DOI 10.12694/scpe.v19i3.1375
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 19, Number 3, pp. 245–257. http://www.scpe.org

SIGNIFICANCE OF HIERARCHICAL AND MARKOV CLUSTERING IN GROUPING-AWARE DATA PLACEMENT FOR DATA INTENSIVE APPLICATIONS WITH INTEREST LOCALITY *

SHANMUGASUNDARAM VENGADESWARAN† AND SADHU RAMAKRISHNAN BALASUNDARAM‡

Abstract. During the execution of complex queries, the execution time increases exponentially, resulting in more waiting time for the user, which may sometimes extend to hours or even days in the worst cases. By virtue of their parallel and distributed computing capability, Hadoop and Spark are considered as an ideal solution for such complex query processing. Even though they are considered as an efficient solution for complex query processing, they have their own limitations when the data to be processed exhibits interest locality (i.e.) the data required for any query execution follows grouping behaviour wherein only a part of the BigData is accessed frequently. Since the data placement provided by these frameworks does not consider interest locality, it is possible that the dependent blocks required for execution will be concentrated within fewer computing nodes, resulting in several lacunas such as underutilisation of resources, and increased query execution time. Hence this paper proposes an Optimal Data Placement (ODP) Strategy based on grouping semantics. The significance of different clustering techniques viz. k-means, Hierarchical Agglomerative Clustering (HAC) and Markov Clustering (MCL), in grouping-aware data placement for data intensive applications with interest locality has been examined in this paper. Initially, the user access pattern is identified by dynamically analysing the history log. Then, clustering techniques (k-means, HAC and MCL) are separately applied over the access pattern to obtain independent clusters. These clusters are interpreted and validated to extract the Optimal Data Groupings (ODG). Finally, the proposed strategy reorganises the default data layouts in Hadoop Distributed File System (HDFS) based on ODG to achieve maximum parallel execution per group subjective to Load Balancer and Rack Awareness. Our proposed strategy is tested in 10 node cluster placed in a multi-rack with Hadoop installed in every node deployed in the cloud platform. The proposed strategy reduces the query execution time, significantly improves the data locality and CPU utilisation, and is proved to be more efficient for massive dataset processing in a heterogeneous distributed environment. In addition, MCL shows a marginal improved performance over HAC and k-means for queries exhibiting interest localities.

Key words: BigData, Storage and Compute Infrastructure, Interest Locality, Data Placement, Hierarchical Agglomerative Clustering, Markov Clustering, Heterogeneous Hadoop Cluster, Cloud

AMS subject classifications. 68-M14, 68-M20, 68-W10, 68-W15

1. Introduction.

In the current data era, massive volumes of data are being generated every second in a variety of domains such as geosciences, the social web, finance, e-commerce, healthcare, climate modelling, physics, astronomy, government sectors etc. BigData is the term applied to such large volumes of datasets whose size is beyond the ability of the commonly used software tools to capture, manage, and process within a tolerable elapsed time [1, 2]. By virtue of their parallel and distributed computing capability, Hadoop and Spark [3, 4, 5] are considered ideal solutions to analyse and gain insights from BigData and are well-recognised as de facto BigData processing platforms in the cloud; they have been adopted extensively and are currently used widely in many application domains. Apache Hadoop [1, 6] facilitates the distributed processing of large datasets across clusters of commodity hardware using simple programming models. Here, local storage and computation are achieved through the two major components namely Hadoop Distributed File System *(HDFS)* and MapReduce *(MR)*. The fundamental concept of HDFS [7] and MR [8] is to distribute data among nodes and process them in parallel. HDFS is a distributed file system capable of storing large files across multiple nodes. It follows a master-slave architecture, consisting of one NameNode and multiple DataNodes. When a file is dumped into HDFS, it is broken into fixed-size blocks and stored on multiple DataNodes. The DataNodes periodically report the blocks stored in them to the NameNode, thereby updating the metadata. When a query is executed from a client, it will reach out to the NameNode to retrieve the metadata, and then reach out to the DataNodes to retrieve the data blocks.

The major challenge in processing BigData in HDFS is faced during query execution, since the time taken

*The research work reported in this paper is supported by Department of Electronics & Information Technology (DeitY), a division of Ministry of Communications and IT, Government of India.

† Department of Computer Applications, National Institute of Technology, Tiruchirappalli 620015, India. (meetvengadesh@gmail.com). Thanks to Microsoft Azure for sponsoring the cloud infrastructure required to carry out the experiments under the Microsoft Azure for Research Award.

‡ Department of Computer Applications, National Institute of Technology, Tiruchirappalli 620015, India. (blsundar@nitt.edu).

