Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2020 SCPE. Volume 21, Issue 1, pp. 73–84, DOI 10.12694:/scpe.v21i1.1616

MINIMIZING DEADLINE MISSES AND TOTAL RUN-TIME WITH LOAD BALANCING FOR A CONNECTED CAR SYSTEMS IN FOG COMPUTING

K. JAIRAM NAIK* AND D. HANUMANTH NAIK†

Abstract. Cloud computing helps in providing the applications with a few number of resources that are used to unload the tasks. But there are certain applications like coordinated lane change assistance which are helpful in cars that connects to internet has strict time constraints, and it may not be possible to get the job done just by unloading the tasks to the cloud. Fog computing helps in reducing the latency i.e the computation is now done in local fog servers instead of remote datacentres and these fog servers are connected to the nearby distance to clients. To achieve better timing performance in fog computing load balancing in these fog servers is to be performed in an efficient manner.

The challenges in the proposed application includes the number of tasks are high, client mobility and heterogeneous nature of fog servers. We use mobility patterns of connected cars and load balancing is done periodically among fog servers. The task model presented here in this paper solves scheduling problem and this is done at the server level and not on the device level. And at last, we present an optimization problem formulation for balancing the load and for reducing the misses in deadline, also the time required for running the task in these cars will be minimized with the help of fog computing. It also performs better than some common algorithms such as active monitoring, weighted round robin and throttled load balancer.

Key words: fog computing, connected car, offloading, mobility prediction, task set, load balancing, deadline miss, link rate, optimization

AMS subject classifications. 68M14, 90C26

1. Introduction.

In cloud computing the distance which is present in between the datacentres and the clients is high as compared to that of fog computing. It is difficult to run some time sensitive applications in cloud as the latency that cloud computing posseses is very high [1]. Fog computing processes the applications closer to the users within the proximity distance [2]. Cloud computing servers are placed in the datacentre which are centralized whereas in fog computing, the servers are distributed to the edge. Fog computing is a distributed computing paradigm that provides the users with cloud services at the edge of a network. This helps in the reduction of traffic on the backbone of network.

But there are certain issues in fog computing while considering the deadline misses and runtime which includes resource allocation and load balancing [3]. To explore the problems in fog computing such as load balancing, we here use the motivating application such as connected cars. Fog computing benefits the connected cars as coordinated lane change assistance in connected cars uses the local fog servers for data processing and responds quickly [4, 5, 6].

Load balancing in fog computing is performed among the fog servers to reduce deadline misses as well as runtime in the applications like connected cars. This helps in improving the performance of connected cars by providing with the accurate results. While load balancing is an issue to be performed with the fog servers, we here in this work develop a task model as well as optimization problem formulation for dealing with issues like load balancing and resource allocation. The result obtained with this technique outperforms common scheduling algorithms.

Fog computing characteristics are the followings:

1. The nature of the fog servers are heterogeneous that is each fog server will be having different connectivity and computation capability.

*Assistant Professor, Department of C.S.E, National Institute of Technology (NIT), Raipur (CG), INDIA (jairam.524@gmail.com)

†Software Engineer, Infosys Technologies, Bangalore, INDIA (hanumanthdhanavath6@gmail.com)

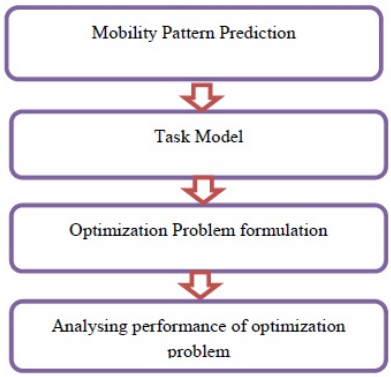

FIG. 1.1. *Workflow diagram*

2. The scale of the fog computing systems is relatively small as compared to a datacentre. This helps in more precise approaches in dealing with the applications.

Here in fog computing the client's position is not static and mobility of the client is also a feature of fog computing.

Connected car is a car that has internet access, and also equipped with wireless local area network. This helps the car and provides with internet access, and data can be shared, with other devices both in and outside of the vehicle. These connected cars ensure the safety in cars. They help in reducing the accidents.

