Received May 25, 2019, accepted June 27, 2019, date of publication July 15, 2019, date of current version July 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927076

A Comprehensive Survey of Load Balancing Strategies Using Hadoop Queue Scheduling and Virtual Machine Migration

NILADRI SEKHAR DEY[1,2] **AND T. GUNASEKHAR**[2]
[1]Department of Information Technology, B. V. Raju Institute of Technology, Hyderabad 500082, India
[2]Department of Computer Science and Engineering, Koneru Lakshmaiah Educational Foundation, Guntur 522502, India

Corresponding author: Niladri Sekhar Dey (sd.niladri@gmail.com)

ABSTRACT The recent growth in the demand for scalable applications from the consumers of the services has motivated the application development community to build and deploy the applications on cloud in the form of services. The deployed applications have significant dependency on the infrastructure available with the application providers. Bounded by the limitations of available resource pools on-premises, many application development companies have migrated the applications to third party cloud environments called data centers. The data center owners or the cloud service providers are entitled to ensure high performance and high availability of the applications and at the same time the desired scalability for the applications. Also, the cloud service providers are also challenging in terms of cost reduction and energy consumption reductions for better manageability of the data center without degrading the performance of the deployed applications. It is to be noted that the performance of the application does not only depend on the responsiveness of the applications rather also must be measured in terms of service level agreements. The violation of the service level agreements or SLA can easily disprove the purpose of application deployments on cloud-based data centers. Thus, the data center owners apply multiple load balancing strategies for maintaining the desired outcomes from the application owners at the minimized cost of data center maintainability. Hence, the demand of the research is to thoroughly study and identify the scopes for improvements in the parallel research outcomes. As the number of applications ranging from small data-centric applications coming with the demand of frequent updates with higher computational capabilities to the big data-centric application as big data analytics applications coming with efficient algorithms for data and computation load managements, the data center owners are forced to think for efficient algorithms for load managements. The algorithms presented by various research attempts have engrossed on application specific demands for load balancing using virtual machine migrations and the solution as the proposed algorithms have become application problem specific. Henceforth, the further demand of the research is a guideline for selecting the appropriate load balancing algorithm via virtual machine migration for characteristics-based specific applications. Hence, this paper presents a comprehensive survey on existing virtual machine migration and selection processes to understand the specific application-oriented capabilities of these strategies with the advantages and bottlenecks. Also, with the understanding of the existing measures for load balancing, it is also important to furnish the further improvement strategies, which can be made possible with a detailed understanding of the parallel research outcomes. Henceforth, this paper also equips the study with guidelines for improvements and for further study. Nonetheless, the study cannot be completed without the mathematical analysis for better understanding and experimental analysis on different standards of datasets for better conclusive decisions. Hence, this paper also presents the discussion on mathematical models and experimental result analysis for the conclusive decision on the improvement factors and the usability of the migration methods for various purposes. Finally, this paper is a comprehensive survey on the background of the research, recent research outcomes using mathematical modeling and experimental studies on various available datasets, and finally identify the scopes of improvements considering various aspects such as execution time, mean time before a VM migration, mean time before a host shutdown, number of node shutdowns, SLA performance degradation, VM migrations, and energy consumption.

INDEX TERMS data center, load balancing, task scheduler, FIFO, FAIR, capacity, hybrid, LATE, SAMR, context-aware, threshold, IQR, LR, MAD, LRR, THR, VM consolidation, VM migration, MC, MMT, RS, MU, planetlab, metric, VM migration analysis, energy consumption analysis, SLA analysis.

I. INTRODUCTION

In the recent time, multiple parallel research outcomes have demonstrated many notable strategies for load balancing on cloud-based data centers. Nonetheless, the strategies are criticized by various other research attempts for several factors. The critics have not only done an extensive testing on the existing strategies for load balancing, rather also opened the newer research directions. Nevertheless, the analysis of the load balancing strategies is limited on few parametric analyses and many of the types are independent to the application domain or the nature of the cloud-based data centers as public, community, hybrid or private.

With the strong believe in this research direction, it is the demand of the modern research to justify the analysis of existing load balancing algorithms on various factors as application domain types, load types and with a significantly strong metric for comparisons. In one of the recent publications by Singh and Kumar [72] have demonstrated the importance of the prior knowledge for selecting a proper load balancing algorithm with all security features. This work by Singh and Kumar [72] shows that the single concentration on workload and workload management cannot justify the other aspects of the cloud-based datacenter demands. Also, the work by Singh and Kumar [72] clearly suggests that the data into the application or the nature of the work load also plays a crucial role in deciding the load balancing strategy or the virtual machine migration process. Further, the reports from the work of Murk *et al.* [73] confirms the believe that, various factors such as domain of the applications or load types or the combination of both the factors can be the decision-making factor for selecting the appropriate load balancing and virtual machine migration techniques.