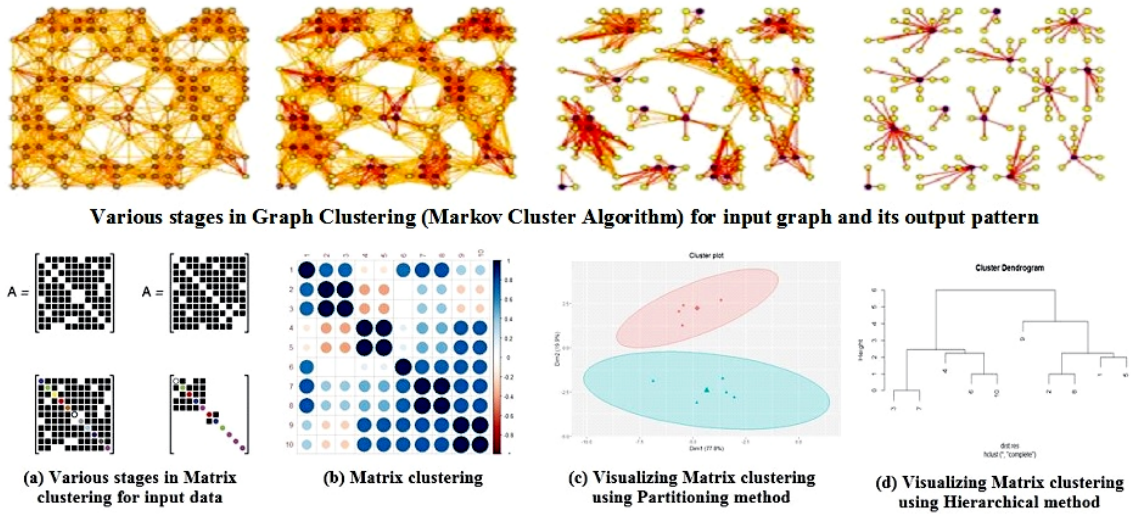

FIG. 1.1. *Various stages of clustered graph by applying clustering algorithm*

to execute a query and return the results increases exponentially as the amount of data increases, leading to a long waiting time for the user [9]. Sometimes, the waiting times could range from minutes, to hours, to days in the worst cases. During query execution, it is commonly observed that most of the data-intensive applications exhibit interest locality [10]. It may be different for different domain analysts based on geographical location, time, person etc. (i.e. domain scientists are only interested in a subset of the whole dataset, and are likely to access one subset more frequently than others. For example, in the bioinformatics domain, X and Y chromosomes are related to the offsprings gender. Both chromosomes are often analysed together in generic research rather than all 24 human chromosomes). Mostly, for query execution, only a part of such BigData sets is utilised. The detailed analysis of various query executions clearly shows a significant similarity in the data required to execute the query during a set of time intervals. These data blocks will then have the highest frequency of being accessed as a group during executions. Data grouping is then formally defined as grouping semantics to represent the possibility of two or more data being accessed as a group. In Hadoops Default Data Placement Strategy *(HDDPS)*, the data blocks are placed randomly across the cluster of nodes without considering the nature of queries likely to be executed in the system. Due to such non- consideration of interest locality, it is possible for the required data blocks to be concentrated within fewer computing nodes, which, in turn, results in an increase in query execution time, query latency etc. In this paper, an Optimal Data Placement *(ODP)* Strategy based on grouping semantics is proposed. The natural behavioural groupings in the dataset are identified by applying clustering algorithms and the data-placement decision is taken based on the observed grouping behaviour. Clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups [11, 12].

In this paper, we experiment the significance of different clustering techniques viz. k-means [13], Hierarchical Agglomerative Clustering *(HAC)* [14] and Markov Clustering *(MCL)* [15] in grouping-aware data placement for data-intensive applications with interest locality. It has been proved in a heterogeneous distributed environment for the e-commerce dataset [16, 17]. The results show that queries are solved by the domain analyst at the earliest possible time to enable quick decisions, as well as deriving maximum utilisation of resources. Fig.1.1 shows the various stages in MCL for an input dataset. Fig.1.1(a) shows the various stages in HAC for an input dataset. The clustered matrix obtained by applying the HAC is shown in Fig.1.1(b). The visualisation of matrix clustering by k-means and HAC methods is depicted in Fig.1.1(c) and Fig.1.1(d) respectively.

2. Related Works. Several works were carried out in data placement for massive datasets in some specific ways to support high-performance data accesses. ODP strategy, which focuses on reducing energy consumption and resource utilisation, was proposed by Ashwin Kumar et al. (2013) [18] and Wu et al. (2017) [19]. They

proposed ODP by locating the related data blocks together. However, the major drawback in this area of focus is the increased query execution time. Here, the focus is on reducing the utilisation of resources, but this cannot be considered as a viable solution, since the real objective of processing BigData is achieving timely results.

Some significant works have also been carried out on data placement to achieve a reduced query execution time. Lee et al. (2014)(2014) [20] proposed an ODP by taking into account the computing capacity of a data node so that faster computing nodes are allocated with more data. This reduces the overall query execution time and provides high throughput of data. However there is no mechanism to ensure that the data blocks which are required for execution are proportionately present in those nodes since the grouping semantics of the dataset is not taken into account. Xiong et.al (2015) [21] proposes a heterogeneity aware data placement algorithm which initially groups the Data-Nodes as several virtual storage tiers (VST). The data blocks are placed across the nodes in each VST circuitously according to the hotness of data. This strategy shows an improved MR performance with reduced disk space utilization. However the individual requirements of data blocks are only assessed for measuring the hotness. But the relative dependency among various blocks for the different task executionsis not considered, which may lead to concentration of popular data within a node leading to reduced parallel execution.

ODP to reduce query execution time based on grouping semantics by applying clustering algorithms is also discussed by few researchers. Wang et al. (2014) [10] and Wu, w et al. (2016) [22] proposed an ODP algorithm based on grouping semantics, which reduces the query execution time and improves the data locality. It improves the parallel execution of datasets with interest locality. This ODP strategy use the Bond energy algorithm (BEA) to cluster the dependency matrix, which leads to a higher execution time. This is due to the time complexity in BEA for finding the permutations of all rows. In addition, for further execution of any new task, all iterations of BEA must be repeated.

Liao et al. (2016) [23] focus on optimising resource utilisation using a novel scheduling algorithm. Similarly, Shivaswamy et al.(2017) [24] suggest scheduling the work flow of jobs during concurrent executions for optimal resource utilisation. However, in both cases, the existence of a general behaviour pattern among the tasks executed during a period of time is not considered. Hence, these queries with interest locality require further consideration. Some studies elucidate that some significant clustering techniques [13, 14, 15] are available that can be applied to find the natural groupings in a dataset with reduced computations without compromising the clustering performance. We harness these clustering approaches in large-scale data management to achieve improved performance in terms of reduced execution time, through ODP, especially when data-intensive applications exhibit interest locality.

3. CORE-Optimal Data Placement Strategy. An ODP strategy based on grouping semantics is proposed in this paper. The entire workflow diagram is shown in Fig. 3.1. The different steps involved in the proposed strategy are detailed below.

Step 1: ***Analysing User History Log*** The meta-information and user history log will be the input for this step. Analysing the characteristics of the cluster from the user history log for various workloads is the key for making an optimal placement decisions. These log files are voluminous and varied (semi-structured). All MapReduce applications executed in the cluster save the task execution details as a log file, which consists of two files *(i) the Job Configuration file* and *(ii) the Job Status file* - for each job executed in the machine.

Step 2: ***Tracing Network Topology*** NameNode contains meta-data from which the network topology is constructed to identify the different DataNodes present in the cluster and the data blocks present in each DataNode.

Step 3: ***Building Task Frequency Table*** Using these logs as input, the task frequency table is constructed, which contains different tasks, the frequency of each task, and the blocks required for each task.

Step 4: ***Constructing Task Execution Graph*** The computations of parallel processing can be solved efficiently, only if the task executions and the blocks required are depicted as a graph. The task execution graph shown in Fig.3.2 is obtained by analysing the task frequency table using the iGraph network analysis tool [25]. The task execution graph is an unordered pair $GTex = (B, T)$, where B represents a set of vertices as blocks and T represents a set of edges as tasks executed. $GTex$ is undirected and may hold parallel edges since some sets of blocks ($B' \subseteq B$) may be required for different task executions Ti.