The relationship between the server and the clients in fog computing depends on their instantaneous locations. The information of client and their mobility is used to perform load balancing. And load balancing is done by predicting the travel patterns of the car and allocating the resources that fits the server to the best and this helps us to reduce processing time and deadline misses can also be abandoned.

We first develop the algorithm for mobility prediction and propose a model that is the resemblance of task model and finally we implement optimization problem for load balancing that is useful in connected cars. The scheduling is done at the server level. In the task model that is proposed we develop an algorithm that requires no knowledge of how the tasks are being scheduled earlier. It is tedious process to obtain the information regarding the scheduling of each task because the traffic present in the network increases more rapidly and the problem size increases. The final step in the workflow diagram is we perform comparison of our proposed work with the common scheduling algorithms like throttled load balancer. Here we analyse the performance of the optimization problem.

The workt scope is high in the connected as well as the autonomous vehicles. It helps in connecting the mobile devices with the vehicles and providing numerous advantage for a user. Fog computing has many applications like traffic control system, health care systems, machinery related systems, video streaming system, smart city and smart home.

Further development is in the progress related to this fog computing.

In this work we will follow the flow of the given solution in the particular order. We will first review existing work that is already there in the load balancing for mobile clients and is developed with the help of fog computing. This is presented in the Literature review section 2.

The proposed system is developed is described in Section 3. It consists of Linear mobility prediction algorithm and it gives the accurate results and shows that online task scheduling can benefit the end user. In this chapter we also present our task model as well as the optimization problem formulation.

In Section 4 result is analysed with the common scheduling algorithms that is throttled load balancer. Conclusions and Future scope are being discussed in Section 5.

2. Related Work. The work proposed by Chen et al [7] is related to connected cars which has numerous advantages with the help of fog computing. Here, a prediction algorithm is proposed which locates a car location and load balancing is done at server level to minimize deadline misses and runtime.

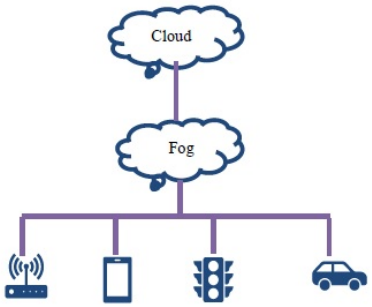

FIG. 1.2. *Architecture of fog computing*

The work presented by Oueis et al [8] is that, constructing two methods which helps in optimizing and in minimizing total transmitting power and total time required for computation. The tasks here are partitioned and distributed among the servers and they are hard deadline constrained, but in the work that we implement further is constrained to soft deadlines and are indivisible.

Zeng et al [9] work is related to the formulation of linear program and it helps in minimizing the average task finishing time but, in our work, our target is the avoidance of deadline misses. Hong et al [10] proposes a model which takes workload for dynamic scaling of the fog system to provide them with available number of resources. In our work we impose deadline constraints and placing of only one task for a fog servers. This is the difference between their and our work.

Takayuki et al [11] uses the routing of smart cars as an application and offloads them to the local servers. Here the energy constraints are taken into account and formulation for implementing optimizing problem is performed. We impose the number of tasks that fit to the server and also constraints on resources, while in the work of Takayuki this is not considered.

Hong et al [12] uses the patterns of the cars and predicts its location to which it is travelling and forward the tasks to local server the fog server that is in the range of obtained location so that it processes the tasks without delay. In this work the focus is only on single client and resource and capacity constraints are not imposed whereas in our work the system is multiclient and it has various constraints on the servers.

Wang et al [13] uses facial recognising system that runs on the handheld devices. They treat the tasks of these devices as a graph and then solves which particular element is matching with other. The difference in their work and our work is that we impose constraints on time and it is multiuser model while their work is not.

Li et al [14] solves the problem of partitioning the tasks for remote cloud and local servers and then allocation of resources based on the information obtained on partitioning. Their work includes optimization formulation for minimizing the finishing time, processing cost and bandwidth. But the disadvantage of their work is they do not include deadline misses.

