y., Wu, C., and Ruan, W. (2018). Asidps: Auto-scaling intrusion detection and prevention system for cloud. In *2018 25th International Conference on Telecommunications (ICT)*, pages 207–212. IEEE.

[139] Xu, C.-Z., Rao, J., and Bu, X. (2012). Url: A unified reinforcement learning approach for autonomic cloud management.

Journal of Parallel and Distributed Computing, 72(2):95–105.

[140] Xu, J., Zhao, M., Fortes, J., Carpenter, R., and Yousif, M. (2007). On the use of fuzzy modeling in virtualized data center management. In *Fourth International Conference on Autonomic Computing (ICAC'07)*, pages 25–25. IEEE.

[141] Zhang, Q., Cherkasova, L., and Smirni, E. (2007). A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In *Fourth International Conference on Autonomic Computing (ICAC'07)*, pages 27–27. IEEE.

[142] Zhu, Q. and Agrawal, G. (2010). Resource provisioning with budget constraints for adaptive applications in cloud environments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 304–307. ACM.

Edited by: Pijush Kanti Dutta Pramanik
Received: Mar 17, 2019
Accepted: Apr 4, 2019

DOI 10.12694/scpe.v20i2.1538
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 433–456. http://www.scpe.org © 2019 SCPE

DYNAMIC TASK SCHEDULING USING BALANCED VM ALLOCATION POLICY FOR FOG COMPUTING PLATFORMS

SIMAR PREET SINGH[*], ANAND NAYYAR[†], HARPREET KAUR[‡], AND ASHU SINGLA[§]

Abstract. The fog computing models are getting popular as the demand and capacity of data processing is rising for the various applications every year. The fog computing models incorporate the various task scheduling algorithms for the resource selection among the given list of virtual machines (VMs). The task scheduling models are designed around the various task metrics, which include the task length (time), energy, processing cost etc. for the various purposes. The cost oriented scheduling models are primarily built for the customer's perspectives, and saves them a handful amount of money by efficiently assigning the resources for the tasks. In this paper, we have worked upon the multiple task scheduling models based upon the Local Regression (LR), Inter Quartile Range (IQR), Local Regression Robust (LRR), Non-Power Aware (NPA), Median Absolute Deviation (MAD), Dynamic Voltage and Frequency Scheduling (DVFS) and The Static Threshold (THR) methods using the ifogsim simulation designed with the 50 nodes and 50 virtual machines, i.e. 1 virtual machine per node. All of the models have been implemented using the standard input simulation parameters for the purpose of performance assessment in the various domains, specifically in the time domain and effective consumption of energy. The results obtained from the experiments have shown the overall time of 86,400 seconds during the simulation, where the DVFS has been recorded with the 52.98 kWh consumption of energy, which shows the efficient processing in comparison to the 150.68 kWh of energy consumption in the NPA model. Also, there are no SLA violations recorded during both of the simulation, because no VM migration model has been utilized among both of the implemented models, which clearly shows that the VM migrations are the major cause of SLA violation cases. The LRR (2520 VMs) has been observed as best contender on the basis of mean of number of VM migrations in comparison with LR (2555 VMs), THR (4769 VMs), MAD (5138 VMs) and IQR (5352 VMs).

Key words: VM allocation, VM selection, fog computing, task scheduling, ifogsim simulator.

AMS subject classifications. 68M14, 90B35

1. Introduction.

In this era, the cloud computing applications are getting popular and more online applications are opting for the cloud computing platforms to effectively execute, manage and optimize the applications [1, 2]. The cloud computing environments provide the flexible application hosting plans, which are primarily divided in three infrastructural variants: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [3, 4, 5, 6].

The SaaS plans offer the hosting of software or application without worrying about the platform and infrastructure related operations, whereas PaaS plans enable the user to take full control over the operating system environment, and can effectively optimize the application performance on the platform level [7, 8]. On the other hand, the IaaS service includes the internal network of various systems (particularly VMs in this case) altogether, which are used to run the applications with high user count. Cloud computing grids are owned by the cloud operators, and is implemented in few grids across the world [3, 9, 10]. Because the cloud computing infrastructure is quite expensive, it is always implemented in form of small number of grids across the globe and provides a high-performance service with abundance of processing resources, i.e. CPU, RAM and storage. When cloud computing is known for a processing powerhouse, it has one primary disadvantage, which is associated with communication cost (i.e. the extra time delay to transfer the request and request-reply between the cloud & end user) [11, 12, 13, 14].

As described the primary disadvantage of cloud computing in the form of communication cost is the preference of extending the cloud computing services on the edge (the computing on the edge). There are several extensions of the cloud computing services, which forms fog computing, edge computing and content delivery networks (CDNs) [15, 16, 17, 18, 19]. The CDNs offer frequent data caching services, which enables the rapid delivery of frequently accessed data from the cloud resources. The frequently requested data is saved in the

[*]Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, India – corresponding author (dr.simarpreetsingh@gmail.com)
[†]Graduate School, Duy Tan University, Da Nang, Vietnam (anandnayyar@duytan.edu.vn)
[‡]Computer Science and Engineering Department, Chandigarh University, Mohali, Punjab, India (harpreet8307@gmail.com)
[§]Computer Science and Engineering Department, Sant Longowal Institute of Engineering and Technology, Longowal, Punjab, India (ashusinglaoct@gmail.com)

caching memory on the internet service providers (ISP) network, which does not offer any additional service [20, 21, 22]. For example, when you browse Facebook website on your PC or smart phone, most of the data associated with your profile and friends is loaded from the local CDN offered by ISP. The edge computing, on the other hand provides the distributed smart grid services, which enables the use of user end nodes to compute the data [23, 24]. The Search for Extraterrestrial Intelligence (SETI) project uses distributed smart grid over the internet by enabling the user nodes to process the satellite data in small chunks per node, and pretty well describes the concept of edge computing. On the contrary, the fog computing is the semi-centralized processing paradigm, which extends the cloud computing close to edge nodes [25, 26, 27]. The semi-centralized infrastructure is owned by cloud operators or its business associates to effectively offer the services with optimized and reduced communication cost, as well as extends the overall processing power of the cloud computing. Unlike, the edge computing and CDN, the fog computing offers the complete service package, which hosts the computing resources and offers computing, storage and event-based or need-based synchronization with primary cloud using synchronous or asynchronous archetypes [28, 29, 30, 31, 32].

In this paper, the proposed model is design and developed to effectively schedule the user tasks on the fog computing resources by combining the VM allocation and VM selection methods in the perfect arrangement. Various methods associated with VM allocation & VM selection are evaluated and combined in suitable combination to discover the best task scheduling combination for the effective and optimized user data processing.

The paper structure is as follows: This section (Sect. 1) discusses the introduction of cloud and fog computing technologies. Next section (Sect. 2) covers the literature review. Section 3 explains the decision parameters (Sect. 3.2) and the proposed algorithm (Sect. 3.3). Section 4 describes the results that are computed using the proposed approach. Finally, Sect. 5 describes the conclusion and future directions.

2. Related Work. Zhuo Tang et al. [33] proposed DVFS enabled Energy Efficient Workflow Task Scheduling algorithm (DEWTS). They used the scheduling order of all the tasks to obtain the makespan in their algorithm. The authors used different algorithms for computation of deadlines. In overall process, their proposed algorithm was able to reduce total power consumption by upto 46.5% for parallel applications. The authors worked on randomly generated workflows in their research work.

Yuan Fang et al. [34] discussed Cyber-Physical Systems (CPS) and proposed Simple and Proximate Time Model (SPTimo) framework. In addition to this, the authors also presented Mix Time Cost and Deadline First (MTCDF) time task scheduling algorithm, which was based on computation model of SPTimo framework. Their research provides an optimal scheduling solution in total time required and time cost parameters.

Zhao, Qing et al. [35] has implemented the energy-aware scheduling of the user tasks over the cloud computing resources. This scheme generates the task binary tree based upon task correlation, which is used to prioritize the user tasks. The authors proposed the Task Requirement Degree (TRD) based calculation method for proficient scheduling, where it also considers the bandwidth to optimize the communication cost.

Nidhi Bansal et al. [36] designed the QoS enabled optimized cost-based scheduling methodology. The authors have focused upon the cost of computing resources (virtual machines) to schedule the given pool of the tasks over the cloud computing model. The cost optimization has been performed over the QoS-task driven task scheduling mechanism, which did not encounter the cost optimization problem earlier. The authors have shown that the earlier QoS-driven task scheduling based studies has been considered the makespan, latency and load balancing. The QoS-based cost evaluation model evaluates the resource computing cost for the scheduling along with the other parameters as in their secondary precedence.

Gaurang patel et al. [37] have worked upon enhancement in the existing algorithm of Min-Min (Minimum-minimum methodology) for scheduling on cloud platform. The authors have proposed the use of active load balancing in processing the tasks over the cloud environments. The authors have proposed the new method for the efficient processing of tasks over the given cloud environment known as the Enhanced Load Balanced Min-Min (ELBMM) algorithm. The authors have recovered the major drawback of the existing model of Min-Min algorithm, where sometimes the makespan and current resource utilization is not properly considered and the tasks is scheduled over the slow resource which causes the latency. In their research, they have effectively overcome the problem concerned with the Min-Min algorithm. The authors have proved their model better than the Min-Min and ELBMM model for the task scheduling. Also, the execution times has been reduced to the optimum levels, and better than the existing model.

Weiwei Chen et al. [38] have proposed the imbalanced metrics for the optimization of task clustering on scientific workflow data executions. The authors have examined the imbalanced nature of the task clustering during the runtime evaluation for the purpose of task clustering in depth. The authors have proposed the improvement to effectively evaluate the problem of runtime task imbalance. The authors have proposed an horizontal and vertical method for the evaluation of series of task clustering for the widely used scientific workflows. Their proposed model has utilized the in-depth metric values for the real time evaluation of their research model.

Xu et al. [39] has worked towards the load balancing of the user tasks, which considers the task partitioning on cloud. The load balancing methods are known to be effective for efficient user task processing on cloud resources, because clouds generally receive high volumes of user data. Y. Tan et. al. [40] worked on a novel scheduling technique for cloud models. The authors complimented the use of particle swarm optimization (PSO) model to analyze the scheduling performance in the terms of delay and resource consumption. An optimized weight based mutation criteria with adaptable indolence oriented methodology is deployed to optimize the scheduling performance. Additionally, this scheme offers the load balancing schema to effectively schedule the user tasks.

K. Li et al. [41] described the feasible resource expansion for centralized, de-centralized and semi-centralized computing platforms, which also involve the parallel processing paradigm. The scheduling problem is described as NP-hard problem, and suggested several feasible solutions to effectual scheduling of the allocated computing resources. The authors proposed the swarm optimization (ACO oriented solution) to deploy the load balancing as effective meta-heuristic scheduling elucidation for the cloud platforms by reducing the individual load and effectively distributing the tasks of multiple users altogether.

X. Luo et al. [42] proposed an algorithm for resource scheduling under cloud computing environment. It is different from the under conventional distributed computing domain because of the high scalability and heterogeneity of computing resources in cloud computing domain. In this paper, based on dynamic load balance, the authors has proposed a resource-scheduling algorithm. The different statistic transferring power and retard between nodes in cloud as well statistic-processing power of nodes in cloud is considered in this algorithm. To increase the efficiency of cloud computing and reduce the median response time of tasks, the algorithm selects the best node to fulfill the task. The simulation results show that the algorithm reduces the average response time of tasks.

N. Bessis et al. [43] discussed in their paper about the new technologies develop fast and their complexity becomes a crucial concern. One proven way to deal with improved complexity was to engage a self-organizing strategy. The many different strategies exists that deal with the load balancing problem but most of the problem are task oriented and it is, therefore, hard to differentiate. So, the researchers of the paper developed and implemented a generic architectural pattern, called self-initiative load balancing agents. It allocates the exchange of different algorithms, both sightful and dense ones, through plugging. In placing at different levels, different algorithms can be tested in combination. The objective was simplicity in the selection of optimal algorithm for a definite problem. Self-initiative load balancing agent was the concern and domain independent, and can be collected towards inconsistent network topologies.

A. Jain and R. Singh [44] described grid computing for classification of non-identical resources that are cast off as virtual resource to a user and impart superior grid domain. Now-a-days, large amount of resource management in peer-to-peer grid environment is used. Load balancing is crucial concern to balance the overall load of the nodes. There are numbers of solutions to achieve load equality state. ACO is used to provide optimal solution for solving a problem of load balancing. In the paper, the authors has proposed Master-Ant Colony Optimization algorithm (M-ACO), and it is used in peer-to-peer environment. The proposed algorithm gives better results in peer-to-peer environment. MATLAB simulation tool was used, which provides different kinds of functions to bloom heuristic algorithms with new notions.

R. Chaukwale et al. [45] discussed the complication of efficiently scheduling jobs on several devices, it is a vital consideration when operating the Job Shop Production (JSP) scheduling system. JSP was a NP hard difficulty. The procedures that focus on fabricating an exact solution of the problem can evince insufficiency in discovering an optimal solution of the problem to Job Shop Production system (JSP). Hence, in such conditions, heuristic methods can be developed to discover a good solution of the problem within reasonable time period.

In their paper, the authors studied the traditional ACO algorithm and has proposed a load balancing ACO algorithm for JSP. The paper also presented the observed results. It was noticed that the proposed algorithm showed better outcomes when compared to traditional ACO. Many researches [46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 70, 71] discussed about scheduling and allocation methods in fog and cloud environments.