Step 4a: ***Clustered Task Execution Graph (CGTex)*** The task execution graph ($GTex$) is then

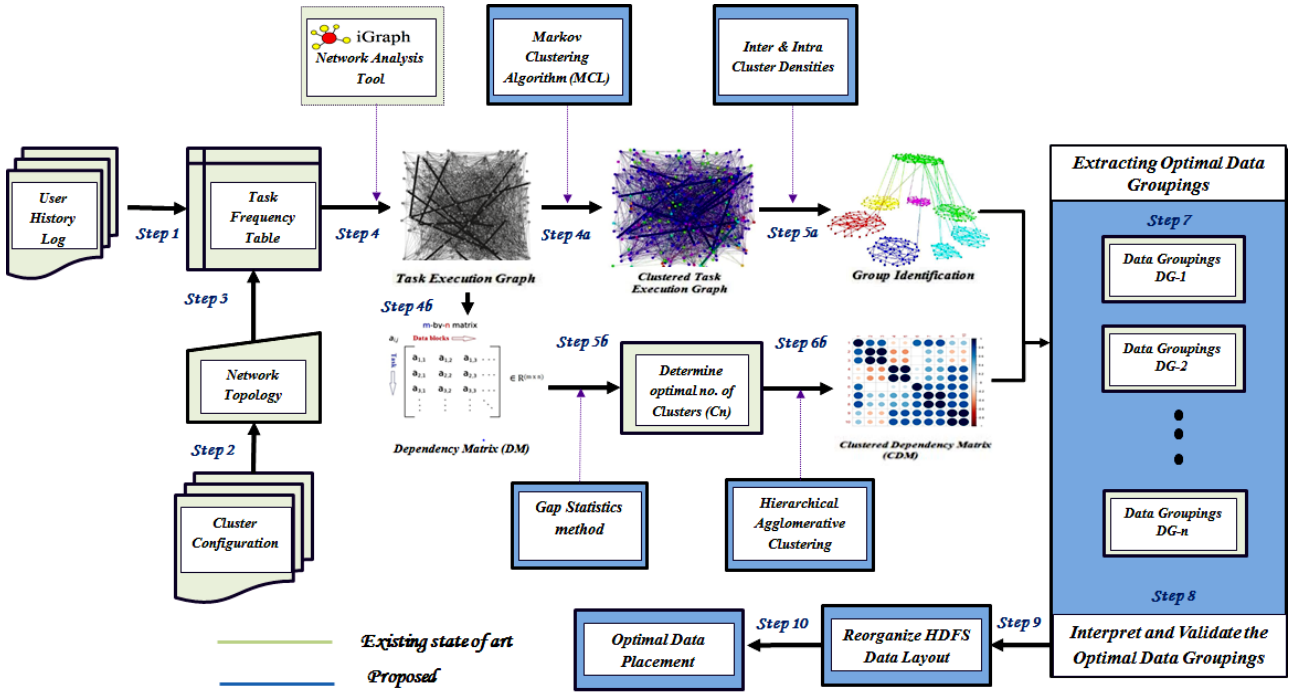

FIG. 3.1. *Workflow diagram for the proposed work*

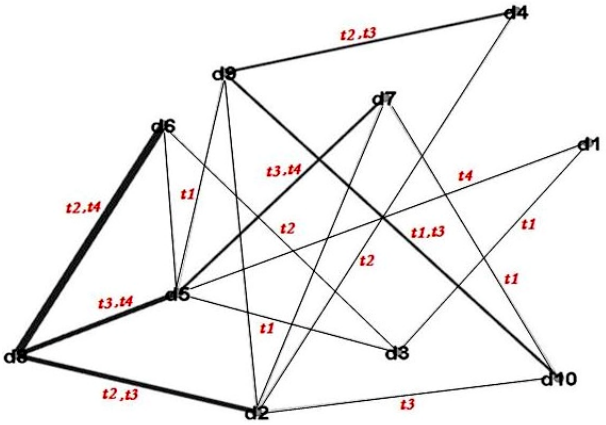

FIG. 3.2. *Task execution graph for sample graph consisting of 10 blocks and 4 tasks*

converted into a clustered task execution graph ($CGTex$) by applying the graph clustering algorithm [15]. The normal representation of the graph may not reveal any natural cluster characteristics. When a uniformly distributed graph is applied with a clustering algorithm, the graph will be arbitrarily grouped into clusters based on the similarity metric. To identify the natural groupings in the graph, MCL algorithm, fast, scalable, and unsupervised algorithm, is applied over the $GTex$ and the various stages of the clustered graph obtained are shown in Fig.3.3.

Step 5a: ***Group Identification*** The clusters obtained from the clustered task execution graph ($CGTex$) are separated into various groups. A subset of vertices can be said to form a good cluster if sub-graphs are dense with more connections within the group and only a very few connections exist from the group to the rest of graph. Accordingly, each group in the cluster will have individual characteristics showing high intra-cluster

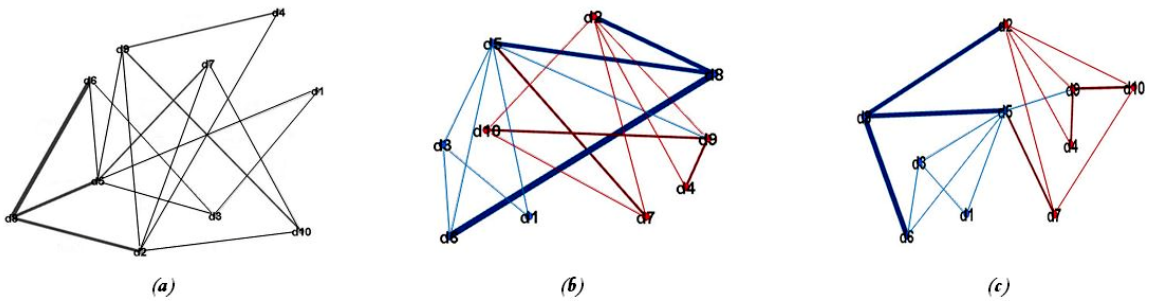

FIG. 3.3. *Various stages of clustered graph by applying MCL algorithm*

and low inter-cluster density (refer eqns 3.1 and 3.2). Based on the grouping behaviour, the associated clusters are grouped together by applying MCL.

$$Intra\ cluster\ density\ \delta_{int}(c) = \frac{|\{\{v,u\}|v \in C, u \in C\}|}{|C|(|C|-1)} \tag{3.1}$$

$$Inter\ cluster\ density\ \delta_{int}(G|C_1,....,C_k) = \frac{1}{k}\sum_{i=1}^{k}\delta_{int}(c_i) \tag{3.2}$$

Step 4b: ***Constructing Dependency Matrix (DM)*** From the task execution graph ($GTex$) and the information available from the task frequency table, the dependency matrix (DM) is constructed. DM is a symmetric matrix of order nxn, where n is the number of blocks present in the cluster. DM exhibits the degree of dependency between various blocks during simultaneous execution of tasks. The diagonal elements of the DM represent the number of tasks for which the corresponding block is required. Any other element in DM_{ij} will show the number of tasks for which one block b_i will be accessed along with the block b_j for execution.