3. Predicting the Pattern of Mobility.

3.1. Model for connected car. Most of work on predicting the movement as well as direction in vehicles are done using the methods Markov Chain [15] or probability distributions [16]. Here we present a simple linear model which gives high accuracy in predicting their locations.

The example model that we present is as given in Fig 3.1. Hexagon represents wireless coverage area of fog server; Rectangle represents fog server; and Circles represents connected cars.

In the model presented above there are three different fog servers and the connected cars which are shown as circle are counted as clients. The fog servers manage the cars to the range upto which it can handle in wireless connection. Here we apply 802.11p [18] protocol in between the tasks and servers, Wireless Access in Vehicular Environments (WAVE) [17].The range is up to 500 metres. The hexagon shape is applicable in implementation of system of network in wireless environment [19].

TABLE 2.1
Tabular representation of existing works

Authors	Work proposed	Advantages	Limitations
Oueis et al.	Optimization problem	Minimize total transmitting power and computation time	Tasks are partitioned and constrained to hard deadlines
Hong et al.	Programming model for dynamic scaling	Scaling fog devices to provide enough resources	Deadline constraints are not considered
Takayuki et al.	Routing for smart cars	Minimizes aggregated task finishing time	Capacity and resource constraint on server is not included
Hong et al.	Car's mobility pattern to predict future location	Processing speed is increased	System focuses on only a single client
Li et al.	Allocation of resources based on partitioning	Minimizes finishing time and cost of processing	System does not allow any deadline misses

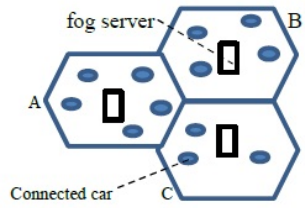

FIG. 3.1. *Fog computing model example*

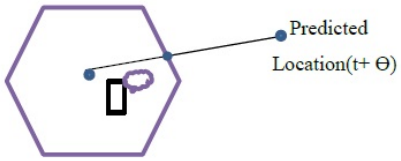

FIG. 3.2. *predicted algorithm for mobility*

3.2. Prediction for pattern of linear mobility. Here the algorithm for predicting linear mobility is proposed that identifies the cars future location with the help of its previous location. We assume that a car updates its location for theta seconds and if the updating of cars location at a particular time t, the fog server gets the timestamp and cars previous location to calculate the direction and the speed of the car. So, the updated location of this system will be its position at timestamp $t + \theta$. If the location obtained is beyond the range of fog server, then the algorithm will identify which nearest server that is closest to the predicted location of the car.

The GPS data of taxis in Rome, Italy [20] is compared with the algorithm we implemented. We divide the region into seven fog nodes with each node consisting of radius 500 metres. With the help of this prediction algorithm we notify that to which of the six servers the car is heading towards leaving the middle server. We get an accuracy of 94.90%.

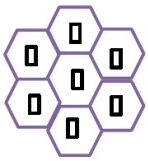

FIG. 3.3. *Fog system*

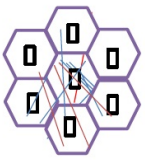

FIG. 3.4. *Graphical result of linear prediction*

Here in this algorithm we see that after calculating the speed and the direction of the car we predict that to which server location the car is travelling to and we allocate the resources prior to its arrival.

In Figure 3.4 the lines given in blue represent correct prediction and the lines in red colour are the incorrect prediction. So this is the result obtained considering the GPS data of taxis.

The obtained accuracy tells us that how the prediction algorithm we implemented is giving the results. It is the simpler algorithm as mentioned with markov chain and probabilistic distributions.

4. Load Balancing.

4.1. Model for the tasks in the system. In applications like coordinated lane change assistance, the data must be ready for the car when it is connected to the new fog server. With the help of mobility prediction algorithm,we try to figure out to which server the car is travelling and perform load balancing before the car reaches the location to reduce the runtime.

There are fog server, cars and remote cloud server present in this task model. And there is interconnection between them.In task model there are three local fog servers and one cloud server. Cars are connected with the local fog servers that are in the specific range given by the hexagonal region.

The workload of a car is treated as a single task. These tasks are either processed by the servers or passed to nearest servers with high capability.