After going through the related work, it was found that with the increase of Internet of Things (IoT) devices, sensors, fog devices, actuators etc., lots of data is getting generated. This will lead to network congestion in coming future. Thus, there is a huge need to schedule and allocate the tasks, that are dynamic in nature, in a proper planned/optimal manner. This research work tries to simplifies the future arising problems in the area of fog computing.

3. Methods and Materials. The fog scheduling solution proposed in this paper is implemented using the ifogsim simulator considering the fog environment. Ifogsim simulator is based upon cloudsim platform for cloud infrastructure simulations. The proposed scheme combines VM allocation & VM selection procedures with performance optimization methods to boost the cloud's capability for user task processing. An idle process sequencing algorithm should be aimed at reducing the overall tasking overhead, tasking time (task completion time) and communication overhead by the whole task considering the incoming and outgoing information. The task management faces the major challenges from the bias-free dynamic resource allocation while keeping the cloud performance to the maximum in terms of execution time and computational overheads. This scheme offers the load balanced paradigm over user task stack, coupled with environmental parameter optimization, and enhances the endurance and general capability of the cloud environment.

3.1. Proposed Approach. The link optimization algorithm is designed as an intelligent solution influenced by behavior of the real Internet of Things (IoT) inter-nodal relations in scenario of increasing number of IoT nodes. A collaboration of IoT nodes in finding the appropriate paths and doing other tasks has been prioritized to achieve the link behavior in cloud systems. The fog resources store the usability for path devising and following while taking a movement from source node to the destination computing resource on cloud environment. With the raise in the number of requests on a singular path, the strength of connection increases on that particular path. The requests of that group select the shortest path on the basis of this usability index. The IoT connection request province optimization method has been applied for resolving the problem of rising number of requests, with the target of discovering the shortest path. The algorithm fully depends upon the history of usability index to take further judgments for optimal solutions for any of the computational requirement. The use of artificial links for the state of development rule and for the selection of optimal resources beyond the grid computation or the cloud environments has been proposed in the prospective work. The artificial links have been used for the purpose of cloud computing scheduling and shortest path identification. The link province system adopts the arbitrary-proportional rule, which is the state of transition rule used for link optimization system and works on the basis of probability or a chance to choose the optimal resource out of k-resources for task assignment in the cloud. The usability index of a resource depends upon the number of available resources, processing cost and estimated time. The VM load has been selected as the prime factor out of all these three factors; hence the computing decision is computed after verifying the cumulative and individual runtime VM load. The usability indexes are regularly updated using particular cloud resources or VMs selected for the act of scheduling. The shortest path is computed after analyzing the runtime parameters, which effectively analyzes the load, availability, communication cost and processing delay of a virtual machine. The VM runtime parameters are procured and continuously updated, and helps the scheduling decision on the cloud systems. Fig. 3.1 describes the shortest path in distributed and/or segmented sub-paths, and explains the Eqs. 3.1 and 3.2.

$$Prob_A = \frac{(k+A_i)^n}{(k+A_i)^n + (k+B_i)^n}, \qquad (Prob_A + Prob_B = 1) \tag{3.1}$$

$$A_{i+1} = A_i + \delta, \qquad B_{i+1} = B_i + (1-\delta) \qquad (A_i + B_i = i), \tag{3.2}$$

where δ describes a binary object and carries only 0 or 1 as value, which is computed over the runtime probability values (described as $Prob_A$ & $Prob_A$). The variable stacks A assigns the primary shortest path and B denotes the optimized shortest path over A.

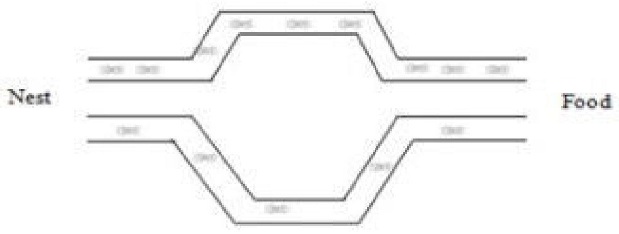


FIG. 3.1. *Shortest path in distributed sub-paths*

3.2. Decision Parameters. The VM load and failure rate has been assigned as the main parameters to take the scheduling decisions. Both of the parameters has been used for the purpose of data scheduling over the given cloud resources. The virtual machine load is the parameter which defines the overall utilization of the resources of the given virtual machine. The VM load can be used to signify the runtime availability in order to process the given task t on the given time t. The tasks running over the given VM, utilizes the certain amount of resources. The total percentage of the resources being used during the time t is considered as the VM load.

When the virtual machines are ordered in the workload allocation pool for process sequencing in the given cloud environment, the load monitoring on each virtual machine becomes very important step to correctly perform the data scheduling tasks. The virtual machine load or overhead is calculated on the basis of different parameters like CPU size, memory size etc. Each VM load must be calculated on the basis of its local parameters. Any use of general parameter values can result the biased load over the given VMs. The CPU and memory overhead or usage on the given VM considerably influences the performance of VMs in the process sequencing practices. The workload on VM can be evaluated on the basis of formula represented in Eq. 3.3. To calculate the total load over the virtual machine in the cloud environment is more than or equal to its capability, the Eq. 3.3 gives the result.

$$\sum\nolimits_{i}^{[v]} Load_i \times X_{ik} \lnsim Capacity, \qquad \forall k, P_k \in P \tag{3.3}$$

Finally, to justify the virtual machine load, Eq. 3.4 is used.

$$X_{ik} = x_{ik} \tag{3.4}$$

where X_{ik} is considered as the components of assignment to the non-overloaded VMs. The overloading or non-overloading defines the current state of the VM calculated after computing overall load and percentage of resources and measuring them against the threshold level.

The failure rate is described in the form of percentage of scheduling failures in processing the assigned tasks over the given VMs in the cloud environment. The failure rate signifies the trust of virtual machine. The VM with the lowest failure rate can be considered as the highly trusted VM and vice-versa. The probability of processing of the task can be increased by assigning the tasks over VMs with optimized & reduced failure rate (FR). The FR can be computed by using the Eq. 3.5.

$$FR = (\frac{T_p}{T_t}) \cdot 100 \tag{3.5}$$

where T_p is the sum of processed tasks and T_t is total amount of tasks assigned over the given VM.

3.3. Link Optimization Based Optimal VM Allocation (Link Optimization-OVA). In this work, the optimal load sharing approach based on the link optimization has been introduced for the load offset approach over the cloud environment in the case of data scheduling. The path A defines the first resource and path B defines the second resource. The other resources can be assigned with the further alphabets with the assumption that all of the resources or assets are logically able to executing all the processes in the cloud environment. The resource selection must be done on the basis of availability of RAM and CPU processing powers, which must make the whole process efficient in terms of response time. The traditional methods are known to allot the random resources for the given task, which effect the performance of cloud scheduling model and hence slow down the query processing procedure resulting with higher response time. The link optimization is the probability-based procedure to choose the appropriate resource in the available list of VMs. The proposed model is aimed at lower task response time for maximizing the number of jobs processing in the span of one second. The proposed model has been made capable of subdividing the task, which facilitates the quicker process and processes the smaller tasks faster than the hefty ones to reduce the overall load and to increase the number of successful requests processing every second. The subdivision of tasks is based on the length of the task. A task is usually divided into $'t'$ slots, where t is smallest time unit available for the task length calculation in our proposed model. A task smaller than or equal to t will be processed in one round, where the tasks larger than t can be scheduled in queue or on different VMs according to the load and time calculation for the faster processing. The arbitrary proportional rule is applied to recognize the ratio of processes in processing the given resource, and has been presented in the Eqs. 3.6 and 3.7.

$$P_1 = \frac{(R_1 + K)^k}{(R_1 + K)^k + (R_2 + K)^h}, \tag{3.6}$$

$$B_1 = P_1 \cdot TR_i, \tag{3.7}$$

where A_1 is the count of assigned tasks on the resource P_1 & A, involves the resource probability, R_1 denotes the usability index based on the available ratio of RAM and CPU on VM under consideration, TR_i depicts the resource availability required to process task i. The k and h are the coefficients used for the choice of probability among the available resources for sequencing of the processes among accessible resources. The value of k and h is calculated on the basis of VM load and resource availability on all of the available VMs. The variation in the values of k and h will define the variability on the basis of current processing load on different VMs, which inspires the task assignment decision of the link optimization algorithm. The used rule for the probability calculation has been represented in the Eq. 3.8.

$$P_j = \frac{(R_i + K)^k}{\sum_{i=1}^{n}((R_i + K)^k)}, \tag{3.8}$$

In the proposed work, the meta tasks are used for testing of the proposed model. The meta tasks does not carry any dependency on other tasks in the processing queue, which means the response time will be calculated for each individual task by evaluating the variation between finish time and start time. The waiting time is also considered as the response time delay, which is caused due to the waiting period spent in the queue.

Figure 3.2 represents the basic flow of Algorithm 13.

4. Results and Discussion. In this research, there are total seven VM allocation and selection policies are described. All seven models are programmed to utilize the different aspects into consideration in order to take the final decision on VM allocation and VM selection for the completion of job assignments. The VM allocation models used in this simulation are Local Regression (LR), Inter Quartile Range (IQR), Local Regression Robust (LRR), Non-Power Aware (NPA), Median Absolute Deviation (MAD), Dynamic Voltage and Frequency Scheduling (DVFS) and Static Threshold (THR) models. The following figures elaborates all of the models implemented under this research paper.

Each of the VM allocation model is further amalgamated with the VM selection models. The NPA and DVFS models are not primarily designed for specific VM selection or allocation policy. The NPA and DVFS models are designed to select all of the available VMs, and allocate sub-tasks or tasks on the optimal resource selected from the list.

Each of the VM allocation model (IQR, LR, LRR, MAD & THR) is combined with all VM selection models including Minimum Migration Time (MMT), Maximum Correlation (MC), Random Selection (RS) and Maximum Utilization (MU). All of the VM selection policies are described in the Fig. 4.1. There are total 22 combinations, which are produced using the combination of VM allocation and selection policies. The Fig. 4.1 shows all of the possible combinations of VM allocation and selection models.

Algorithm 13 Link Optimization - OVA Algorithm

```
1: Acquire the environmental parameters for task scheduling
2: Analyze & acquire the list of available VMs in the VM stack over cloud segment
3: Analyze & acquire the runtime performance of available VMs in the form of CPU, RAM, storage capacity,
   power consumption, etc.
4: Represent the acquired parameter list obtained on Step 2 & 3;
       $VM_l = V_1, V_2, V_3, V_4, ...V_n,$   (3.9)
   where VM is the virtual machine list and $V_1$ to $V_n$ represents the virtual machine IDs
5: Obtain cumulative & independent list of resources in the form of computing capacity
       $VM_r = VM_1, VM_2, VM_3, ...VM_n,$   (3.10)
   where $VM_r$ represents the resource capacity of each resource $VM_1$ to $VM_n$ to represent the virtual machine
   IDs
6: Begin the iterative structure to process tasks with every effective resource
     a. Obtain & acquire the resource availability from every VM on availability stack
            $VM_E = \int_{i=1}^{N} VM_i,$   (3.11)
        where $VM_E$ gives the resource availability after calculating the resource load using Eq. 3.12.
            $L = \frac{VCPU_u}{VCPU_T},$   (3.12)
        where $L$ represents the overall resource load on the particular VM, whereas the $VCPU_u$ and $VCPU_T$
        gives the currently used resources and total resources available respectively.
            $L_i = L_1, L_2, L_3, ...L_n,$   (3.13)
        where $L_i$ represents the list of resource load for all the VMs in simulation.
     b. The fundamental utilization factor is computed for individual resource
7: Terminate the iterative structure initiated on step 5
8: Assign the task stack to runtime cloud environment
       $T = t_1, t_2, t_3, ...t_n,$   (3.14)
   where $T$ vector represents the task vector and $t_1$ to $t_n$ represents the individual tasks
9: Determine the workflow's task stack and compute the length of each independent task in the stack
       $t_c(t_i) = (EST_{finishtime} - EST_{starttime}),$   (3.15)
   where $t_c$ and $t_i$ gives the overall time length for each of the task by subtracting the estimated start time
   from estimated finish time
10: In case a task is dependent or multivariate, sub-divide it into sub-stacks involving minor tasks recognizable
    with index $i$
11: Initialize the iterative structure to process each sub-task on sub-task stack indexed with index $i$
     a. Obtain the resource availability factors for each VM on the VM list
     b. Compute and validate the task duration (estimated) against the computational capacity (resource
        availability) against each available VM
     c. Determine the current load of each VM on the list by analyzing the resource engagement
                                    $A_j = P_j \cdot TR_i,$                                  (3.16)
        where $A_j$ depicts the availability of the VMs
     d. Observe and accumulate failure events of each VM on the list and prepare the $FR$ value to evaluate
        its endurance
     e. Confiscate all the VMs on the list with $FR$ below threshold to process current task of sub-task ($t$) to
        prepare the allocated VM resource list ($aVMrl$)
     f. Finally select the appropriate resource based upon best combination of time (estimated) and resource
        engagement from $aVMrl$
                                    $IfT_c(i) \ln VC(j),$                                     (3.17)
                                    $VM_E(K) = V(V_{c(i)}),$                                  (3.18)
        where $VM_E(K)$ resource availability after calculating the resource load for particular machine with id
        $K$, where $K$ any can be any value from the given VM IDs. VM represents the virtual machine list and
        $V_{c(i)}$ gives the capacity of the VM with ID as $i$.
     g. Revise resource allocation record accordingly and also update total load of allocated VM after task
        assignment
     h. Further, revise the utilization record enlisting resource availability
                                    $R_i = R_j + 1,$                                          (3.19)
        where $R_i$ is the usability and this equation shows the incremental usability index with the movement
        of each VM.
     i. Go the step 9(a) if not end of task list
12: Terminate the iterative structure and exit the program
```

The simulation results of all the unique combinations are acquired in the form of various performance parameters. These performance parameters are included to analyze the performance on the basis of time, VM migrations, Service Level Agreements (SLA) related parameters, Energy consumption, Host Shutdowns etc. Detailed statistical analysis of host shutdowns, VM & host migrations, VM & host selections and overall time-based analysis in the terms of mean and standard deviation is also computed. The Table 4.1 shows the detailed list of performance parameters.