Step 5b: ***Determining optimal no. of clusters - Gap Statistics (GS)*** The gap statistic method can be used to calculate the optimal number of clusters for the given dataset. The gap statistic compares the total within intra-cluster variation (w_k, wk) for different values of k with their expected values under null reference distribution of the data. The gap statistic for a given k is defined as follows:

$$Gap_n(k) = E_n^*\{log(W_k)\} - log(w_k) \tag{3.3}$$

$$Gapn(k) = En * log(Wk) - log(Wk) \tag{3.4}$$

The standard deviation (sd_k sdk) of $log(W_k^*)log(Wk_*)$ is also computed in order to define the standard error (S_k sk) of the simulation as follows.

$$s_k = sd_k * \sqrt{1 + \frac{1}{B}} \tag{3.5}$$

$$sk = sdk * 1 + 1/B \tag{3.6}$$

Finally, a more robust approach is to choose the optimal number of clusters K as the smaller k, such that:

$$Gap(k) \geq Gap(k+1) - s_{k+1} \tag{3.7}$$

The smallest value of k is chosen so that the gap statistics is within one standard deviation of the gap at $k+1$. Based on this, we can calculate the optimal number of clusters for a given dataset.

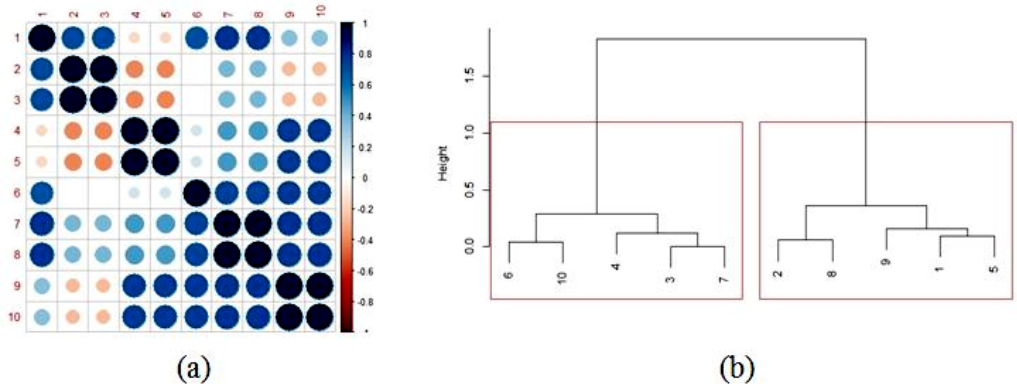

FIG. 3.4. *(a) Clustered correlation matrix and (b) Dendogram for hierarchical clusters*

Step 6b: ***Clustered Dependency Matrix (CDM)*** The dependency matrix (DM) is then converted into a clustered dependency matrix (CDM) by applying the matrix clustering algorithm. In this paper, HAC is used to cluster the matrix into groups. The application of HAC technique is examined for the proposed work and is explained below. In the HAC method, a hierarchy of clusters is formed to identify the natural groupings in the dataset. A bottom-up approach is used in this algorithm. Initially, HAC considers each data block as a single entity, then it combines the blocks with most similar blocks to form a bigger cluster. The iteration is then repeated, with further merging of the clusters with the output obtained earlier. Once all the blocks are merged into the required number of clusters derived from the gap statistics in step 5b, the algorithm ends.

The clusters obtained are merged based on similarity/dissimilarity measures. The Euclidean distance is used to measure the distance between each pair of data blocks (d_i, d_j) from the dataset D ($(d_i, d_j) \subseteq D$).

$$Dist_{eq}(d_i, d_j) = \sqrt{(x_{d_i} - x_{d_j})^2 + (y_{d_i} - y_{d_j})^2} \tag{3.8}$$

To merge two blocks in a cluster, linkage methods can be used to decide the neighbouring pair of blocks to be merged. In this paper, a single linkage method is adopted. It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2, and considers the smallest of these dissimilarities as a linkage criterion.

$X_1, X_2, ..., X_k$ = Observations from Cluster1,
$Y_1, Y_2, ..., Y_k$ = Observations from Cluster2,
$d(x, y)$ = Distance between observation vector X with Y.

$$SingleLinkage \ : \ d_{12} = min_{i,j} \ d(X_i, Y_j) \tag{3.9}$$

The reason for the use of HAC is due to its flexibility, versatility and, mostly, its lower computational complexity. The HAC algorithm clusters the highly associated data together based on the grouping behaviour and generates data groupings as shown in Fig.3.4. Initially, each data is considered as a cluster. Computing the distances between all pairs of data blocks takes $O(m^2)$ computation. Then, the data is sorted to find the smallest, which takes $O(m^2\ log\ m)$ time. The closest pair are then merged and all the distance pairs are again recomputed with the new cluster, which takes $O(m\ log\ m)$. The iteration process continues $(m-1)$ times until all the data merges to form a single cluster, which takes $(m-1)*O(m\ log\ m)$. Hence, the computational complexity for HAC to find the natural groupings of data blocks in the dataset takes $O(m^2\ log\ m)+(m-1)*O(m\ log\ m) = O(m^2\ log\ m)$.

Step 7: ***Extracting ODG*** Then, both the HAC and MCL algorithms with the optimal number of clusters are independently applied over the history log. The resulting output of each method will be a unique set of data groupings. It is confirmed that each grouping obtained is conceptually distinguishable by validating and interpreting each obtained group.