The load balancing techniques on cloud computing is the generic framework-based process where the generated workloads are distributed over multiple data centers resources. These techniques bring the advantage of lower response time. Nonetheless, the cost of replication of resources is also to be taken care as an additional cost. The cloud data center-based load balancing is distinguished from the domain name service-based load balancing. The domain name service load balancers deploy the hardware and software components to balance load for the hardware resources, whereas the cloud-based load balancing techniques deploys the software algorithms or protocols to distribute the load over multiple software services. Also, it is to be understood that, the cloud-based load balancing techniques allows the customers to use the global or geodetically distributed services based on geodetically distributed servers. The benefits achieved from data center load balancing are numerous as Handling the high unexpected traffic generally referred to Cyber Spikes, Making the application scalable based on the demand without degrading the performance, Increases the reliability at the cost of replication of services and Reduces the hardware cost compared to the DNS based load balancing.

Henceforth, in this chapter of the work, the existing algorithms or strategies for load balancing are analyzed on the proposed factors of survey with theoretical and experimental analysis.

II. BACKGROUND OF THE LOAD BALANCING STRATEGY

In this section of the work the background of the research is analyzed. This strategical survey on the origination of the load balancing starts from the distribution of the task and further task scheduling for the distributed processing. In the initial phases of the research, the major challenge is to identify the distribution schedule and task concurrency control. The parallel researches searched for a framework, where the task, the data, on which the tasks will be operational, and the data generated by the task will be managed without the concurrency issues. Hence, the further research continues towards building such frameworks and scheduling tasks on these frameworks. The work of Wylie *et al.* [1] demonstrated the task scheduling for distributed computing environments. This work enables the idea of utilizing MapReduce computing techniques for processing jobs and connected data during task scheduling. The experimental work by Dean and Ghemawat [2] and the initial roadmap for using Google distributed file system by S. Leung *et al.* [3] demonstrates the benefits of using Hadoop and MapReduce. The demand for using distributed processing systems kept on growing with the demand for faster processing for big data applications using several analytics, learning and deep mining of the large-scale data. These applications demand faster processing, which can be accomplished using distributed sorting mechanism, analysis of task logs and indexes or specifically the inverted indexes. The use of indexes ensures the job allocation must be aware of the resources available as showcased by Tan *et al.* [4]. Also, the work of Mao *et al.* [5] discovers a newer paradigm of building dynamic indexes to enable load driven task allocations. A wide range of resource or load aware task scheduling techniques where been introduced in the recent past. The work of Konar *et al.* [6] provides the guideline for building improved scheduling algorithms and Liu *et al.* [7] demonstrates the guidelines for measuring the performance of any scheduling algorithms.

During the last decade several researchers have contributed significantly towards building the most sophisticated scheduling algorithm for distributed tasks. The fundamental algorithm for scheduling distributed jobs is the FIFO scheduler and the work of Pei *et al.* [8] have showcased the

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

improvements over FIFO scheduler. The major drawback of this strategy is the significant ignorance of priority of the tasks and the locality of the resources.

Utilization of the indexing mechanism for making the scheduler algorithm aware of the available resources is one of the most prominent improvements over FIFO scheduler which is demonstrated by Yong *et al.* [9]. The other issue with the traditional scheduling methods is resolved by Sarma *et al.* [10] by utilizing the locality of the resource principle.

The above-mentioned efforts are fundamentally criticized to obtain the basic demand from the consumers of the distributed applications. The basic demand of the consumers is to achieve higher responsive application. The resource towards achieving higher responsiveness or lower response time is intrigued by the research of Dai and Bensaou [11]. This research establishes the road map for building higher responsive algorithms for scheduling tasks on any framework managing the task and resource distribution.

Yet another challenge faced by the research community is to deal with the resources which are heterogeneous in nature. The major bottleneck of this situation is to decide the priority or weightage of resources due to the heterogeneous nature and the demand of the resources. An initial proposal to resolve this issue is proposed by Elkholy *et al.* [12]. This work ensures the demand-based adaptive scheduling policy for task scheduling. Further the enhancements of this proposal are presented by Nayak *et al.* [13] utilizing the benefits from cloud-based services for distributed task and resource managements. Yet another direction of these proposals is presented by Xu *et al.* [14] for prioritizing the utilization of the load types with the resources and resource types.