In the task model (Fig 4.1) five tasks are assigned to server A, there are four tasks in server B and three tasks in server C and cloud server is initially passed with no tasks.

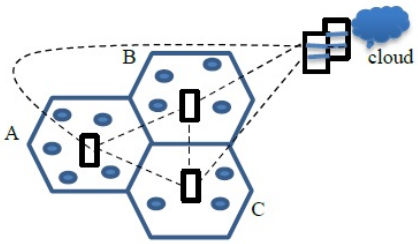

FIG. 4.1. *Task model example*

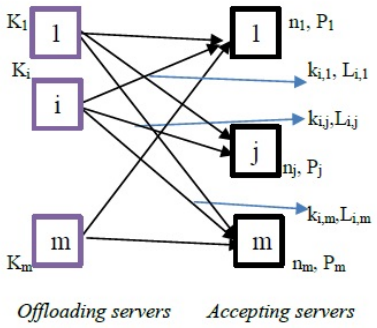

FIG. 4.2. *Task model in fog computing graphically*

TABLE 4.1
Variables of the model

Variables used inbetween servers and tasks	
m	Total servers
K_i	Tasks initially assigned to server i
$L_{i,j}$	Link rate present inbetween the server i and j (tasks/sec)
P_j	server j capacity(tasks)
y	CPU cycles of a task (M cycles)
n_j	Frequency of CPU at the accepting server (MHz)
D	Total tasks deadline (sec)
t	Time for processing of each server (sec)
Optimizing variables	
$K_{i,j}$	Total tasks distribution number from server i to server j

In the model shown in graphical format in Figure 4.1:

$k_{i,j}$ = Total task distribution from one server to another,
Li, j=Link rate from server i to server j

The local servers in the task model are interconnected and these servers can distribute the tasks among the other servers or the remote cloud. This implies that the servers in the task model can offload as well as accept the tasks from other servers. All the servers are allotted with various constraints and resources.

In Figure 4.2, server i offloads its tasks to the accepting server j. These tasks are further processed by the server j.

The fog nodes are placed in between the inbetween the cloud and the edge devices. The fog servers are placed as in the Figure 4.2 and these servers are static. So the total number of servers here in the system is 8 which includes the cloud server.

Table 4.1 shows all the variables that are being used in this model.

From the above table m represents the servers count in the model, K_i is tasks initially assigned to server i, $L_{i,j}$ speed of data transmission present inbetween the server i and j (tasks/sec).

Consider the task model example in which $m = 4$, and $KA = 5, KB = 4, KC = 3$ and $Kcloud = 0, P_j$ is the capacity of server j, y is cycles number required for CPU that consume bandwidth equally, n_j, frequency of CPU at the accepting server. D is total tasks deadline and t is the time for processing a server. Here t is required because load balancing can be done based on prediction algorithm in which periodic calculation is

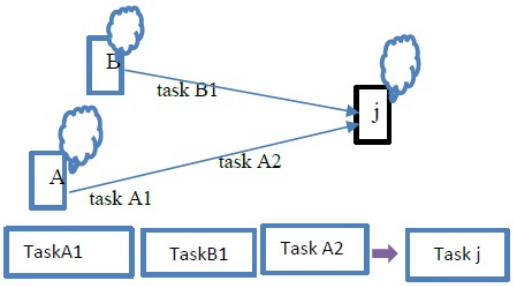

FIG. 4.3. *Aggregated task example*

done. Each time when the prediction is done load balancing is to be done for sure. The condition here is that time taken to process the total tasks is to be less than the time that will remain that is $D - t$, and this time is helpful for implementing the prediction algorithm and balancing the load. $k_{i,j}$ is the optimization variable that is to be solved and it tells that total tasks distribution number from one server to another.

The drawback of this model is it is difficult to predict the scheduling of tasks by the server that are being sent by different servers and their arriving pattern of these tasks is also bit complex. So, it is difficult in calculating the complete time the task is executed. The solution is to balance the load not at device level but by making the tasks as one large aggregated task it can be resolved. By doing this it becomes easy to calculate the time taken to complete one aggregate task at each server in contrast with how the server is scheduling its tasks. Figure 4.3 shows the aggregated task example.