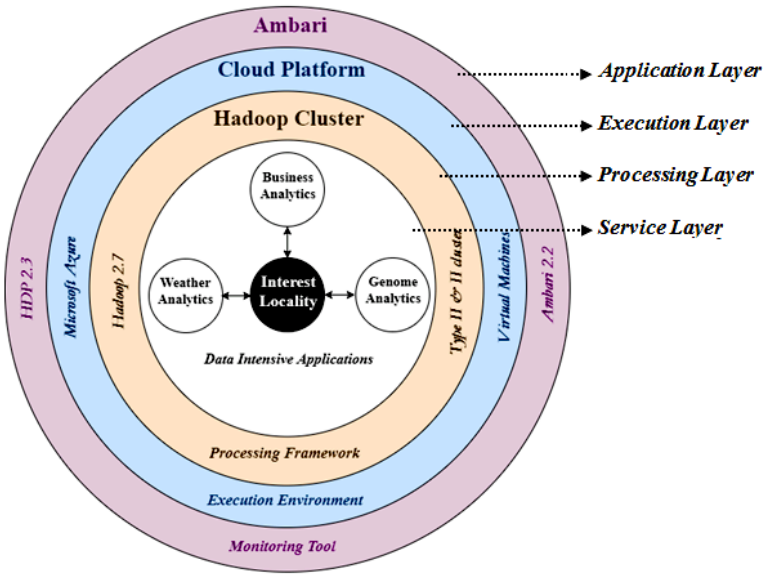

FIG. 4.1. *Schematic diagram for execution framework*

Step 8: ***Interpreting and validating ODG*** Then, the extracted data groupings must be interpreted and analysed to find how well the obtained groupings fit the data without reference to external attributes. Then, the data groupings obtained from the two different sets of cluster analysis are compared to determine the optimal data groupings using the silhouette method. We can separately execute and test the validated groupings for local map tasks in distributed settings.

Step 9: ***Reorganising HDFS data layout*** The implementation of our proposed strategy will dynamically reorganise the HDFS data layout in order to achieve an optimal data placement for improved execution; this program for proposed work is launched as a utility to be executed manually as and when required. The execution of this utility modifies the machine instruction, which is a triplet $< B_{id}, SN, DN >$, where B_id is the Block ID, SN is the Source Node, and DN is the Destination Node. If SN and DN are different, then the reorganisation has been carried out considering the rack topology and the load balancer.

Step10: ***Achieving optimal data layout*** After reorganisation of the default data layouts in HDFS, our proposed work achieved an optimal data layout that ensures maximum parallel execution per group. It does not guarantee 100% local map task execution every time, but it will always produce an improved result over the naive data placement strategy, which is tested with the production cluster (explained in detail in the subsequent section).

4. Experimental Results and Analysis. The experiments were tested in a cloud platform, since the cloud is emerging as a preferred paradigm to deploy highly available and scalable systems for the processing of BigData [27]. It is also a reliable, fault-tolerant, flexible, and low-cost environment. Microsoft Azure provides a platform to collect, store, process, analyse, and visualise BigData in the cloud.

In order to carry out the experiments, 10 node heterogeneous clusters, deployed in a multi-rack environment, with every node having Hadoop, were established in the Azure cloud. The cluster was configured with one Master (NameNode) and nine Slaves (DataNodes). In order to have a heterogeneous environment, the DataNodes were chosen with varied configurations. Table 4.1 and Table 4.2 depicts the detailed cluster configuration, file system configuration respectively. The clusters were provisioned, managed, and monitored using Apache Ambari. The schematic diagram for the execution framework is shown in Fig. 4.1.

To evaluate the performance of MR, we experimented with an Amazon product review dataset [28] consisting of product reviews from Amazon, spanning approximately 18 years (1996-2014). This dataset covers reviews of multiple products such as Books, Baby products, Electronics, Kindle store, Movies and TV, Health and

TABLE 4.1
Cluster configuration

Property	**NameNode- 1**	**DataNode- 9**		
	NN- 1	DN- 2	DN- 3	DN- 4
Instance Type	DS5 v2	DS4 v2	DS3 v2	DS2 v2
vCPU	16	8	4	2
RAM	56 GB	28 GB	14 GB	8 GB
Processor	Intel Xeon E5-2673@2.4 GHz			
OS	CentOS 7.3			
Hadoop Version	Hadoop 2.7.2 (Stable Version)			

personal care etc. This dataset (size 19.5GB) is freely available to download from Stanford Network Analysis Project (SNAP). Each of the reviews will contain the following information (ProductID, Title, UserID, Price, Helpfulness, ProfileName, Score, Time, Summary).

TABLE 4.2
Data, distributed file system and cluster - configuration parameters

HDFS Status : Healthy	
Total Size	19651541778 B
Total files	5
Average block size	66390343 B
Total blocks (validated)	296
Default replication factor	1
Number of DataNodes	9
Number of racks	3

TABLE 4.3
Data relating to interest domain

Name	**Size**	**Block Size**	**No. of Records**
Reviews_Baby.json	580.22 MB	64 MB	915446
Reviews_Books.json	13.74 GB	64 MB	16302134
Reviews_Electronics.json	1.38 GB	64 MB	1689188
Reviews_Kindle_Store.json	789.46 MB	64 MB	982619
Reviews_Movies_and_TV.json	1.85 GB	64 MB	1697533

During the execution of interest-based queries, it is observed that there is a severe drag in MR performance. In the business forecasting domain [16, 17] in particular, to predict future product demand/sales of particular products, the reviews in respective categories alone need to be analysed rather than sweeping through the reviews in all categories. The data relating to the interest domain in the Amazon review data is shown in Table.4.3. When this data relating to the interest domain is uploaded in HDFS, the data splits into even-sized data blocks and distributed randomly across the DataNodesThe data are placed without any consideration of the nature of the queries likely to be executed. Due to this, it is possible that dependent blocks required for execution will be concentrated within fewer computing nodes, resulting in several lacunas such as underutilisation of resources and increased query execution time.

To prove the significance of clustering in data placement, several experiments were conducted by executing various interest-based queries (Join and Aggregate) related to business analytics (e-commerce dataset). The tasks were chosen in such a way that they had specific dependent blocks and were executable only within a subset of the whole dataset. Application benchmark performance was also executed for the evaluation, e.g. different tasks related to prediction modelling (regression) for different products was executed for evaluation.