The simulation of all results, based on the parameters discussed in Table 4.1, are obtained and listed in this section for each of the VM allocation and VM selection models. The only exceptions are Dynamic Voltage Frequency Scaling (DVFS) and Non-Power Aware (NPA) models. For these two exceptions, total 15 parameters are recorded in contrast to the 23 parameters for all other models.

The DVFS model has been described with the random nature, where all of the available VM are used in the random order without any qualitative based allocation parameters. The Fig. 4.2 shows the results obtained for the random DVFS.

In this sub-section, the VM allocation model of Inter Quartile Range (IQR) has been used along with the Maximum correlation (MC) method. Fig. 4.3 the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Inter Quartile Range (IQR) has been used along with the Minimum Migration Time (MMT) method. Fig. 4.4 represents the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Inter Quartile Range (IQR) has been used along with the Maximum Utilization (MU) method. Fig. 4.5 shows the results obtained from this model for all of the enlisted

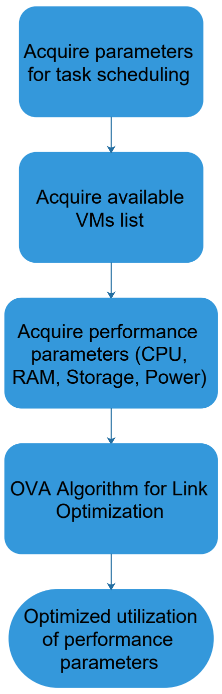

Acquire parameters for task scheduling
Acquire available VMs list
Acquire performance parameters (CPU, RAM, Storage, Power)
OVA Algorithm for Link Optimization
Optimized utilization of performance parameters

FIG. 3.2. *Basic flow of Proposed Algorithm*

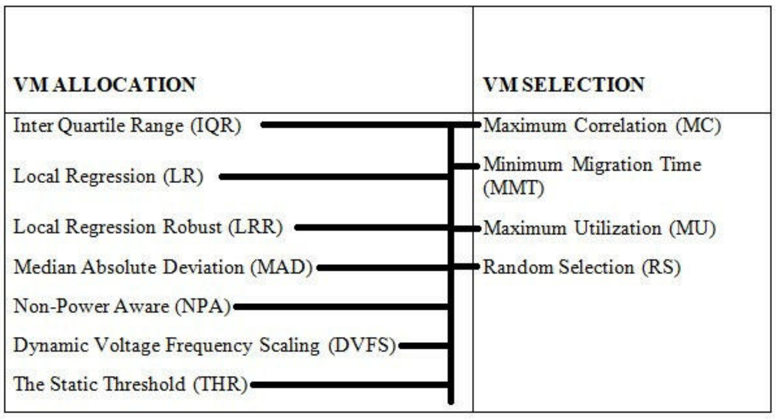

VM ALLOCATION
VM SELECTION
Inter Quartile Range (IQR)
Local Regression (LR)
Local Regression Robust (LRR)
Median Absolute Deviation (MAD)
Non-Power Aware (NPA)
Dynamic Voltage Frequency Scaling (DVFS)
The Static Threshold (THR)
Maximum Correlation (MC)
Minimum Migration Time (MMT)
Maximum Utilization (MU)
Random Selection (RS)

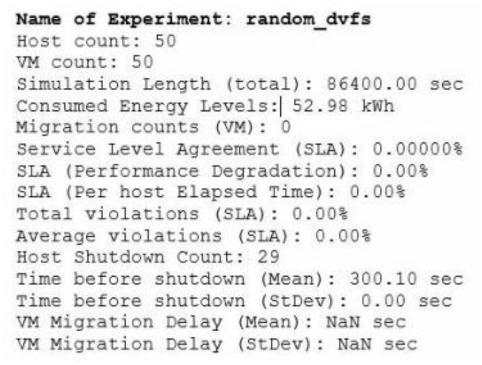
FIG. 4.1. *Possible combinations of VM allocation and VM selection models*

parameters.

In this sub-section, the VM allocation model of Inter Quartile Range (IQR) has been used along with the Random Selection (RS) method. The results shown in Fig. 4.6 is obtained from this model for all of the enlisted parameters.

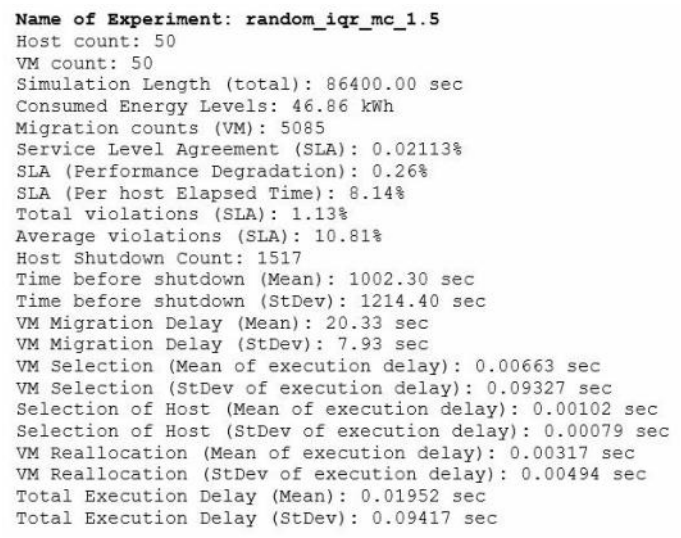
In this sub-section, the VM allocation model of Local Regression (LR) has been used along with the Maximum Correlation (MC) method. Fig. 4.7 represents the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Local Regression (LR) has been used along with the Minimum Migration Time (MMT) method. The results represented in Fig. 4.8 are obtained from this model for all of the enlisted parameters.



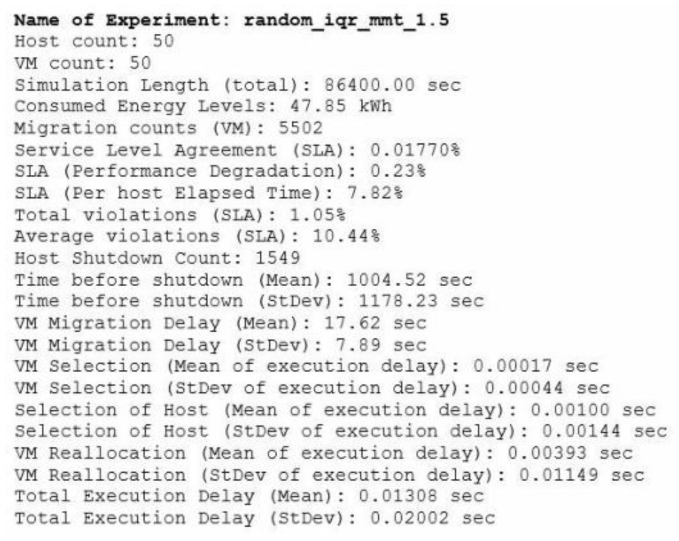
```
Name of Experiment: random_dvfs
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels:| 52.98 kWh
Migration counts (VM): 0
Service Level Agreement (SLA): 0.00000%
SLA (Performance Degradation): 0.00%
SLA (Per host Elapsed Time): 0.00%
Total violations (SLA): 0.00%
Average violations (SLA): 0.00%
Host Shutdown Count: 29
Time before shutdown (Mean): 300.10 sec
Time before shutdown (StDev): 0.00 sec
VM Migration Delay (Mean): NaN sec
VM Migration Delay (StDev): NaN sec
```

FIG. 4.2. *Results obtained for random DVFS*



```
Name of Experiment: random_iqr_mc_1.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 46.86 kWh
Migration counts (VM): 5085
Service Level Agreement (SLA): 0.02113%
SLA (Performance Degradation): 0.26%
SLA (Per host Elapsed Time): 8.14%
Total violations (SLA): 1.13%
Average violations (SLA): 10.81%
Host Shutdown Count: 1517
Time before shutdown (Mean): 1002.30 sec
Time before shutdown (StDev): 1214.40 sec
VM Migration Delay (Mean): 20.33 sec
VM Migration Delay (StDev): 7.93 sec
VM Selection (Mean of execution delay): 0.00663 sec
VM Selection (StDev of execution delay): 0.09327 sec
Selection of Host (Mean of execution delay): 0.00102 sec
Selection of Host (StDev of execution delay): 0.00079 sec
VM Reallocation (Mean of execution delay): 0.00317 sec
VM Reallocation (StDev of execution delay): 0.00494 sec
Total Execution Delay (Mean): 0.01952 sec
Total Execution Delay (StDev): 0.09417 sec
```

FIG. 4.3. *Results obtained for Inter Quartile Range (IQR)*



```
Name of Experiment: random_iqr_mmt_1.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 47.85 kWh
Migration counts (VM): 5502
Service Level Agreement (SLA): 0.01770%
SLA (Performance Degradation): 0.23%
SLA (Per host Elapsed Time): 7.82%
Total violations (SLA): 1.05%
Average violations (SLA): 10.44%
Host Shutdown Count: 1549
Time before shutdown (Mean): 1004.52 sec
Time before shutdown (StDev): 1178.23 sec
VM Migration Delay (Mean): 17.62 sec
VM Migration Delay (StDev): 7.89 sec
VM Selection (Mean of execution delay): 0.00017 sec
VM Selection (StDev of execution delay): 0.00044 sec
Selection of Host (Mean of execution delay): 0.00100 sec
Selection of Host (StDev of execution delay): 0.00144 sec
VM Reallocation (Mean of execution delay): 0.00393 sec
VM Reallocation (StDev of execution delay): 0.01149 sec
Total Execution Delay (Mean): 0.01308 sec
Total Execution Delay (StDev): 0.02002 sec
```

FIG. 4.4. *Results obtained for Inter Quartile Range (IQR) with Minimum Migration Time (MMT) method*

TABLE 4.1
List of performance parameters for the results evaluation

Parameter Name	Units
Host Count	Count (default 50)
VM Count	Count (default 50)
Simulation Length (Total)	Seconds (default 86400 seconds)
Consumed Energy Levels	kWh (kilo Watt per hour)
Migration counts (VM)	Count
Service Level Agreement (SLA)	Percentage
SLA (Performance Degradation)	Percentage
SLA (Per host Elapsed Time)	Percentage
Total violations (SLA)	Percentage
Average violations (SLA)	Percentage
Host Shutdown Count	Counts
Time before shutdown (Mean)	Seconds
Time before shutdown (StDev)	Seconds
VM Migration Delay (Mean)	Seconds
VM Migration Delay (StDev)	Seconds
VM Selection (Mean of execution delay)	Seconds
VM Selection (StDev of execution delay)	Seconds
Selection of Host (Mean of execution delay)	Seconds
Selection of Host (StDev of execution delay)	Seconds
VM Reallocation (Mean of execution delay)	Seconds
VM Reallocation (StDev of execution delay)	Seconds
Total Execution Delay (Mean)	Seconds
Total Execution Delay (StDev)	Seconds

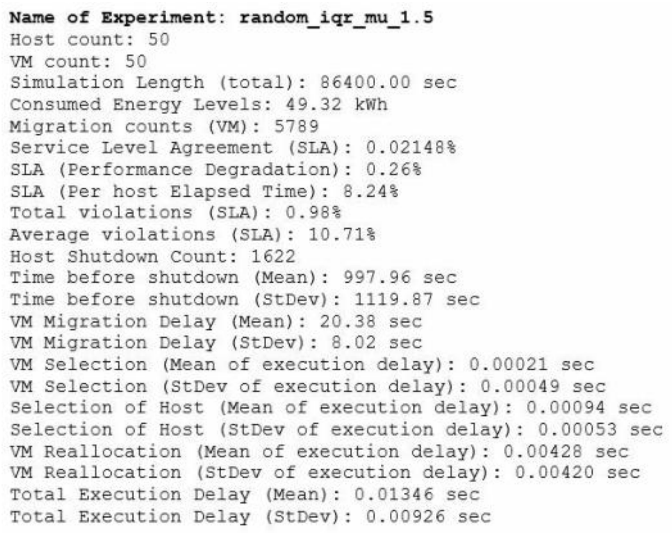
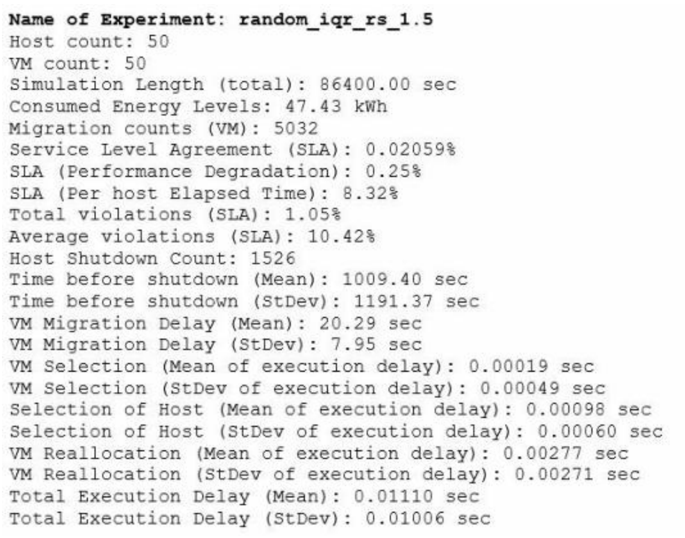

```
Name of Experiment: random_iqr_mu_1.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 49.32 kWh
Migration counts (VM): 5789
Service Level Agreement (SLA): 0.02148%
SLA (Performance Degradation): 0.26%
SLA (Per host Elapsed Time): 8.24%
Total violations (SLA): 0.98%
Average violations (SLA): 10.71%
Host Shutdown Count: 1622
Time before shutdown (Mean): 997.96 sec
Time before shutdown (StDev): 1119.87 sec
VM Migration Delay (Mean): 20.38 sec
VM Migration Delay (StDev): 8.02 sec
VM Selection (Mean of execution delay): 0.00021 sec
VM Selection (StDev of execution delay): 0.00049 sec
Selection of Host (Mean of execution delay): 0.00094 sec
Selection of Host (StDev of execution delay): 0.00053 sec
VM Reallocation (Mean of execution delay): 0.00428 sec
VM Reallocation (StDev of execution delay): 0.00420 sec
Total Execution Delay (Mean): 0.01346 sec
Total Execution Delay (StDev): 0.00926 sec
```

FIG. 4.5. *Results obtained for Inter Quartile Range (IQR) with Maximum Utilization (MU) method*

In this sub-section, the VM allocation model of Local Regression (LR) has been used along with the Maximum Utilization (MU) method. Fig. 4.9 shows the results obtained from this model for all of the enlisted parameters.