Aggregate task transmission time at server j:

$$\textit{Turnaround time: } \sum_{i=1}^{m} \frac{k_{ij} \cdot y}{n_j} \tag{4.1}$$

Time taken for offloading server i for transmitting the tasks.

(k_{ij}) to accepting server is division between $k_{i,j}$ and the link rate (L_{ij}). The total time taken for all the combination of tasks at the accepting server is total sum of time required for transmission. The equation is given in equation (4.1).

Aggregate task turnaround time at server j:

$$\textit{Turnaround time: } \sum_{i=1}^{m} \frac{k_{ij} \cdot y}{n_j} \tag{4.2}$$

The processing time required for the accepting server j for all the tasks that are being shared by other servers.

Aggregate task completion time at server j:

$$\textit{Completion time: } \sum_{i=1}^{m} \frac{k_{ij}}{L_{ij}} + \sum_{i=1}^{m} \frac{k_{ij} \cdot y}{n_j} \tag{4.3}$$

Completion time is known as the time taken for server j to complete its aggregate tasks. It is the addition of turnaround time and transmission time.

Aggregate task lateness server j:

$$\textit{Lateness at server } j\textit{: } \left(\sum_{i=1}^{m} \frac{k_{ij}}{L_{ij}} + \sum_{i=1}^{m} \frac{k_{ij} \cdot y}{n_j} \right) - D \tag{4.4}$$

Lateness is calculated by subtracting the deadline of a task from its completion time.

4.2. Problem formulation. Here in the problem of load balancing the final result is minimizing the misses in deadline and runtime. The objective function is

$$\textbf{Min}: \sum_{j=1}^{m}\left[\left(\sum_{i=1}^{m}\frac{k_{ij}}{L_{ij}}+\frac{k_{ij}\cdot y}{n_j}\right)-D+v\cdot\max\cdot\max\left(0,\left(\sum_{i=1}^{m}\frac{k_{ij}}{L_{ij}}+\frac{k_{ij}\cdot y}{n_j}\right)-D\right)\right] \tag{4.5}$$

The goal for optimizing is to minimize total runtime and deadline misses. Minimizing the term lateness of completing the aggregated tasks is reflected in minimizing the total runtime in the objective function.

If we minimize the second term $(v\cdot\max*\max\left(0,\left(\sum_{i=1}^{m}\frac{k_{ij}}{L_{ij}}+\frac{k_{ij}\cdot y}{n_j}\right)\right))$ then it minimizes the misses in deadline. v is the weighing factors so that more emphasis is put to minimize the deadline misses and runtime. From the objective function above the term $v*\max\left(0,\left(\sum_{i=1}^{m}\frac{k_{ij}}{L_{ij}}+\frac{k_{ij}\cdot y}{n_j}\right)-D\right)$ is taken as $v\cdot x_j$.

Therefore, the objective function now becomes:

$$\textbf{Min}: \sum_{j=1}^{m}\left[\left(\sum_{i=1}^{m}\frac{k_{ij}}{L_{ij}}+\frac{k_{ij}\cdot y}{n_j}\right)-D+v\cdot x_j\right] \tag{4.6}$$

The constraints subjected to the above given objective function are:

$$\sum_{i=1}^{m}k_{ij}\cdot y\le n_j\cdot t \quad \text{for } j\in\{1,2,\dots m\} \tag{4.7}$$

$$\sum_{i=1}^{m}k_{ij}\le P_j \quad \text{for } j\ \in\{1,2,\dots m\} \tag{4.8}$$

$$\sum_{j=1}^{m}k_{ij}=K_i \quad \text{for } i\ \in\{1,2,\dots m\} \tag{4.9}$$

$$k_{ij}\in Z+ \quad \text{for } i\ \in\{1,2,\dots m\},\ \text{for } j\ \in\{1,2,\dots m\} \tag{4.10}$$

$$x_j\ge 0 \quad \text{for } j\ \in\{1,2,\dots m\} \tag{4.11}$$

$$x_j\ge\left(\sum_{i=1}^{m}\frac{k_{ij}}{L_{ij}}+\frac{k_{ij}\cdot y}{n_j}\right)-D \quad \text{for } j\ \in\{1,2,\dots m\} \tag{4.12}$$

Equation (4.7) states that there are there sufficient number of cycles for CPU are present for offloading tasks. Constraint Equation (4.8) is about that all tasks loaded to server will be not exceeding its capability to hold the tasks. Equation (4.9) will make sure that the tasks are distributed and are processed. [Eq-10]states that the tasks are not divided to subtasks. Here in this methodology we will not be partitioning the tasks i.e the tasks will remain itself and not gets divided. We solve the optimization function using lp_solve.