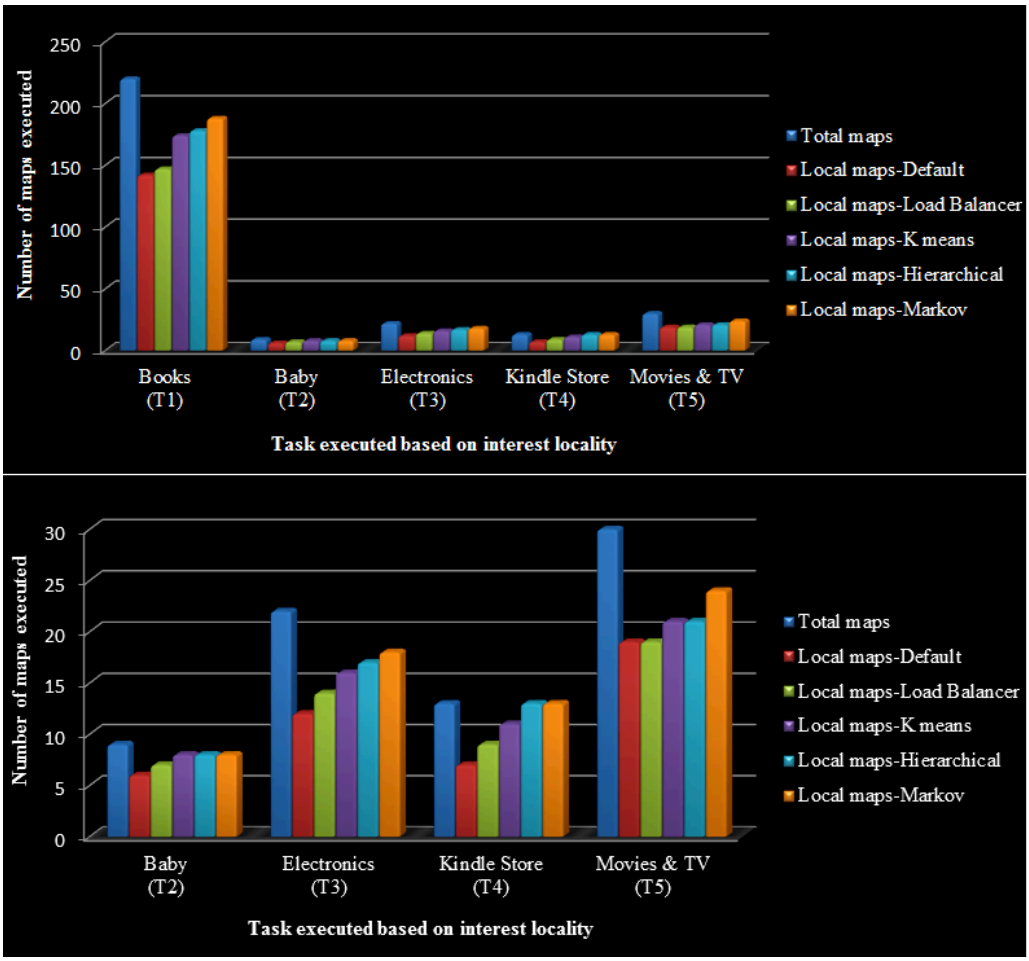

FIG. 4.2. *Graphs showing the performance improvement in local map task execution*

Other join and aggregate queries were also taken into consideration, e.g. finding an electronic product with a higher rating during a specific period, finding a book that has been reviewed more, finding the usefulness of a Kindle product during 2006 to 2007etc.

The join and aggregate queries on e-commerce were written using PIG scripts and executed in a TEZ execution engine. The prediction modelling for business analytics was written using Mahout, a scalable machine learning library, and executed in the MR execution engine. The output metrics were collected using Ambari monitoring tool, deployed in the HDP platform. These applications were executed in real time and the performance was compared with existing data placements such as HDDPS, load balancer, and proposed data placement with different clustering algorithms (k-means, HAC, MCL). The output presented in Table 4.4 shows an interesting result, with improved local map task and reduced execution time. Fig. 4.2 and 4.3 depict the graphical representations.

From Table 4.5, with a maximum of 296 maps required for execution, HDDPS has 186 data local maps (i.e. 62.8%), whereas as MCL has 251 local maps (i.e. 84.7%), showing an improvement of 34.9% ((251-186)/186) of local map executions. Similarly, the execution time was also decreased from 18,879 secs to 13,762 secs, thereby showing an overall improvement of 27.1% ((18879-13762)/18879). In addition, the data placement based on MCL shows an improved performance (5.9% in data locality, 4.3% in execution time) over HAC and an improved performance (9.1% in data locality, 12.5% in execution time) over data placement based on k-means for queries exhibiting interest localities.