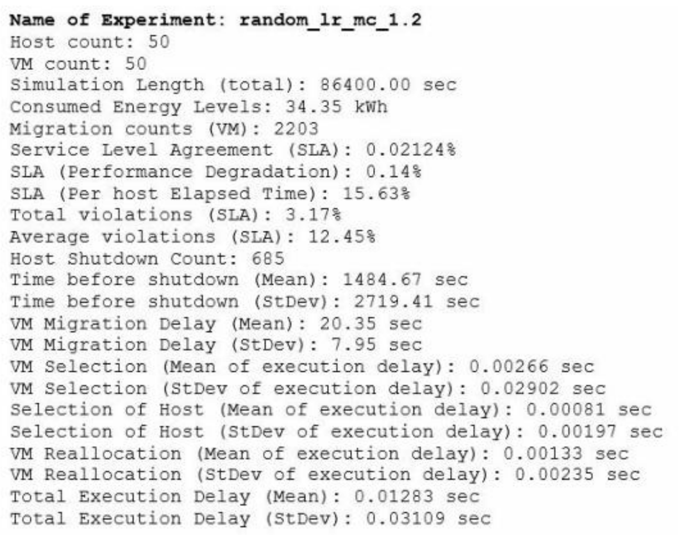
In this sub-section, the VM allocation model of Local Regression (LR) has been used along with the Random Selection (RS) method. Fig. 4.10 shows the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Local Regression Robust (LRR) has been used along with the Maximum Correlation (MC) method. The results represented in Fig. 4.11 is obtained from this model for all of the enlisted parameters.



```
Name of Experiment: random_iqr_rs_1.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 47.43 kWh
Migration counts (VM): 5032
Service Level Agreement (SLA): 0.02059%
SLA (Performance Degradation): 0.25%
SLA (Per host Elapsed Time): 8.32%
Total violations (SLA): 1.05%
Average violations (SLA): 10.42%
Host Shutdown Count: 1526
Time before shutdown (Mean): 1009.40 sec
Time before shutdown (StDev): 1191.37 sec
VM Migration Delay (Mean): 20.29 sec
VM Migration Delay (StDev): 7.95 sec
VM Selection (Mean of execution delay): 0.00019 sec
VM Selection (StDev of execution delay): 0.00049 sec
Selection of Host (Mean of execution delay): 0.00098 sec
Selection of Host (StDev of execution delay): 0.00060 sec
VM Reallocation (Mean of execution delay): 0.00277 sec
VM Reallocation (StDev of execution delay): 0.00271 sec
Total Execution Delay (Mean): 0.01110 sec
Total Execution Delay (StDev): 0.01006 sec
```

FIG. 4.6. *Results obtained for Inter Quartile Range (IQR) with Random Selection (RS) method*



```
Name of Experiment: random_lr_mc_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 34.35 kWh
Migration counts (VM): 2203
Service Level Agreement (SLA): 0.02124%
SLA (Performance Degradation): 0.14%
SLA (Per host Elapsed Time): 15.63%
Total violations (SLA): 3.17%
Average violations (SLA): 12.45%
Host Shutdown Count: 685
Time before shutdown (Mean): 1484.67 sec
Time before shutdown (StDev): 2719.41 sec
VM Migration Delay (Mean): 20.35 sec
VM Migration Delay (StDev): 7.95 sec
VM Selection (Mean of execution delay): 0.00266 sec
VM Selection (StDev of execution delay): 0.02902 sec
Selection of Host (Mean of execution delay): 0.00081 sec
Selection of Host (StDev of execution delay): 0.00197 sec
VM Reallocation (Mean of execution delay): 0.00133 sec
VM Reallocation (StDev of execution delay): 0.00235 sec
Total Execution Delay (Mean): 0.01283 sec
Total Execution Delay (StDev): 0.03109 sec
```

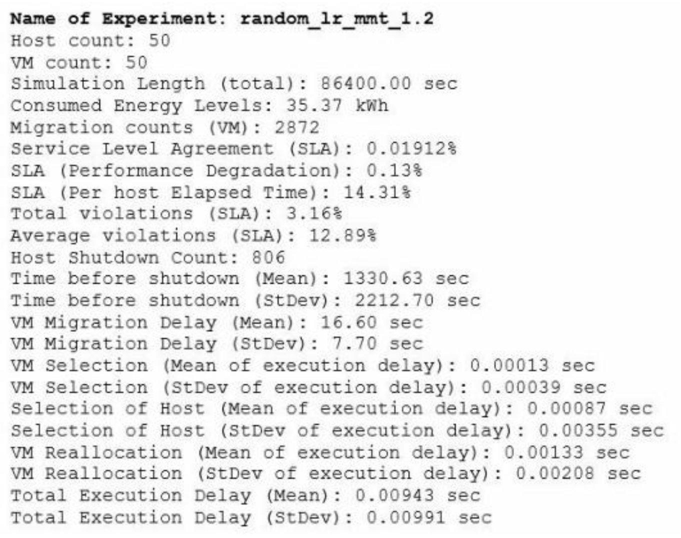
FIG. 4.7. *Results obtained for Local Regression (LR) with Maximum Correlation (MC) method*

In this sub-section, the VM allocation model of Local Regression Robust (LRR) has been used along with the Minimum Migration Time (MMT) method. Fig. 4.12 shows the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Local Regression Robust (LRR) has been used along with the Maximum Utilization (MU) method. The results obtained from this model for all of the enlisted parameters is shown in Fig. 4.13.

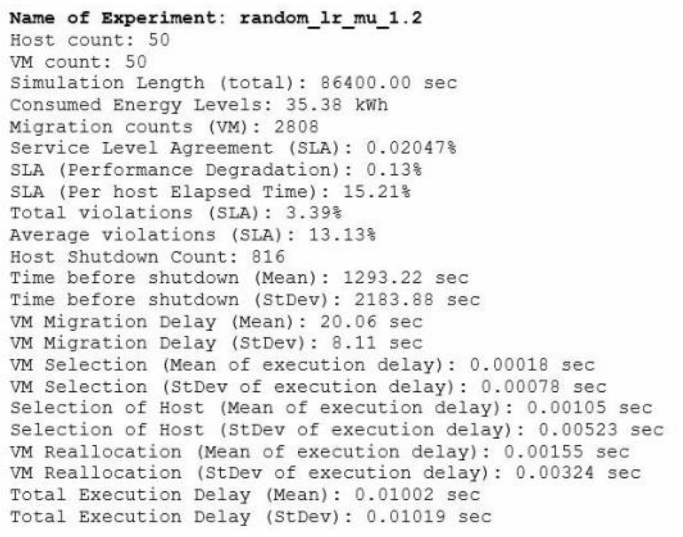
In this sub-section, the VM allocation model of Local Regression Robust (LRR) has been used along with the Random Selection (RS) method. The results obtained from this model for all of the enlisted parameters are represented in Fig. 4.14.

In this sub-section, the VM allocation model of Median Absolute Deviation (MAD) has been used along with the Maximum Correlation (MC) method. Fig. 4.15 represents the results obtained from this model for all of the enlisted parameters.



```
Name of Experiment: random_lr_mmt_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 35.37 kWh
Migration counts (VM): 2872
Service Level Agreement (SLA): 0.01912%
SLA (Performance Degradation): 0.13%
SLA (Per host Elapsed Time): 14.31%
Total violations (SLA): 3.16%
Average violations (SLA): 12.89%
Host Shutdown Count: 806
Time before shutdown (Mean): 1330.63 sec
Time before shutdown (StDev): 2212.70 sec
VM Migration Delay (Mean): 16.60 sec
VM Migration Delay (StDev): 7.70 sec
VM Selection (Mean of execution delay): 0.00013 sec
VM Selection (StDev of execution delay): 0.00039 sec
Selection of Host (Mean of execution delay): 0.00087 sec
Selection of Host (StDev of execution delay): 0.00355 sec
VM Reallocation (Mean of execution delay): 0.00133 sec
VM Reallocation (StDev of execution delay): 0.00208 sec
Total Execution Delay (Mean): 0.00943 sec
Total Execution Delay (StDev): 0.00991 sec
```

FIG. 4.8. *Results obtained for Local Regression (LR) with Minimum Migration Time (MMT) method*



```
Name of Experiment: random_lr_mu_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 35.38 kWh
Migration counts (VM): 2808
Service Level Agreement (SLA): 0.02047%
SLA (Performance Degradation): 0.13%
SLA (Per host Elapsed Time): 15.21%
Total violations (SLA): 3.39%
Average violations (SLA): 13.13%
Host Shutdown Count: 816
Time before shutdown (Mean): 1293.22 sec
Time before shutdown (StDev): 2183.88 sec
VM Migration Delay (Mean): 20.06 sec
VM Migration Delay (StDev): 8.11 sec
VM Selection (Mean of execution delay): 0.00018 sec
VM Selection (StDev of execution delay): 0.00078 sec
Selection of Host (Mean of execution delay): 0.00105 sec
Selection of Host (StDev of execution delay): 0.00523 sec
VM Reallocation (Mean of execution delay): 0.00155 sec
VM Reallocation (StDev of execution delay): 0.00324 sec
Total Execution Delay (Mean): 0.01002 sec
Total Execution Delay (StDev): 0.01019 sec
```

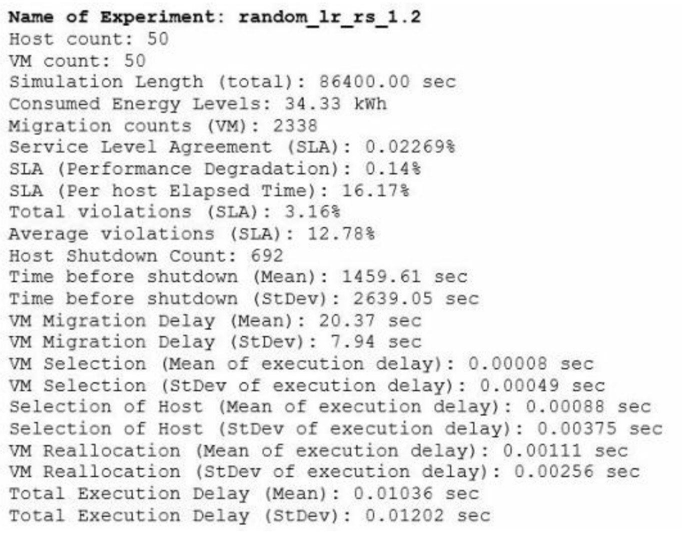
FIG. 4.9. *Results obtained for Local Regression (LR) with Maximum Utilization (MU) method*

In this sub-section, the VM allocation model of Median Absolute Deviation (MAD) has been used along with the Minimum Migration Time (MMT) method. The results, shown in Fig. 4.16, are obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Median Absolute Deviation (MAD) has been used along with the Maximum Utilization (MU) method. Fig. 4.17 represents the results obtained from this model for all of the enlisted parameters.

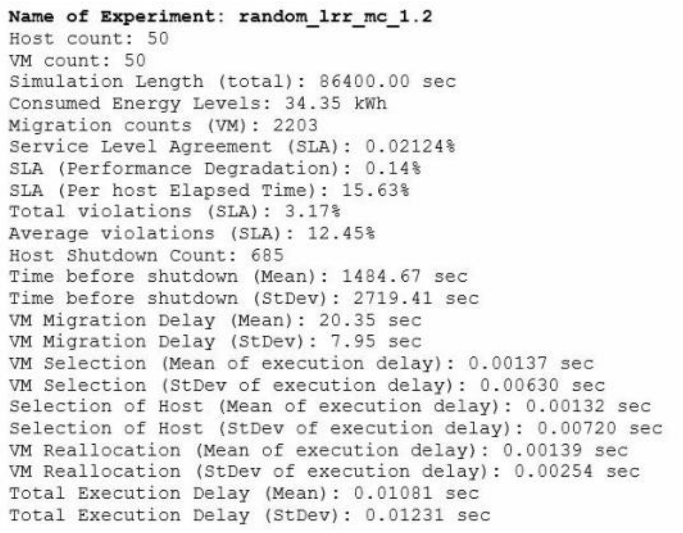
In this sub-section, the VM allocation model of Median Absolute Deviation (MAD) has been used along with the Random Selection (RS) method. The results obtained from this model for all of the enlisted parameters are shown in Fig. 4.18.