We have implemented the above given constraints and objective function with the help of IBM ILOG CPLEX optimization studio. We solve for k_{ij} that is the task distribution number from one server that offloads to another server that accepts.

5. Results and discussion.

5.1. Experimental results. The simulation in this experiment adopts a system that consists of a cloud server and the fog servers are seven. The simulation parameter values are given in the table 5.1.

The taskset generation is randomly done by selecting number for Ki from a range of integers. Different loadings can be created for the system by varying the integers. Optimizing principle is applied to every taskset that is created newly.

The experiments performed are implemented on machine that supports AMD operation with quadcore of dual nature that has the specification of 2.3 GHz and 16GB memory. The time taken to execute both optimizing principle and predicting algorithm on this machine is below 0.01 seconds. It satisfies the remaining time $D - t$ which has a value of 0.02 and here the experiments are run below that time which is in the range of overhead budget.

5.2. Comparison between deadline misses and total runtime. Here we vary the values of weighing parameter. In the objective function from equation (4.6) a higher v value results in lesser misses in deadline with high runtime.

The result obtained by varying v is shown in below graphs. From the below graphs we see that as the value of v increases, the deadline misses count decreases and the total runtime increases.

From the above equation (4.6) as the value of v (weighing parameter) increases then the term total runtime value gets reduced and the value which represents penalty in tardiness i.e. the deadline misses value gets reduced.

5.3. Comparison with throttled load balancing algorithm. We perform comparison of our optimization result with commonly used load balancing algorithm throttled load balancer [23].

In throttled load balancing the task is processed by the server which is best suitable, the server with lowest

TABLE 5.1
Parameters for simulation

System parameters for simulation		
m	8	Total servers
y	35	CPU cycles of a task (M cycles)
D	0.5	Total tasks deadline (sec)
t	0.48	Time to process a task at each server (sec)
Local servers parameters for simulation		
K_i	Table 4.2	Tasks initially assigned for a server
$L_{i,j}$	[16,64]	Speed of transmission of tasks inbetween the servers, for $i \neq j$(tasks/sec)
$L_{i,j}$	∞	Speed of transmission of tasks inbetween the servers, for $i = j$ (tasks/sec)
P_j	[10,35]	server j capability to hold the tasks (tasks)
n_j	[2700,3600]	CPU frequency (MHz)
Cloud server parameters for simulation		
Kcloud	0	Initial tasks of cloud
Lcloud	4	Link rate present inbetween the server and cloud
Pcloud	∞	Capacity of cloud
ncloud	4500	Cloud server frequency of CPU
Variable of optimization		
$k_{i,j}$	Optimization solve	Total tasks distribution number inbetween servers.

TABLE 5.2
Workloads

Range for Ki	[10,26]	[10,28]	[10,30]	[10,32]	[10,34]
Total tasks	12687	13472	14143	14739	15051