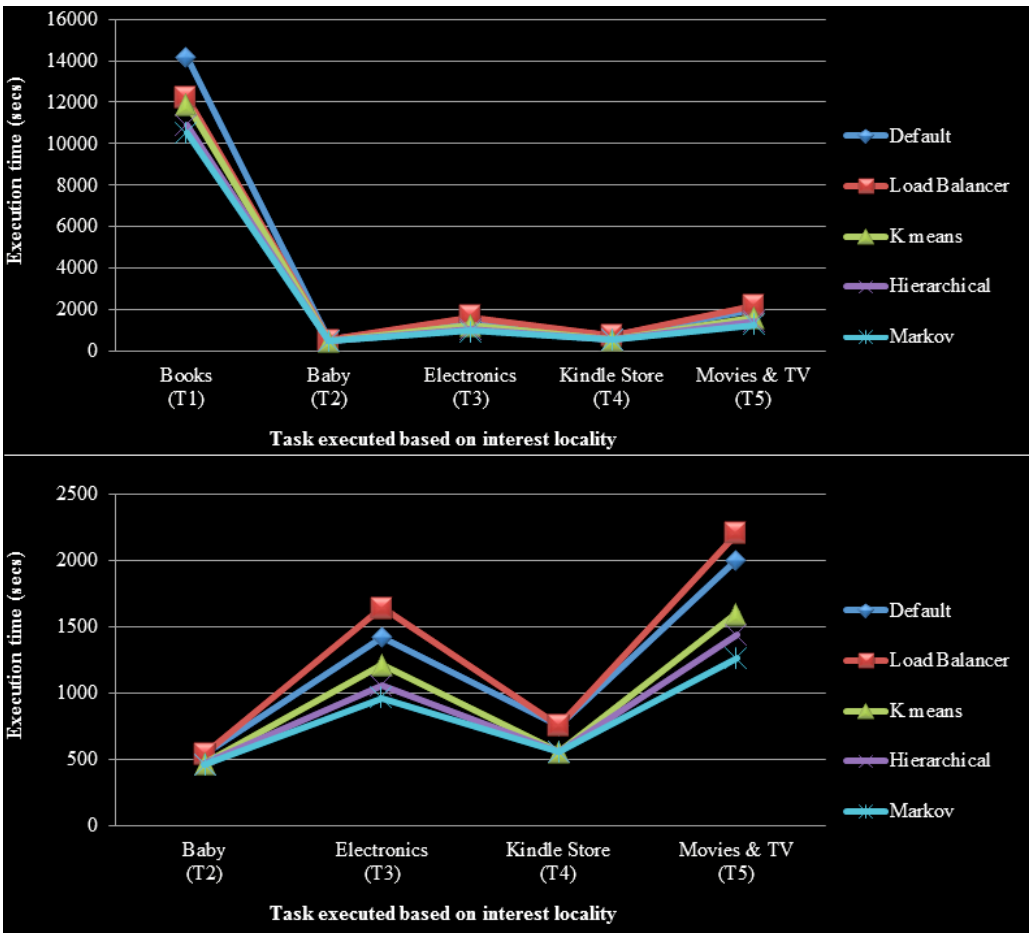

FIG. 4.3. *Graphs showing the performance improvement in execution time*

Data placement based on Markov clustering shows improved performance, especially when data-intensive applications have interest locality. This is because the natural clusters obtained through MCL exhibit higher utilisation of resources with less complexity. The reduced execution time is due to the significant improvement achieved in CPU utilisation through Markov clustering. The CPU utilisation of each node and every node in the cluster is improved. Also, the average CPU utilisation of the cluster increased from 55.2% to 81.3%, showing an improvement of 26.1%, as depicted in Fig.4.4. When tested in the worst case, where any interest locality does not exist, i.e. all data blocks are required to be accessed for execution, the proposed strategy shows the same efficiency as default.

5. Conclusion and Future Work. Optimal Data Placement (ODP) Strategy based on grouping semantics is proposed in this paper. The significance of different clustering techniques viz. k-means, Hierarchical Agglomerative Clustering (HAC) and Markov Clustering (MCL) in grouping-aware data placement for data-intensive applications with interest locality has been tested. The experiments were carried out in a 10-node cluster placed in a multi-rack environment deployed in the Azure cloud. The results conclude that the MCL-based data placement strategy improves the local map execution by 34.5% and reduces the execution time by 27.8% compared to Hadoops Default Data Placement Strategy (HDDPS). In addition, it can be inferred that the MCL-based data placement strategy shows an improved performance (5.9% in local map execution, 4.3% in execution time) over HAC (9.1% in local map execution, 12.5% in execution time) and over data placement based on k-means for queries exhibiting interest localities. The results strengthen the proposed work and prove to be more efficient for massive datasets processing in a distributed environment.

TABLE 4.4
Performance improvement in local maps and execution time - for various interest domains

Amazon Review Product Dataset - SNAP (20 GB)						
Data Placement	**Interest Domain**	**Books (T1)**	**Baby (T2)**	**Electronic (T3)**	**Kindle (T4)**	**Movies (T5)**
	Total maps (nos)	220	9	22	13	30
Default	*Local maps (%)*	64.5	66.6	54.5	53.8	63.3
	Exe. time (secs)	14168	**533**	**1422**	**753**	**2003**
Load Balancer	*Local maps (%)*	66.8	77.7	63.6	69.2	63.3
	Exe. time (secs)	12224	533	1640	753	2202
K-means	*Local maps (%)*	79.0	88.8	72.2	84.6	70.0
	Exe. time (secs)	11882	472	1216	557	1602
Hierarchical	*Local maps (%)*	80.9	88.8	77.2	100	70
	Exe. time (secs)	10877	464	1057	557	1440
Markov	*Local maps (%)*	85.4	88.8	81.8	100	83.3
	Exe. time (secs)	10520	**457**	**963**	**557**	**1265**

TABLE 4.5
Overall comparison of proposed strategy with existing data placement policies

	Total maps (nos)	**Local maps (nos)**	**Local maps (%)**	**Exe. time (secs)**
Default	*296*	*186*	*62.8*	*18879*
Load Balancer		*196*	*66.2*	*17352*
k-means		*230*	*77.7*	*15729*
Hierarchical		*237*	*80.0*	*14395*
Markov		*251*	*84.7*	*13762*

Even though the results are very optimistic, there is still scope for improvement, since the layout obtained in the proposed work considers only the horizontal relationships among the data. Hence, an Optimal Data Placement (ODP) Algorithm considering inter-relationships (vertical) among the blocks can be proposed in the additional data groupings obtained, which could further improve the performance during the execution of simultaneous map tasks.

Acknowledgments. This work is supported under Visvesvaraya PhD scheme funded by Ministry of Electronics and Information Technology, Government of India. We also like to thank Microsoft Azure for having sponsored the cloud Infrastructure required for carrying out the experiments under Microsoft Research Award.