In this sub-section, the VM allocation model of Non-Power Aware has been used with no method for VM selection. The VM selection policy is simple random method like DVFS, which is unlike the random selection (RS) method for other VM allocation policies. Fig. 4.19 shows the results obtained from this model for all of



```
Name of Experiment: random_lr_rs_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 34.33 kWh
Migration counts (VM): 2338
Service Level Agreement (SLA): 0.02269%
SLA (Performance Degradation): 0.14%
SLA (Per host Elapsed Time): 16.17%
Total violations (SLA): 3.16%
Average violations (SLA): 12.78%
Host Shutdown Count: 692
Time before shutdown (Mean): 1459.61 sec
Time before shutdown (StDev): 2639.05 sec
VM Migration Delay (Mean): 20.37 sec
VM Migration Delay (StDev): 7.94 sec
VM Selection (Mean of execution delay): 0.00008 sec
VM Selection (StDev of execution delay): 0.00049 sec
Selection of Host (Mean of execution delay): 0.00088 sec
Selection of Host (StDev of execution delay): 0.00375 sec
VM Reallocation (Mean of execution delay): 0.00111 sec
VM Reallocation (StDev of execution delay): 0.00256 sec
Total Execution Delay (Mean): 0.01036 sec
Total Execution Delay (StDev): 0.01202 sec
```

FIG. 4.10. *Results obtained for Local Regression (LR) with Random Selection (RS) method*



```
Name of Experiment: random_lrr_mc_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 34.35 kWh
Migration counts (VM): 2203
Service Level Agreement (SLA): 0.02124%
SLA (Performance Degradation): 0.14%
SLA (Per host Elapsed Time): 15.63%
Total violations (SLA): 3.17%
Average violations (SLA): 12.45%
Host Shutdown Count: 685
Time before shutdown (Mean): 1484.67 sec
Time before shutdown (StDev): 2719.41 sec
VM Migration Delay (Mean): 20.35 sec
VM Migration Delay (StDev): 7.95 sec
VM Selection (Mean of execution delay): 0.00137 sec
VM Selection (StDev of execution delay): 0.00630 sec
Selection of Host (Mean of execution delay): 0.00132 sec
Selection of Host (StDev of execution delay): 0.00720 sec
VM Reallocation (Mean of execution delay): 0.00139 sec
VM Reallocation (StDev of execution delay): 0.00254 sec
Total Execution Delay (Mean): 0.01081 sec
Total Execution Delay (StDev): 0.01231 sec
```

FIG. 4.11. *Results obtained for Local Regression Robust (LRR) with Maximum Correlation (MC) method*

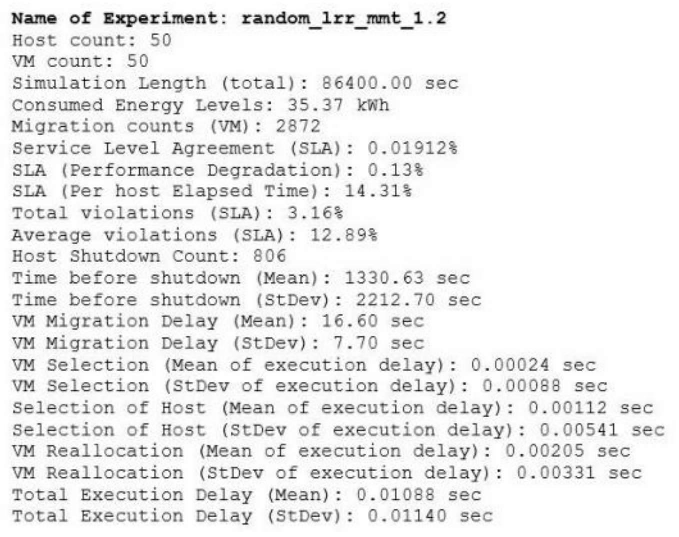
the enlisted parameters.

In this sub-section, the VM allocation model of Static Threshold (THR) has been used along with the Maximum Correlation (MC) method. Fig. 4.20 shows the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Static Threshold (THR) has been used along with the Minimum Migration Time (MMT) method. The results obtained from this model for all of the enlisted parameters are shown in Fig. 4.21.

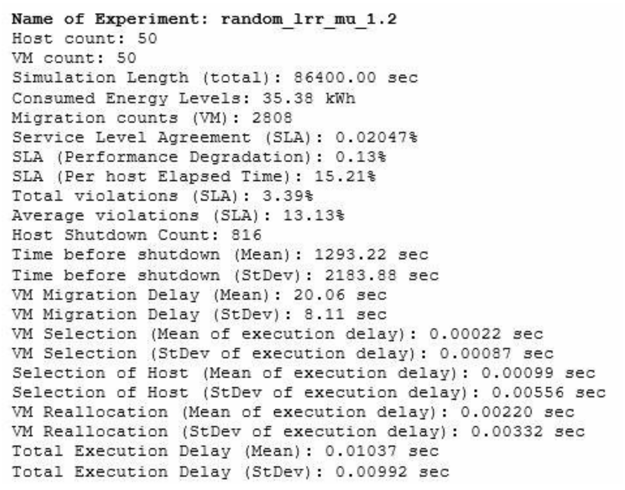
In this sub-section, the VM allocation model of Static Threshold (THR) has been used along with the Maximum Utilization (MU) method. Fig. 4.22 shows the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Static Threshold (THR) has been used along with the Random Selection (RS) method. Fig. 4.23 represents the results obtained from this model for all of the enlisted



```
Name of Experiment: random_lrr_mmt_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 35.37 kWh
Migration counts (VM): 2872
Service Level Agreement (SLA): 0.01912%
SLA (Performance Degradation): 0.13%
SLA (Per host Elapsed Time): 14.31%
Total violations (SLA): 3.16%
Average violations (SLA): 12.89%
Host Shutdown Count: 806
Time before shutdown (Mean): 1330.63 sec
Time before shutdown (StDev): 2212.70 sec
VM Migration Delay (Mean): 16.60 sec
VM Migration Delay (StDev): 7.70 sec
VM Selection (Mean of execution delay): 0.00024 sec
VM Selection (StDev of execution delay): 0.00088 sec
Selection of Host (Mean of execution delay): 0.00112 sec
Selection of Host (StDev of execution delay): 0.00541 sec
VM Reallocation (Mean of execution delay): 0.00205 sec
VM Reallocation (StDev of execution delay): 0.00331 sec
Total Execution Delay (Mean): 0.01088 sec
Total Execution Delay (StDev): 0.01140 sec
```

FIG. 4.12. *Results obtained for Local Regression Robust (LRR) with Minimum Migration Time (MMT) method*



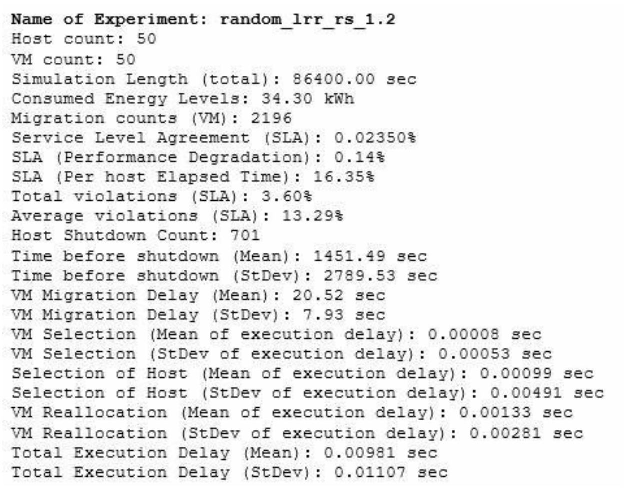
```
Name of Experiment: random_lrr_mu_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 35.38 kWh
Migration counts (VM): 2808
Service Level Agreement (SLA): 0.02047%
SLA (Performance Degradation): 0.13%
SLA (Per host Elapsed Time): 15.21%
Total violations (SLA): 3.39%
Average violations (SLA): 13.13%
Host Shutdown Count: 816
Time before shutdown (Mean): 1293.22 sec
Time before shutdown (StDev): 2183.88 sec
VM Migration Delay (Mean): 20.06 sec
VM Migration Delay (StDev): 8.11 sec
VM Selection (Mean of execution delay): 0.00022 sec
VM Selection (StDev of execution delay): 0.00087 sec
Selection of Host (Mean of execution delay): 0.00099 sec
Selection of Host (StDev of execution delay): 0.00556 sec
VM Reallocation (Mean of execution delay): 0.00220 sec
VM Reallocation (StDev of execution delay): 0.00332 sec
Total Execution Delay (Mean): 0.01037 sec
Total Execution Delay (StDev): 0.00992 sec
```

FIG. 4.13. *Results obtained for Local Regression Robust (LR) with Maximum Utilization (MU) method*

parameters.

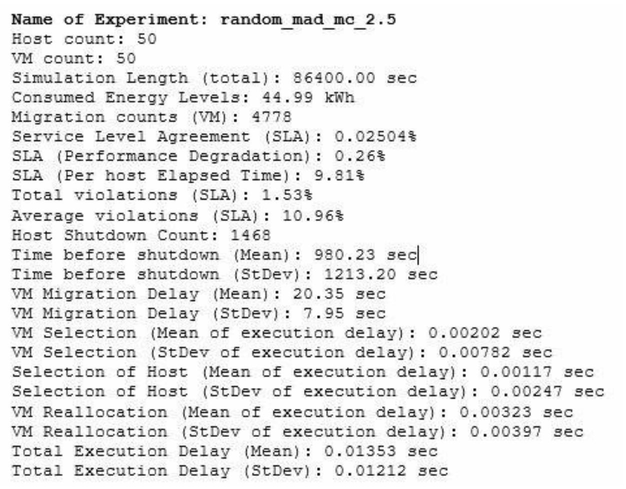
Table 4.2 shows the summary of the results for each experiment. This table represents the experiment name and the result obtained by that particular experiment with respect to each parameter. This summary will help us to evaluate and analyze the conducted experiments in much easier way.

All the experiments were conducted keeping the host count and VM count fixed (as 50) so as to compute the results on the same platform. This helps us in comparison with the different algorithms. From this, it is seen that experiment name: random_npa consumes the highest energy levels than all the experiments.



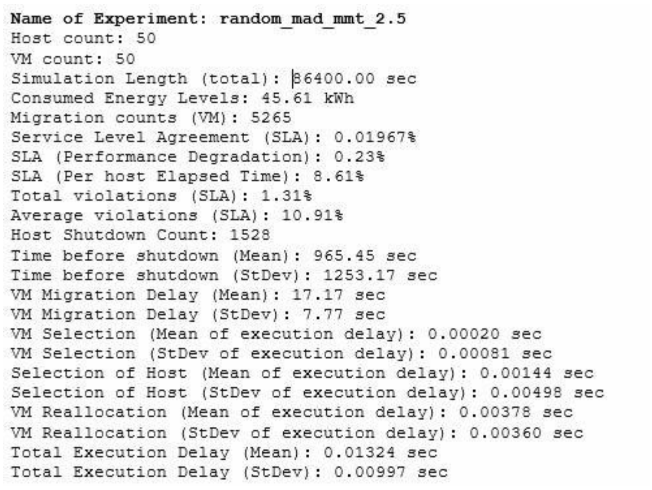
```
Name of Experiment: random_lrr_rs_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 34.30 kWh
Migration counts (VM): 2196
Service Level Agreement (SLA): 0.02350%
SLA (Performance Degradation): 0.14%
SLA (Per host Elapsed Time): 16.35%
Total violations (SLA): 3.60%
Average violations (SLA): 13.29%
Host Shutdown Count: 701
Time before shutdown (Mean): 1451.49 sec
Time before shutdown (StDev): 2789.53 sec
VM Migration Delay (Mean): 20.52 sec
VM Migration Delay (StDev): 7.93 sec
VM Selection (Mean of execution delay): 0.00008 sec
VM Selection (StDev of execution delay): 0.00053 sec
Selection of Host (Mean of execution delay): 0.00099 sec
Selection of Host (StDev of execution delay): 0.00491 sec
VM Reallocation (Mean of execution delay): 0.00133 sec
VM Reallocation (StDev of execution delay): 0.00281 sec
Total Execution Delay (Mean): 0.00981 sec
Total Execution Delay (StDev): 0.01107 sec
```

Fig. 4.14. *Results obtained for Local Regression Robust (LRR) with Random Selection (RS) method*



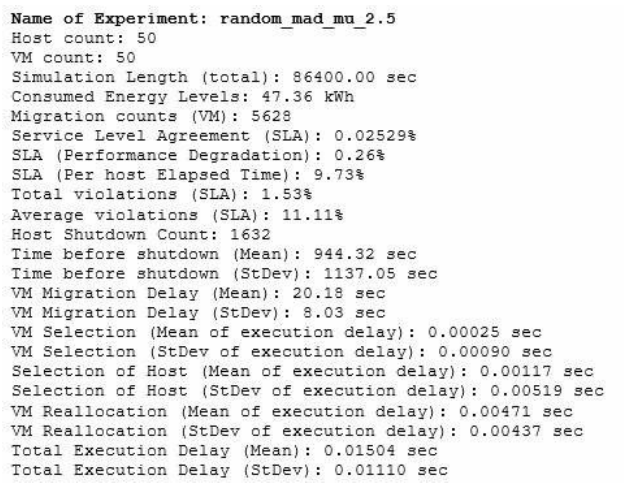
```
Name of Experiment: random_mad_mc_2.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 44.99 kWh
Migration counts (VM): 4778
Service Level Agreement (SLA): 0.02504%
SLA (Performance Degradation): 0.26%
SLA (Per host Elapsed Time): 9.81%
Total violations (SLA): 1.53%
Average violations (SLA): 10.96%
Host Shutdown Count: 1468
Time before shutdown (Mean): 980.23 sec
Time before shutdown (StDev): 1213.20 sec
VM Migration Delay (Mean): 20.35 sec
VM Migration Delay (StDev): 7.95 sec
VM Selection (Mean of execution delay): 0.00202 sec
VM Selection (StDev of execution delay): 0.00782 sec
Selection of Host (Mean of execution delay): 0.00117 sec
Selection of Host (StDev of execution delay): 0.00247 sec
VM Reallocation (Mean of execution delay): 0.00323 sec
VM Reallocation (StDev of execution delay): 0.00397 sec
Total Execution Delay (Mean): 0.01353 sec
Total Execution Delay (StDev): 0.01212 sec
```