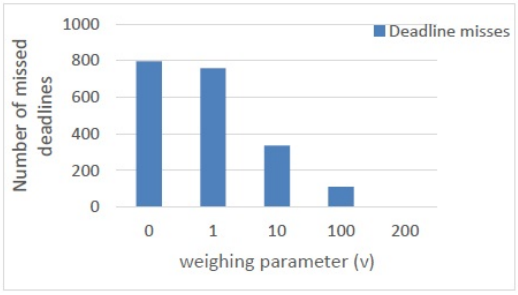

FIG. 5.1. *Comparison of deadline miss and weighing parameter (v)*

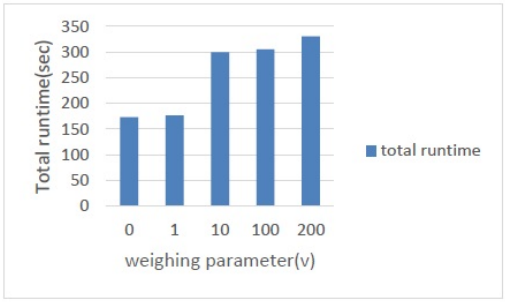

FIG. 5.2. *Comparison of runtime and weighing parameter(v)*

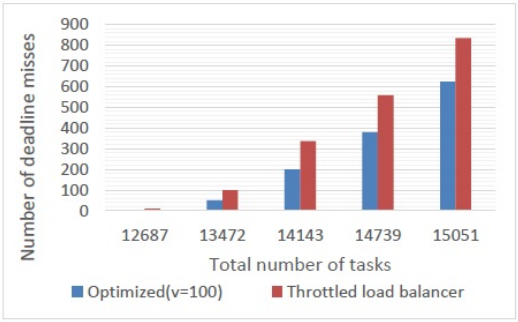

FIG. 5.3. *Deadline misses comparison with throttled load balancer*

completion time is assigned with the new task that did not reach its maximum capacity. The cloud server is used only when the local servers are not available.Our optimization outperforms throttled load balancer by almost 50% considering deadline misses as a parameter.

As the value of the number of tasks increases then the performance percentage of optimized as compared to throttled load balancer get reduced to 25%. The reason for this is that if the load of a system increases then the options get reduced for task distribution. So, the performance of optimized algorithm outperforms 25% which is reduced half as the number of tasks increases. But the optimized performs better than the common scheduling algorithm.

In Figure 5.4 the locations of the taxi drivers are plotted and with the given datasets [20] and verified if

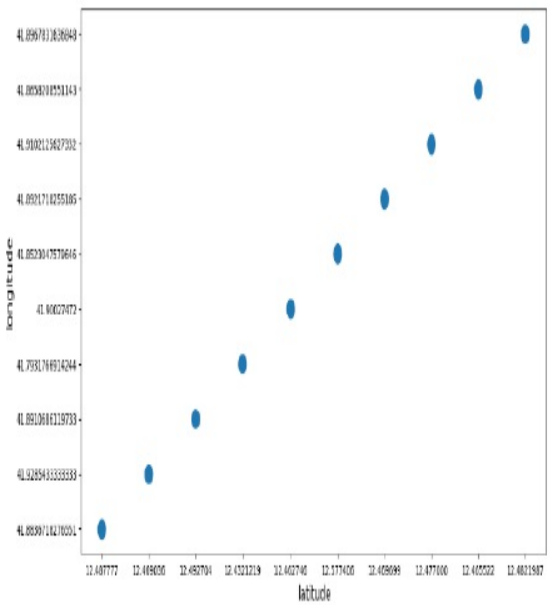

FIG. 5.4. *Graphical result for linear prediction*

the user is travelling to the predicted server or not.

6. Conclusion and future work. We performed scheduling model in fog for the connected cars. This model schedules the tasks at server level instead of device level. The calculation of the completion time of a task at device level will be a stochastic process as it is difficult when the tasks are high in number and the arrival order of the tasks is also difficult to predict. So, we combine the tasks into one aggregated task at server level and perform load balancing.

Optimization load balancing formulation is also discussed for reducing the total runtime and minimizing the deadline misses. We showed that our optimization performs better than common scheduling algorithms like throttled load balancer.

Fog and edge computing are the emerging techniques for data processing in IOT. It also helps in reducing latency and network congestion thus reducing the traffic at the backend cloud so that the processing speedup is done.

There is a scope to make the cars more updated as well as other application which can benefit from fog computing.

Fog computing is expanding to provide services in connected cars. There is high potential in the work related to fog computing and the technology being implemented in connected cars.

For simulation purposes we use fogsim which is inbuilt with libraries and basic functionalities to execute the applications which helps users with numerous advantages.

Fog computing has many applications like Traffic control system, Health care systems machinery related systems, video streaming system, smart city and smart home.