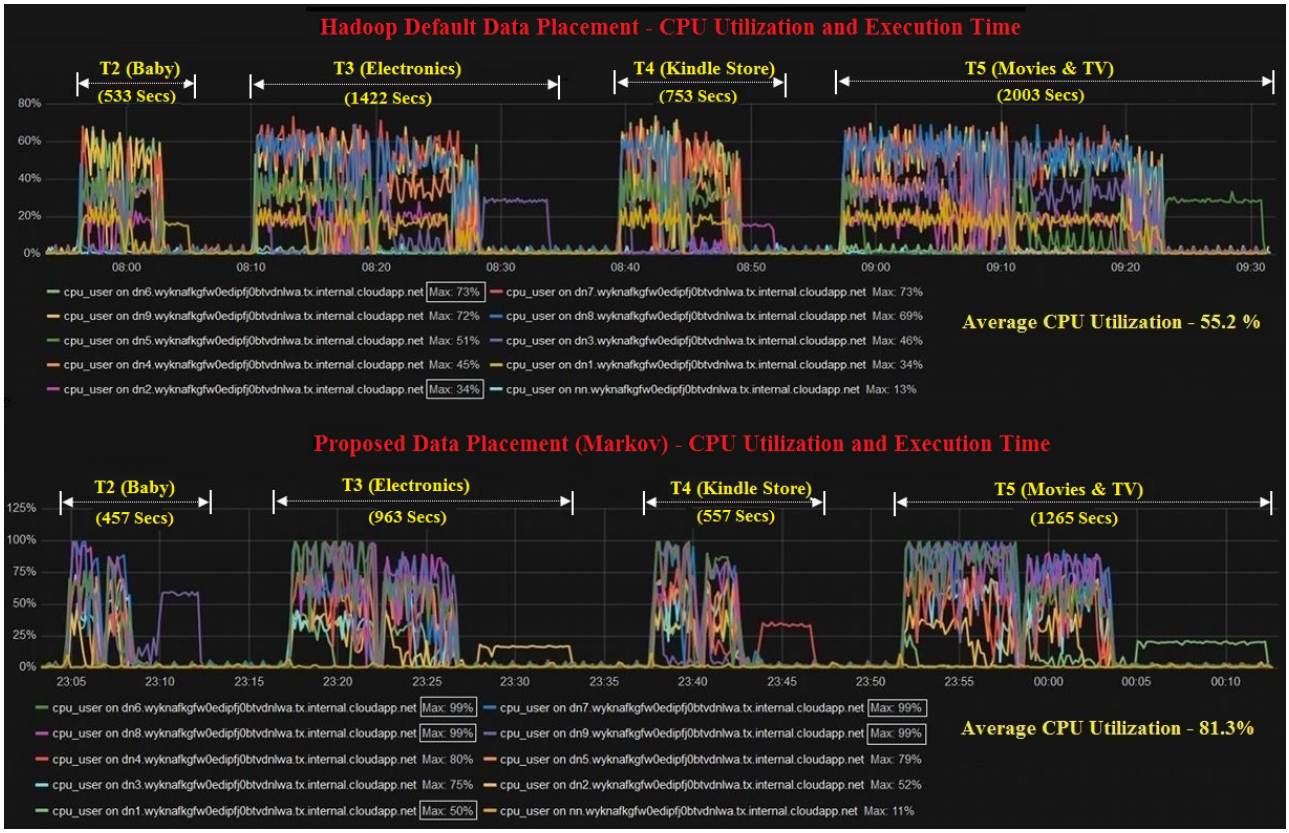

FIG. 4.4. *Graphs showing the performance improvement in CPU utilization of each DataNode*

REFERENCES

[1] WHITE, T.: Hadoop: The definitive guide. OReilly Media, Inc., 2012

[2] HU, H., WEN, Y., CHUA, T. S., & LI, X.: Toward scalable systems for big data analytics: A technology tutorial, IEEE access, 2014, Vol.2, pp.652-687

[3] SAMMER, E.: Hadoop operations. O'Reilly Media, 2012

[4] ZHANG, Y., REN, J., LIU, J., XU, C., GUO, H., & LIU, Y.: A survey on emerging computing paradigms for big data. Chinese Journal of Electronics, 2017, Vol.26(1), pp.1-12

[5] SHIRAHATA, K, & MATSUOKA, S.: Big Data processing using Apache Spark and Hadoop. Big Data and Software Defined Networks, IET, 2018, Chap.6, pp. 115-138

[6] LITKE, W., & BUDKA, M.: Scaling Beyond One Rack and Sizing of Hadoop Platform. Scalable Computing: Practice and Experience, 2015, Vol.16 (4), pp.423-436

[7] SHVACHKO, K., KUANG, H., RADIA, S., & CHANSLER, R.: The hadoop distributed file system. Proceedings of the 26th Symposium on Mass Storage Systems and Technologies (MSST), IEEE, May 2010, pp.1-10

[8] DEAN, J., & GHEMAWAT, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM, 2008, Vol.51 (1), pp.107-113

[9] CHEN, C. P., & ZHANG, C. Y.: Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences, 2014, Vol.275, pp.314-347

[10] WANG J, XIAO Q, YIN J, & SHANG P.: 'Draw: A new data-grouping-aware data placement scheme for data intensive applications with interest locality', IEEE Transactions on Magnetics, 2013, Vol.49 (6), pp.25142520

[11] AGGARWAL, C. C., & REDDY, C. K.: Data clustering: algorithms and applications. Chapman and Hall/CRC, 2013

[12] FAHAD, A., ALSHATRI, N., TARI, Z., ALAMRI, A., KHALIL, I., ZOMAYA, A.Y., & BOURAS, A.: A survey of clustering algorithms for big data:Taxonomy and empirical analysis. IEEE transactions on emerging topics in computing, 2014, Vol.2 (3), pp.267-279

[13] KHEDR, A. M., & BHATNAGAR, R. K.: New Algorithm for Clustering Distributed Data Using k-Means. Computing & Informatics, 2014, Vol.33 (4), pp.943-964

[14] BOUGUETTAYA, A., YU, Q., LIU, X., ZHOU, X., & SONG, A.: Efficient agglomerative hierarchical clustering. Expert Systems with Applications, 2015, Vol.42 (5), pp.2785-2797

[15] Dongen, S.: Performance criteria for graph clustering and Markov cluster experiments, 2000
[16] Shao, F., & Yao, J.: The Establishment of Data Analysis Model about E-Commerces Behavior Based on Hadoop Platform. Proceedings of the International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), IEEE, January 2018, pp. 436-439
[17] Suguna, S., Vithya, M., & Eunaicy, J. C.: Big data analysis in e-commerce system using HadoopMapReduce. Proceedings of the International Conference on Inventive Computation Technologies (ICICT), IEEE, August 2016, Vol. 2, pp. 1-6
[18] Kumar, K. A., Deshpande, A., & Khuller, S.: Data placement and replica selection for improving co-location in distributed environments. arXiv preprint arXiv:1302.4168, 2013
[19] Wu, W., Lin, W., Hsu, C. H., & He, L.: Energy-efficient hadoop for big data analytics and computing: A systematic research insights. Future Generation Computer Systems, 2017
[20] Lee, C. W., Hsieh, K. Y., Hsieh, S. Y., & Hsiao, H. C.: A dynamic data placement strategy for hadoop in heterogeneous environments, Big Data Research, 2014, Vol.1, pp.14-22
[21] Xiong, R., Luo, J., & Dong, F.: Optimizing data placement in heterogeneous Hadoop clusters, Cluster Computing, 2015, Vol.18 (4), pp.1465-1480
[22] Wu, J. X., Zhang, C. S., Zhang, B., & Wang, P.: A new data-grouping-aware dynamic data placement method that take into account jobs execute frequency for Hadoop. Microprocessors and Microsystems, 2016, Vol.47, pp.161-169
[23] Liao, J., Zhang, L., Li, T., Wang, J., & Qi, Q.: Efficient and fair scheduler of multiple resources for MapReduce system. IET Software, 2016, Vol.10 (6), pp.182-188
[24] Rashmi, S., & Basu, A.: Resource optimised workflow scheduling in Hadoop using stochastic hill climbing technique. IET Software, 2017, Vol.11 (5), pp.239-244
[25] Csardi, G., & Nepusz, T.: The igraph software package for complex network research. InterJournal, Complex Systems, 2006, Vol.1695 (5), pp.1-9
[26] Tibshirani, R., Walther, G., & Hastie, T.: Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society, 2001, Vol.63 (2), pp.411-423
[27] Manogaran, G., & Lopez, D.: Big Data in cloud data centers. Big Data and Software Defined Networks, IET, 2018, pp.159-182
[28] McAuley, J., Pandey, R., & Leskovec, J.: Inferring networks of substitutable and complementary products, Proceedings of the International conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 785-794

Edited by: Sasko Ristov
Received: Mar 1, 2018
Accepted: Sep 3, 2018