Fig. 4.15. *Results obtained for Median Absolute Deviation (MAD) with Maximum Correlation (MC) method*



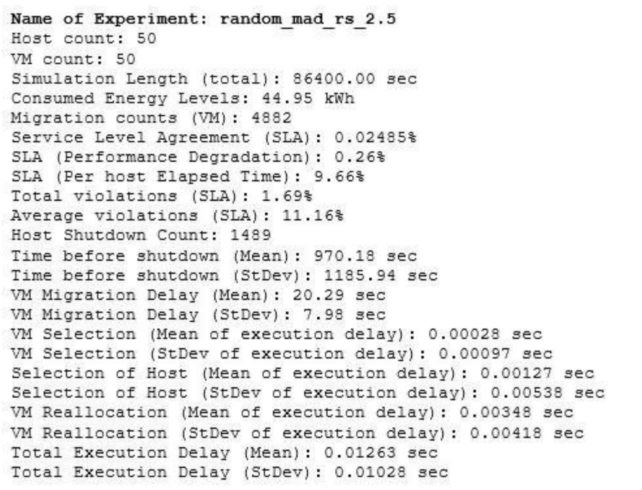
```
Name of Experiment: random_mad_mmt_2.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 45.61 kWh
Migration counts (VM): 5265
Service Level Agreement (SLA): 0.01967%
SLA (Performance Degradation): 0.23%
SLA (Per host Elapsed Time): 8.61%
Total violations (SLA): 1.31%
Average violations (SLA): 10.91%
Host Shutdown Count: 1528
Time before shutdown (Mean): 965.45 sec
Time before shutdown (StDev): 1253.17 sec
VM Migration Delay (Mean): 17.17 sec
VM Migration Delay (StDev): 7.77 sec
VM Selection (Mean of execution delay): 0.00020 sec
VM Selection (StDev of execution delay): 0.00081 sec
Selection of Host (Mean of execution delay): 0.00144 sec
Selection of Host (StDev of execution delay): 0.00498 sec
VM Reallocation (Mean of execution delay): 0.00378 sec
VM Reallocation (StDev of execution delay): 0.00360 sec
Total Execution Delay (Mean): 0.01324 sec
Total Execution Delay (StDev): 0.00997 sec
```

Fig. 4.16. *Results obtained for Median Absolute Deviation (MAD) with Minimum Migration Time (MMT) method*



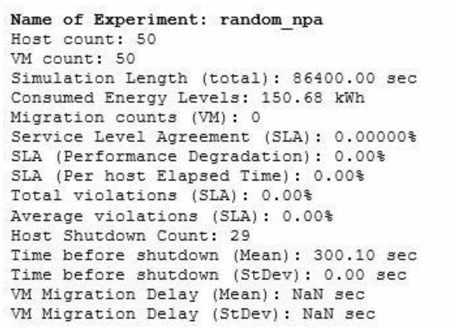
```
Name of Experiment: random_mad_mu_2.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 47.36 kWh
Migration counts (VM): 5628
Service Level Agreement (SLA): 0.02529%
SLA (Performance Degradation): 0.26%
SLA (Per host Elapsed Time): 9.73%
Total violations (SLA): 1.53%
Average violations (SLA): 11.11%
Host Shutdown Count: 1632
Time before shutdown (Mean): 944.32 sec
Time before shutdown (StDev): 1137.05 sec
VM Migration Delay (Mean): 20.18 sec
VM Migration Delay (StDev): 8.03 sec
VM Selection (Mean of execution delay): 0.00025 sec
VM Selection (StDev of execution delay): 0.00090 sec
Selection of Host (Mean of execution delay): 0.00117 sec
Selection of Host (StDev of execution delay): 0.00519 sec
VM Reallocation (Mean of execution delay): 0.00471 sec
VM Reallocation (StDev of execution delay): 0.00437 sec
Total Execution Delay (Mean): 0.01504 sec
Total Execution Delay (StDev): 0.01110 sec
```

Fig. 4.17. *Results obtained for Median Absolute Deviation (MAD) with Maximum Utilization (MU) method*



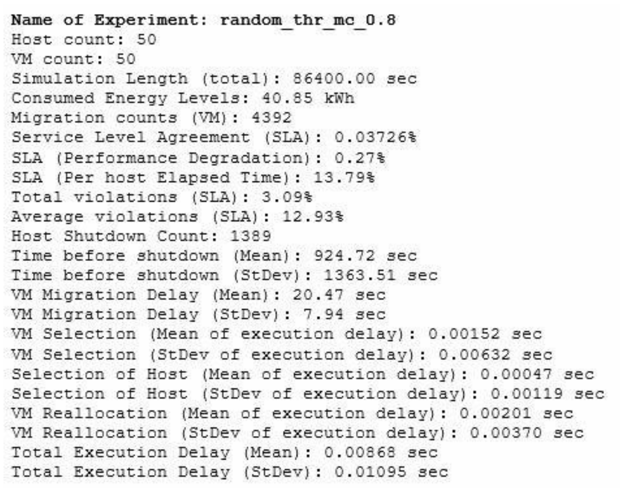
```
Name of Experiment: random_mad_rs_2.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 44.95 kWh
Migration counts (VM): 4882
Service Level Agreement (SLA): 0.02485%
SLA (Performance Degradation): 0.26%
SLA (Per host Elapsed Time): 9.66%
Total violations (SLA): 1.69%
Average violations (SLA): 11.16%
Host Shutdown Count: 1489
Time before shutdown (Mean): 970.18 sec
Time before shutdown (StDev): 1185.94 sec
VM Migration Delay (Mean): 20.29 sec
VM Migration Delay (StDev): 7.98 sec
VM Selection (Mean of execution delay): 0.00028 sec
VM Selection (StDev of execution delay): 0.00097 sec
Selection of Host (Mean of execution delay): 0.00127 sec
Selection of Host (StDev of execution delay): 0.00538 sec
VM Reallocation (Mean of execution delay): 0.00348 sec
VM Reallocation (StDev of execution delay): 0.00418 sec
Total Execution Delay (Mean): 0.01263 sec
Total Execution Delay (StDev): 0.01028 sec
```

Fig. 4.18. *Results obtained for Median Absolute Deviation (MAD) with Random Selection (RS) method*



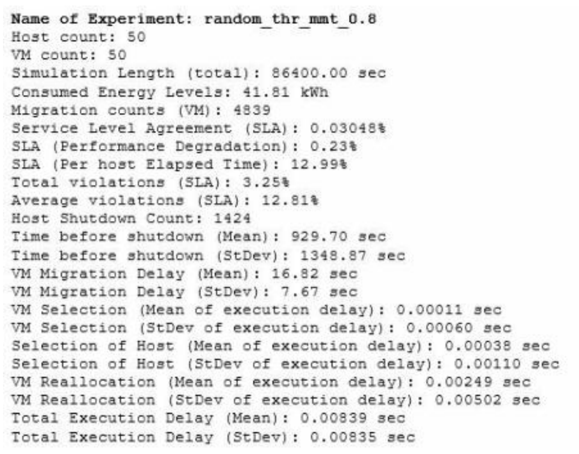
```
Name of Experiment: random_npa
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 150.68 kWh
Migration counts (VM): 0
Service Level Agreement (SLA): 0.00000%
SLA (Performance Degradation): 0.00%
SLA (Per host Elapsed Time): 0.00%
Total violations (SLA): 0.00%
Average violations (SLA): 0.00%
Host Shutdown Count: 29
Time before shutdown (Mean): 300.10 sec
Time before shutdown (StDev): 0.00 sec
VM Migration Delay (Mean): NaN sec
VM Migration Delay (StDev): NaN sec
```

Fig. 4.19. *Results obtained for Non-Power Aware (NPA)*



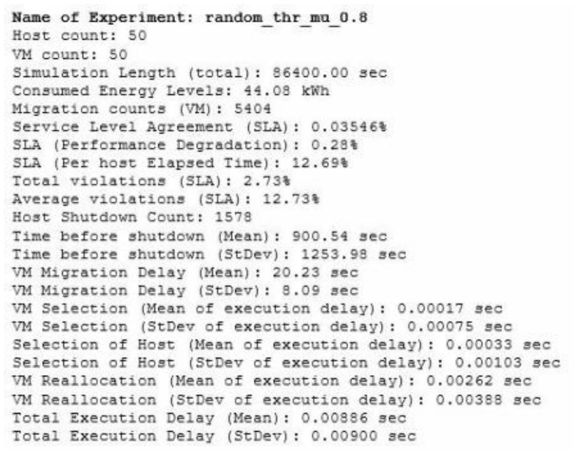
```
Name of Experiment: random_thr_mc_0.8
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 40.85 kWh
Migration counts (VM): 4392
Service Level Agreement (SLA): 0.03726%
SLA (Performance Degradation): 0.27%
SLA (Per host Elapsed Time): 13.79%
Total violations (SLA): 3.09%
Average violations (SLA): 12.93%
Host Shutdown Count: 1389
Time before shutdown (Mean): 924.72 sec
Time before shutdown (StDev): 1363.51 sec
VM Migration Delay (Mean): 20.47 sec
VM Migration Delay (StDev): 7.94 sec
VM Selection (Mean of execution delay): 0.00152 sec
VM Selection (StDev of execution delay): 0.00632 sec
Selection of Host (Mean of execution delay): 0.00047 sec
Selection of Host (StDev of execution delay): 0.00119 sec
VM Reallocation (Mean of execution delay): 0.00201 sec
VM Reallocation (StDev of execution delay): 0.00370 sec
Total Execution Delay (Mean): 0.00868 sec
Total Execution Delay (StDev): 0.01095 sec
```

Fig. 4.20. *Results obtained for Static Threshold (THR) with Maximum Correlation (MC) method*



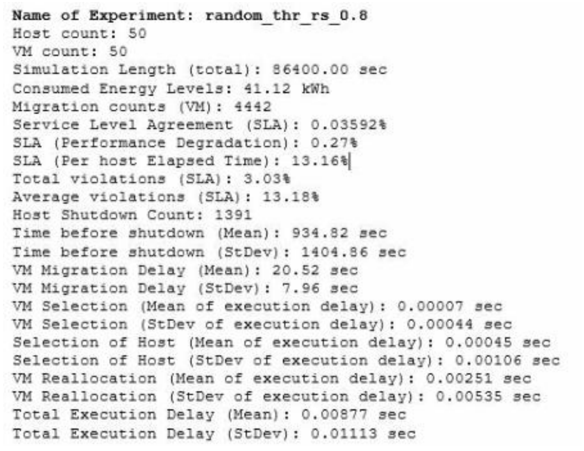
```
Name of Experiment: random_thr_mmt_0.8
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 41.81 kWh
Migration counts (VM): 4839
Service Level Agreement (SLA): 0.03048%
SLA (Performance Degradation): 0.23%
SLA (Per host Elapsed Time): 12.99%
Total violations (SLA): 3.25%
Average violations (SLA): 12.81%
Host Shutdown Count: 1424
Time before shutdown (Mean): 929.70 sec
Time before shutdown (StDev): 1348.87 sec
VM Migration Delay (Mean): 16.82 sec
VM Migration Delay (StDev): 7.67 sec
VM Selection (Mean of execution delay): 0.00011 sec
VM Selection (StDev of execution delay): 0.00060 sec
Selection of Host (Mean of execution delay): 0.00038 sec
Selection of Host (StDev of execution delay): 0.00110 sec
VM Reallocation (Mean of execution delay): 0.00249 sec
VM Reallocation (StDev of execution delay): 0.00502 sec
Total Execution Delay (Mean): 0.00839 sec
Total Execution Delay (StDev): 0.00835 sec
```

Fig. 4.21. *Results obtained for Static Threshold (THR) with Minimum Migration Time (MMT) method*



```
Name of Experiment: random_thr_mu_0.8
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 44.08 kWh
Migration counts (VM): 5404
Service Level Agreement (SLA): 0.03546%
SLA (Performance Degradation): 0.28%
SLA (Per host Elapsed Time): 12.69%
Total violations (SLA): 2.73%
Average violations (SLA): 12.73%
Host Shutdown Count: 1578
Time before shutdown (Mean): 900.54 sec
Time before shutdown (StDev): 1253.98 sec
VM Migration Delay (Mean): 20.23 sec
VM Migration Delay (StDev): 8.09 sec
VM Selection (Mean of execution delay): 0.00017 sec
VM Selection (StDev of execution delay): 0.00075 sec
Selection of Host (Mean of execution delay): 0.00033 sec
Selection of Host (StDev of execution delay): 0.00103 sec
VM Reallocation (Mean of execution delay): 0.00262 sec
VM Reallocation (StDev of execution delay): 0.00388 sec
Total Execution Delay (Mean): 0.00886 sec
Total Execution Delay (StDev): 0.00900 sec
```