REFERENCES

[1] TAHERIZADEH, S., JONES, A. C., TAYLOR, I., ZHAO, Z., & STANKOVSKI, V. (2018). Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review. Journal of Systems and Software, 136, 19-38.

[2] F. BONOMI, R. MILITO, J. ZHU, AND S. ADDEPALLI, Fog computing and its role in the internet of things, in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, USA: ACM, 2012, pp. 13–16. doi: 10.1145/2342509.2342513

[3] P. Mach and Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, in *IEEE Communications Surveys Tutorials*, 2017.
[4] K. Kai, W. Cong, and L. Tao, Fog computing for vehicular ad-hoc networks: paradigms, scenarios, and issues, *The Jour-nal of China Universities of Posts and Telecommunications*, vol. 23, no. 2, pp. 56 – 96, 2016.
[5] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, Software defined networking-based vehicular adhoc network with fog computing, in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 1202–1207.
[6] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions, *IEEE Communications Surveys Tutorials*, vol. 13, no. 4, pp. 584–616, Fourth 2011.
[7] Chen, Yu-An, John Paul Walters, and Stephen P. Crago. Load balancing for minimizing deadline misses and total runtime for connected car systems in fog computing. *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*. IEEE, 2017.
[8] J. Oueis, E. C. Strinati, and S. Barbarossa, The fog balancing: Load distribution for small cell cloud computing, in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015, pp. 1–6.
[9] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system, *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, Dec 2016.
[10] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwalder, and B. Koldehofe, Mobile fog: A programming model for large-scale applications on the internet of things, in *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, ser. MCC '13. New York, NY, USA: ACM, 2013, pp. 15–20. doi: /10.1145/2491266.2491270
[11] T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam, Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud, in *Proceedings of the First International Workshop on Mobile Cloud Computing & # 38; Networking*, ser. MobileCloud '13. New York, USA: ACM, 2013, pp. 19–26. doi: 10.1145/2492348.2492354
[12] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwalder, and B. Koldehofe, Opportunistic spatio-temporal event processing for mobile situation awareness, in *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, ser. DEBS '13. New York, USA: ACM, 2013, pp. 195–206. doi: 10.1145/2488222.2488266
[13] S. Wang, M. Zafer, and K. K. Leung, Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.
[14] L. Chunlin, L. Yanpei, and L. Youlong, Energy-aware cross-layer resource allocation in mobile cloud," *International Journal of Communication Systems*, vol.30, no.12, pp. e3258–n/a, 2017, e3258 IJCS-16-0378.R1. doi: 10.1002/dac.3258
[15] P.Salvador and A.Nogueira, Markov modulated bi-variate gaussian processes or mobility modelling and location prediction" in proceedings of the 10th International IFIP TC 6 conference on networking -volume part 1 ser NETWORKING'11, Berling Heidalberg : Springer-Verlag, 2011, pp. 227-240.
[16] D. Stynes, K. N. Brown, and C. J. Sreenan, A probabilistic approach to user mobility prediction for wireless services," in *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Sept 2016, pp. 120–125.
[17] Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments," *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pp. 1–51, July 2010.
[18] J. Gozalvez, M. Sepulcre, and R. Bauza, Ieee 802.11p vehicle to infrastructure communications in urban environments, *IEEE Communications Magazine*, vol. 50, no. 5, pp. 176–183, May 2012.
[19] K. B. Baltzis, Hexagonal vs circular cell shape: a compar-ative analysis and evaluation of the two popular modeling approximations, in *Cellular Networks-Positioning, Perfor-mance Analysis, Reliability*. InTech, 2011.
[20] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, CRAWDAD dataset roma/taxi (v. 2014-07-17), Available at http://crawdad.org/roma/taxi/20140717
[21] M. Berkelaar, K. Eikland, P. Notebaert et al., lpsolve: Open source (mixed-integer) linear programming system, *Eindhoven U. of Technology*, 2004.
[22] Gupta, Harshit, et al. 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.
[23] S. G. Domanal and G. R. M. Reddy, Load balancing in cloud computing using modified throttled algorithm, in *2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, Oct 2013, pp. 1–5.

Edited by: Swaminathan JN
Received: Oct 6, 2019
Accepted: Feb 11, 2020