Fig. 4.22. *Results obtained for Static Threshold (THR) with Maximum Utilization (MU) method*



```
Name of Experiment: random_thr_rs_0.8
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 41.12 kWh
Migration counts (VM): 4442
Service Level Agreement (SLA): 0.03592%
SLA (Performance Degradation): 0.27%
SLA (Per host Elapsed Time): 13.16%
Total violations (SLA): 3.03%
Average violations (SLA): 13.18%
Host Shutdown Count: 1391
Time before shutdown (Mean): 934.82 sec
Time before shutdown (StDev): 1404.86 sec
VM Migration Delay (Mean): 20.52 sec
VM Migration Delay (StDev): 7.96 sec
VM Selection (Mean of execution delay): 0.00007 sec
VM Selection (StDev of execution delay): 0.00044 sec
Selection of Host (Mean of execution delay): 0.00045 sec
Selection of Host (StDev of execution delay): 0.00106 sec
VM Reallocation (Mean of execution delay): 0.00251 sec
VM Reallocation (StDev of execution delay): 0.00535 sec
Total Execution Delay (Mean): 0.00877 sec
Total Execution Delay (StDev): 0.01113 sec
```

Fig. 4.23. *Results obtained for Static Threshold (THR) with Random Selection (RS) method*

Table 4.2
Result Summary for Each Experiment

Experiment Name/ Parameter	Host count	VM count	SimulationLength	Consumed Energy Levels	Migration counts	Service Level Agreement	Performance SLA	Per Host Elapsed Time SLA	Total violations	Average violations	Host shutdown count	Time before shutdown_Mean	Time before shutdown_StDev	Mean VM Migration Delay	StDev VM Migration Delay	Mean VM Selection	StDev VM Selection	Mean Host Selection	StDev Host Selection	VM Reallocation Mean	VM Reallocation StDev	Total Execution Delay Mean	StDev Total Execution Delay
random_dvfs	50	50	86400	52.98	0	0	0	0	0	0	29	300.1	0	NaN	NaN								
random_iqr_mc_1.5	50	50	86400	46.86	5085	0.02113	0.26	8.14	1.13	10.81	1517	1002.3	1214.4	20.33	7.93	0.00663	0.09327	0.00102	0.00079	0.00317	0.00494	0.01952	0.09417
random_iqr_mmt _1.5	50	50	86400	47.85	5502	0.0177	0.23	7.82	1.05	10.44	1549	1004.52	1178.23	17.62	7.89	0.00017	0.00044	0.001	0.00144	0.00393	0.01149	0.01308	0.02002
random_iqr_mu_1.5	50	50	86400	49.32	5789	0.02148	0.26	8.24	0.98	10.71	1622	997.96	1119.87	20.38	8.02	0.00021	0.00049	0.00094	0.00053	0.00428	0.0042	0.01346	0.00926
random_iqr_rs_1.5	50	50	86400	47.43	5032	0.02059	0.25	8.32	1.05	10.42	1526	1009.4	1191.37	20.29	7.95	0.00019	0.00049	0.00098	0.0006	0.00277	0.00271	0.0111	0.01006
random_lr_mc_1.2	50	50	86400	34.35	2203	0.02124	0.14	15.63	3.17	12.45	685	1484.67	2719.41	20.35	7.95	0.00266	0.02902	0.00081	0.00197	0.00133	0.00235	0.01283	0.03109
random_lr_mmt_1.2	50	50	86400	35.37	2872	0.01912	0.13	14.31	3.16	12.89	806	1330.63	2212.7	16.6	7.7	0.00013	0.00039	0.00087	0.00355	0.00133	0.00208	0.00943	0.00991
random_lr_mu_1.2	50	50	86400	35.38	2808	0.02047	0.13	15.21	3.39	13.13	816	1293.22	2183.88	20.06	8.11	0.00018	0.00078	0.00105	0.00523	0.00155	0.00324	0.01002	0.01019
random_lr_rs_1.2	50	50	86400	34.33	2338	0.02269	0.14	16.17	3.16	12.78	692	1459.61	2639.05	20.37	7.94	0.00008	0.00049	0.00088	0.00375	0.00111	0.00256	0.01036	0.01202
random_lrr_mc_1.2	50	50	86400	34.35	2203	0.02124	0.14	15.63	3.17	12.45	685	1484.67	2719.41	20.35	7.95	0.00137	0.0063	0.00132	0.0072	0.00139	0.00254	0.01081	0.01231
random_lrr_mmt_1.2	50	50	86400	35.37	2872	0.01912	0.13	14.31	3.16	12.89	806	1330.63	2212.7	16.6	7.7	0.00024	0.00088	0.00112	0.00541	0.00205	0.00331	0.01088	0.0114
random_lrr_mu_1.2	50	50	86400	35.38	2808	0.02047	0.13	15.21	3.39	13.13	816	1293.22	2183.88	20.06	8.11	0.00022	0.00087	0.00099	0.00556	0.0022	0.00332	0.01037	0.00992
random_lrr_rs_1.2	50	50	86400	34.3	2196	0.0235	0.14	16.35	3.6	13.29	701	1451.49	2789.53	20.52	7.93	0.00008	0.00053	0.00099	0.00491	0.00133	0.00281	0.00981	0.01107
random_mad_mc_2.5	50	50	86400	44.99	4778	0.02504	0.26	9.81	1.53	10.96	1468	980.23	1213.2	20.35	7.95	0.00202	0.00782	0.00117	0.00247	0.00323	0.00397	0.01353	0.01212
random_mad_mmt_2.5	50	50	86400	45.61	5265	0.01967	0.23	8.61	1.31	10.91	1528	965.45	1253.17	17.17	7.77	0.0002	0.00081	0.00144	0.00498	0.00378	0.0036	0.01324	0.00997
random_mad_mu_2.5	50	50	86400	47.36	5628	0.02529	0.26	9.73	1.53	11.11	1632	944.32	1137.05	20.18	8.03	0.00025	0.0009	0.00117	0.00519	0.00471	0.00437	0.01504	0.0111
random_mad_rs_2.5	50	50	86400	44.95	4882	0.02485	0.26	9.66	1.69	11.16	1489	970.18	1185.94	20.29	7.98	0.00028	0.00097	0.00127	0.00538	0.00348	0.00418	0.01263	0.01028
random_npa	50	50	86400	150.68	0	0	0	0	0	0	29	300.1	0	NaN	NaN								
random_thr_mc_0.8	50	50	86400	40.85	4392	0.03726	0.27	13.79	3.09	12.93	1389	924.72	1363.51	20.47	7.94	0.00152	0.00632	0.00047	0.00119	0.00201	0.0037	0.00868	0.01095
random_thr_mmt_0.8	50	50	86400	41.81	4839	0.03048	0.23	12.99	3.25	12.81	1424	929.7	1348.87	16.82	7.67	0.00011	0.0006	0.00038	0.0011	0.00249	0.00502	0.00839	0.00835
random_thr_mu_0.8	50	50	86400	44.08	5404	0.03546	0.28	12.69	2.73	12.73	1578	900.54	1253.98	20.23	8.09	0.00017	0.00075	0.00033	0.00103	0.00262	0.00388	0.00886	0.009
random_thr_rs_0.8	50	50	86400	41.12	4442	0.03592	0.27	13.16	3.03	13.18	1391	934.82	1404.86	20.52	7.96	0.00007	0.00044	0.00045	0.00106	0.00251	0.00535	0.00877	0.01113

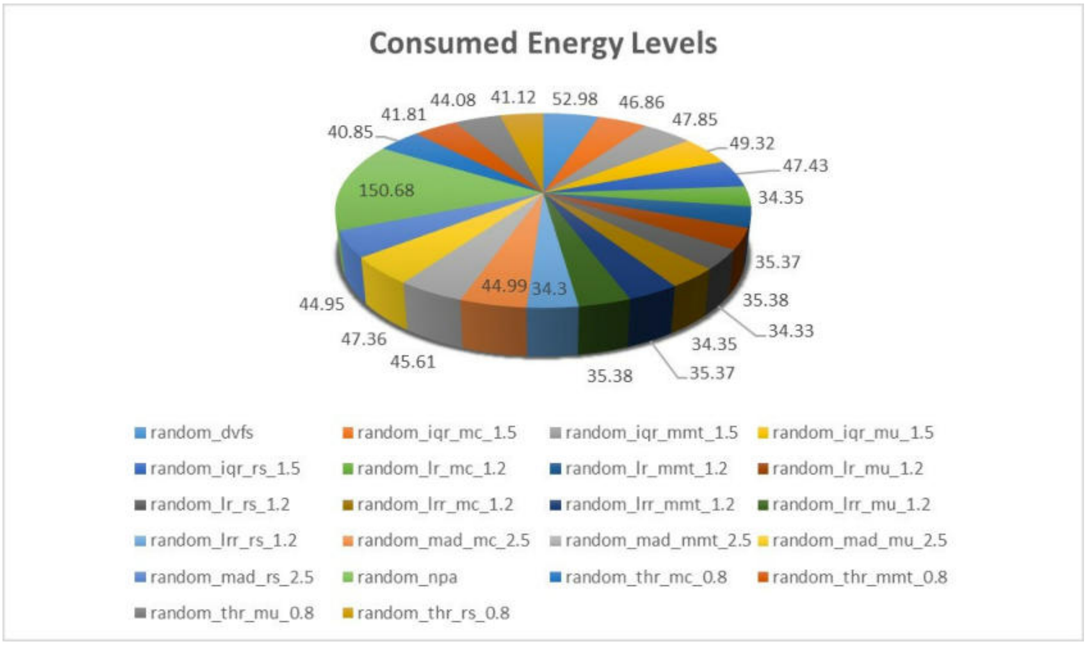


Fig. 4.24. *Consumed Energy Level per Experiment*

Fig. 4.24 shows consumed energy levels along with their experiment names.

From the Table 4.2, migration counts can also be computed and it is seen that experiment name: random_iqr_mu_1.5 involves maximum number of migration counts. Fig. 4.25 shows the migration counts for each experiment.

5. Conclusion and Future Directions. The fog computing resource allocation methods proposed in this paper combines the allocation and selection techniques altogether with optimal parameter stack to make scheduling decisions. This paper primarily focused to reduce the task load by implementing the rapid task processing, while also incorporating the sub-group oriented scheduling on available resources. This scheme is believed to improve the user contentment by improving the cost to operation length ratio, which eventually reduces the customer churn, and can effectively boost the operational revenue. The failure event tracking also plays a vital role in scheduling operations by avoiding the computing resources with high failure probability. The proposed model is learnt to reduce the queue size by effectively allocating the resources, which resulted in the form of quicker completion of user workflows. The prospective method results are evaluated against the state of the art scene with non-power aware based task scheduling mechanism. Out of the random VM allocation and selection policy, the DVFS (52.98 kWh) scheme outperforms NPA (150.68 kWh) model for the cloud task processing. Out of the particular VM allocation and selection models, which includes IQR, LR, LRR, MAD & THR. The results have obtained and analyzed using the energy, SLA infringement and workflow execution delay. The performance of the proposed schema has been analyzed in various experiments particularly designed to analyze various aspects for workflow processing on given fog resources. The LRR (35.85 kWh) model has been found most efficient on the basis of average energy consumption in comparison to the LR (34.86 kWh), THR (41.97 kWh), MAD (45.73 kWh) and IQR (47.87 kWh). The LRR model has been also observed as the leader when compared on the basis of number of VM migrations. The LRR (2520 VMs) has been observed as best contender on the basis of mean of number of VM migrations in comparison with LR (2555 VMs), THR (4769 VMs), MAD (5138 VMs) and IQR (5352 VMs).

In future, this work may not only confine to task allocation and task scheduling, but can be extended towards various load balancing algorithms that compute the load that gets generated on each VM. Moreover, this work of allocation and scheduling can be extended to the emerging technologies like bigdata to solve problems arising due to huge data in daily routine. This work may also be extended towards machine learning and deep learning

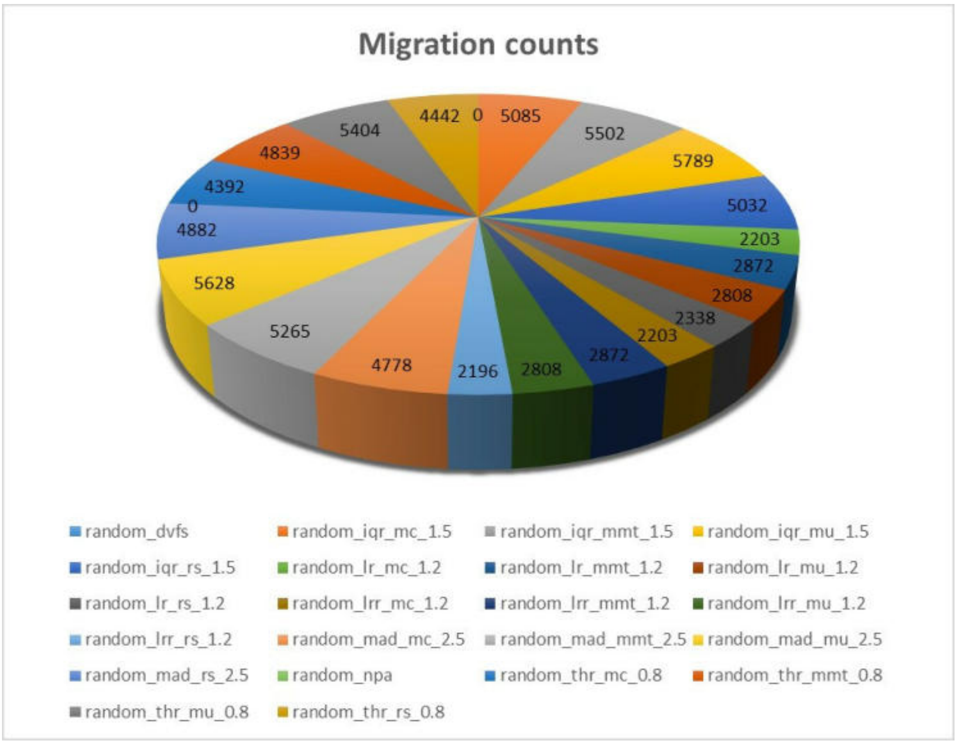


FIG. 4.25. *Migration counts per Experiment*

for pre-judgement of the upcoming difficulties and can set up a recovery/maintenance module accordingly.

REFERENCES

[1] W. A. Jansen, T. Grance, et al., Guidelines on security and privacy in public cloud computing (2011).
[2] Basic concept and terminology of cloud computing (Jan. 2015).
[3] E. A. Pansotra, E. S. P. Singh, Cloud security algorithms, International Journal of Security and Its Applications 9 (10) (2015) 353–360 (2015).
[4] A. T. Velte, T. J. Velte, R. C. Elsenpeter, R. C. Elsenpeter, Cloud computing: a practical approach, McGraw-Hill New York, 2010 (2010).
[5] S. P. Singh, A. Sharma, R. Kumar, Analysis of load balancing algorithms using cloud analyst, International Journal of Grid and Distributed Computing 9 (9) (2016) 11–24 (2016).
[6] M. Rahman, S. Iqbal, J. Gao, Load balancer as a service in cloud computing, in: 2014 IEEE 8th International Symposium on Service Oriented System Engineering, IEEE, 2014, pp. 204–211 (2014).
[7] Gartner highlights five attributes of cloud computing (Dec. 2013).
[8] A. Bala, I. Chana, Fault tolerance-challenges, techniques and implementation in cloud computing, International Journal of Computer Science Issues (IJCSI) 9 (1) (2012) 288–293 (2012).
[9] Cloud computing: A delicate balance of risk and benefit (Feb. 2015).
[10] Prerakmody, Cloud computing (Jan. 2015).
[11] L. M. Vaquero, L. Rodero-Merino, J. Caceres, M. Lindner, A break in the clouds: towards a cloud definition, ACM SIGCOMM Computer Communication Review 39 (1) (2008) 50–55 (2008).
[12] S. Patel, A. S. Singh, Fault tolerance mechanisms and its implementation in cloud computing–a review, International Journal 3 (12) (2013).
[13] A. M. Alakeel, et al., A guide to dynamic load balancing in distributed computer systems, International Journal of Computer Science and Information Security 10 (6) (2010) 153–160 (2010).
[14] W. Zhao, P. Melliar-Smith, L. E. Moser, Fault tolerance middleware for cloud computing, in: 2010 IEEE 3rd International Conference on Cloud Computing, IEEE, 2010, pp. 67–74 (2010).
[15] Q. Li, J. Zhao, Y. Gong, Q. Zhang, Energy-efficient computation offloading and resource allocation in fog computing for internet of everything, China Communications 16 (3) (2019) 32–41 (2019).
[16] S. P. Singh, A. Nayyar, R. Kumar, A. Sharma, Fog computing: from architecture to edge computing and big data

processing, The Journal of Supercomputing (2018) 1–36 (2018).
[17] M. Dorigo, Optimization, learning and natural algorithms, PhD Thesis, Politecnico di Milano (1992).
[18] M. Aazam, K. A. Harras, S. Zeadally, Fog computing for 5g tactile industrial internet of things: Qoe-aware resource allocation model, IEEE Transactions on Industrial Informatics (2019).
[19] Y. Jie, M. Li, C. Guo, L. Chen, Game-theoretic online resource allocation scheme on fog computing for mobile multimedia users, China Communications 16 (3) (2019) 22–31 (2019).
[20] X. Li, J. Wan, H.-N. Dai, M. Imran, M. Xia, A. Celesti, A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing, IEEE Transactions on Industrial Informatics (2019).
[21] S. Dam, G. Mandal, K. Dasgupta, P. Dutta, An ant colony based load balancing strategy in cloud computing, in: Advanced Computing, Networking and Informatics-Volume 2, Springer, 2014, pp. 403–413 (2014).
[22] A. Tasiopoulos, O. Ascigil, I. Psaras, S. Toumpis, G. Pavlou, Fogspot: Spot pricing for application provisioning in edge/fog computing, IEEE Transactions on Services Computing (2019).
[23] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J. P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, Journal of Systems Architecture (2019).
[24] G. Yoon, D. Choi, J. Lee, H. Choi, Management of iot sensor data using a fog computing node, Journal of Sensors 2019 (2019).
[25] Z. Ning, J. Huang, X. Wang, Vehicular fog computing: Enabling real-time traffic management for smart cities, IEEE Wireless Communications 26 (1) (2019) 87–93 (2019).
[26] S. K. Goyal, M. Singh, Adaptive and dynamic load balancing in grid using ant colony optimization, International Journal of Engineering and Technology 4 (4) (2012) 167–174 (2012).
[27] B. Donassolo, I. Fajjari, A. Legrand, P. Mertikopoulos, Fog based framework for iot service provisioning, in: 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2019, pp. 1–6 (2019).
[28] S. K. Dhurandher, M. S. Obaidat, I. Woungang, P. Agarwal, A. Gupta, P. Gupta, A cluster-based load balancing algorithm in cloud computing, in: 2014 IEEE International Conference on Communications (ICC), IEEE, 2014, pp. 2921–2925 (2014).
[29] V. S. Kushwah, S. K. Goyal, P. Narwariya, A survey on various fault tolerant approaches for cloud environment during load balancing, Int J Comput Netw Wirel Mobile Commun 4 (6) (2014) 25–34 (2014).
[30] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, R. Rastogi, et al., Load balancing of nodes in cloud using ant colony optimization, in: 2012 UKSim 14th International Conference on Computer Modelling and Simulation, IEEE, 2012, pp. 3–8 (2012).
[31] R. Mishra, A. Jaiswal, Ant colony optimization: A solution of load balancing in cloud, International Journal of Web & Semantic Technology 3 (2) (2012) 33 (2012).
[32] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, S. Dam, A genetic algorithm (ga) based load balancing strategy for cloud computing, Procedia Technology 10 (2013) 340–347 (2013).
[33] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, K. Li, An energy-efficient task scheduling algorithm in dvfs-enabled cloud environment, Journal of Grid Computing 14 (1) (2016) 55–74 (2016).
[34] F. Yuan, S. E. OOI, L. Yuto, T. Yasuo, Time task scheduling for simple and proximate time model in cyber-physical systems, in: Computational Science and Technology, Springer, 2019, pp. 185–194 (2019).
[35] Q. Zhao, C. Xiong, C. Yu, C. Zhang, X. Zhao, A new energy-aware task scheduling method for data-intensive applications in the cloud, Journal of Network and Computer Applications 59 (2016) 14–27 (2016).
[36] N. Bansal, A. Maurya, T. Kumar, M. Singh, S. Bansal, Cost performance of qos driven task scheduling in cloud computing, Procedia Computer Science 57 (2015) 126–130 (2015).
[37] G. Patel, R. Mehta, U. Bhoi, Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing, Procedia Computer Science 57 (2015) 545–553 (2015).
[38] W. Lin, C. Liang, J. Z. Wang, R. Buyya, Bandwidth-aware divisible task scheduling for cloud computing, Software: Practice and Experience 44 (2) (2014) 163–174 (2014).
[39] G. Xu, J. Pang, X. Fu, A load balancing model based on cloud partitioning for the public cloud, Tsinghua Science and Technology 18 (1) (2013) 34–39 (2013).
[40] Z. Liu, X. Wang, A pso-based algorithm for load balancing in virtual machines of cloud computing environment, in: International conference in swarm intelligence, Springer, 2012, pp. 142–147 (2012).
[41] K. Li, G. Xu, G. Zhao, Y. Dong, D. Wang, Cloud task scheduling based on load balancing ant colony optimization, in: 2011 Sixth Annual ChinaGrid Conference, IEEE, 2011, pp. 3–9 (2011).
[42] H. Chang, X. Tang, A load-balance based resource-scheduling algorithm under cloud computing environment, in: International Conference on Web-Based Learning, Springer, 2010, pp. 85–90 (2010).
[43] V. Šešum-Čavić, E. Kühn, Self-organized load balancing through swarm intelligence, in: Next Generation Data Technologies for Collective Computational Intelligence, Springer, 2011, pp. 195–224 (2011).
[44] A. Jain, R. Singh, An innovative approach of ant colony optimization for load balancing in peer to peer grid environment, in: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), IEEE, 2014, pp. 1–5 (2014).
[45] R. Chaukwale, S. S. Kamath, A modified ant colony optimization algorithm with load balancing for job shop scheduling, in: 2013 15th International Conference on Advanced Computing Technologies (ICACT), IEEE, 2013, pp. 1–5 (2013).
[46] S. Razzaq, A. Wahid, F. Khan, N. ul Amin, M. A. Shah, A. Akhunzada, I. Ali, Scheduling algorithms for high-performance computing: An application perspective of fog computing, in: Recent Trends and Advances in Wireless and IoT-enabled Networks, Springer, 2019, pp. 107–117 (2019).

[47] J. Wang, D. Li, Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing, Sensors 19 (5) (2019) 1023 (2019).
[48] J. Luo, L. Yin, J. Hu, C. Wang, X. Liu, X. Fan, H. Luo, Container-based fog computing architecture and energy-balancing scheduling algorithm for energy iot, Future Generation Computer Systems (2019).
[49] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, W. Dai, Energy efficient task allocation and energy scheduling in green energy powered edge computing, Future Generation Computer Systems 95 (2019) 89–99 (2019).
[50] C. Li, J. Tang, H. Tang, Y. Luo, Collaborative cache allocation and task scheduling for data-intensive applications in edge computing environment, Future Generation Computer Systems 95 (2019) 249–264 (2019).
[51] C. Li, J. Bai, J. Tang, Joint optimization of data placement and scheduling for improving user experience in edge computing, Journal of Parallel and Distributed Computing 125 (2019) 93–105 (2019).
[52] Y. Deng, Z. Chen, X. Yao, S. Hassan, J. Wu, Task scheduling for smart city applications based on multi-server mobile edge computing, IEEE Access 7 (2019) 14410–14421 (2019).
[53] S. Javaid, N. Javaid, T. Saba, Z. Wadud, A. Rehman, A. Haseeb, Intelligent resource allocation in residential buildings using consumer to fog to cloud based framework, Energies 12 (5) (2019) 815 (2019).
[54] L. Li, Q. Guan, L. Jin, M. Guo, Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system, IEEE Access 7 (2019) 9912–9925 (2019).
[55] R. Mahmud, R. Buyya, Modeling and simulation of fog and edge computing environments using ifogsim toolkit, Fog and Edge Computing: Principles and Paradigms (2019) 433–465 (2019).
[56] B. Singh, O. Gupta, S. P. Singh, Performance evaluation of dns based load balancing techniques for web servers (2011).
[57] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, M. Song, Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing, IEEE Transactions on Wireless Communications 18 (1) (2019) 695–708 (2019).
[58] P. T. I. M. C. COMPUTING, An efficient job sharing strategy for prioritized tasks in mobile cloud computing environment using acs-js algorithm, Journal of Theoretical and Applied Information Technology 97 (4) (2019).
[59] L. Yang, B. Liu, J. Cao, Y. Sahni, Z. Wang, Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds, IEEE Transactions on Services Computing (2019).
[60] S. Kim, Novel resource allocation algorithms for the social internet of things based fog computing paradigm, Wireless Communications and Mobile Computing 2019 (2019).
[61] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, J. Rodriguez, Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach, IEEE Transactions on Vehicular Technology (2019).
[62] B. Mondal, K. Dasgupta, P. Dutta, Load balancing in cloud computing using stochastic hill climbing-a soft computing approach, Procedia Technology 4 (2012) 783–789 (2012).
[63] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and experience 41 (1) (2011) 23–50 (2011).
[64] K. Braekers, R. F. Hartl, S. N. Parragh, F. Tricoire, A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience, European Journal of Operational Research 248 (2) (2016) 428–443 (2016).
[65] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future generation computer systems 28 (5) (2012) 755–768 (2012).
[66] S. Zaman, D. Grosu, A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds, IEEE Transactions on Cloud Computing 1 (2) (2013) 129–141 (2013).
[67] S. Zaman, D. Grosu, Combinatorial auction-based dynamic vm provisioning and allocation in clouds, in: 2011 IEEE Third International Conference on Cloud Computing Technology and Science, IEEE, 2011, pp. 107–114 (2011).
[68] Z. Cao, S. Dong, Dynamic vm consolidation for energy-aware and sla violation reduction in cloud computing, in: 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE, 2012, pp. 363–369 (2012).
[69] K. Kumar, J. Feng, Y. Nimmagadda, Y.-H. Lu, Resource allocation for real-time tasks using cloud computing, in: 2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN), IEEE, 2011, pp. 1–7 (2011).
[70] A. Beloglazov, R. Buyya, Energy efficient allocation of virtual machines in cloud data centers, in: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE, 2010, pp. 577–578 (2010).
[71] A. Beloglazov, R. Buyya, Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints, IEEE Transactions on Parallel and Distributed Systems 24 (7) (2013) 1366–1379 (2013).

Edited by: Pijush Kanti Dutta Pramanik
Received: Mar 18, 2019
Accepted: Apr 2, 2019

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:
- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:
- programming environments,
- debugging tools,
- software libraries.

Performance:
- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:
- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:
- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in LaTeX 2ε using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.