The scheduling of tasks based on only on the principal of locality is also widely accepted in the research community. The placement of the tasks using the nearest locality policies is demonstrated by the parallel research attempt of Archana *et al.* [15]. This work is criticized by various other research attempts for lack of precision. The enhancement is also being proposed and the work of Mao *et al.* [16] can demonstrate significant improvement of accuracy for detecting the most highly populated node with the resources for task allocation. The fine tuning of the location for task allocation can be decided using the density of the resources and dependent tasks in any node. The calculation method for these purposes is proposed by Yao *et al.* [17] and Moseley *et al.* [18].

Nonetheless, none of the recent research demonstrates the comparative analysis of the baseline for distributed task processing and does not present any understandable configuration differentiations using mathematical analysis.

Henceforth, this section of the work, firstly formulates the task distribution paradigm and further analyses the load or task scheduling mechanisms.

III. MATHEMATICAL MODELLING FOR TASK ALLOCATION

This section of the work analyses the fundamental factors and the processes for scheduling distributed tasks. This component of the research will certainly highlight the entry point for enhancement of the research attempts. The mathematical analysis is carried out considering the open source framework Hadoop [19].

The task sets running on Hadoop framework can be denoted as T and each Map-Reduced component can be denoted as t. Thus, the relation of taskset cane can be formulated as,

$$T = \sum_{i=1}^{n} t_i \tag{1}$$

Assuming that, total number of members in the taskset is n.

Further, considering the multi-user environment, the user set can be denoted as U and each member of the user set is denoted as u. Hence, the formulation can be furnished as,

$$U = \sum_{j=1}^{m} u_j \tag{2}$$

Also, based on the principle of task affinity of the users, the relation between the users and submitted tasks can be observed as,

$$u_j \rightarrow t_i \tag{3}$$

Furthermore, assuming the default multiple queue scheduling arrangements in the Hadoop framework, the entire queue configuration can be denoted as Q and each level of the queues can be comprised of multiple dependent queue as q. Hence, the configuration can be represented as,

$$Q = \sum_{k=1}^{r} q_k \tag{4}$$

As mentioned, each queue in the queue group is a dependent multi-level queue as can be denoted as,

$$\frac{d}{dL} q_k = q_k(L_v) \tag{5}$$

And the level of the queue configuration can be represented as,

$$L = \int_{k=1}^{X^Y} L_k \tag{6}$$

where, X and Y denotes the partitions and levels for queue configurations respectively.

Henceforth, the task allocation can be formulated as,

$$t_i \rightarrow q_k(L_v) \tag{7}$$

And the user affinity can be furnished as,

$$u_j \rightarrow \langle t_i \rightarrow q_k(L_v)\rangle \tag{8}$$

Thus, in the light of Eq. (1.8), the traditional system for distributed task scheduling is explained.

IV. EXPERIMENTAL MODEL FOR HADOOP TASK ALLOCATION

This section of the work analyses the default scheduler arrangements and the factors measuring the performances for each scheduler methods.

The scheduling of the work is the most issue settling part in MapReduce. In any case, the evil administration of the planning methods may prompt high time multifaceted nature, low information area or synchronization issues. In this work, the current calculations and methods accessible in the Hadoop system are dissected with comprehension of the centrality in time and information multifaceted nature worldview.

A. FIFO SCHEDULING TECHNIQUE (FIRST IN FIRST OUT)

The initial and primitive type of job scheduling technique available with Hadoop framework is the FIFO scheduler [19]. The basic principle for FIFO scheduler is to allocate the tasks to the Hadoop queue based on the time of job submission and reduction of the tasks. During the deployment of the enhanced version of the FIFO scheduling algorithm, the tasks can be made parallel in processing and can reside back on the job queue.

The performance of the FIFO scheduler must be measured under the situation of heterogenous task types and by responsiveness of the tasks [Table – 1].

TABLE 1. Performance metric for FIFO.

Metric Parameter	Metric Measures
Job Type	The job type parameter measures the amount heterogeneity available in the job types.
Responsiveness	The response time or the completion time for the entire job set can be measured using the responsiveness parameter.

B. FAIR SCHEDULING TECHNIQUE

The second type of the scheduler algorithm is the FAIR scheduler. This scheduler works on the basic principle of identifying the available resource slots for mapper and reducer process and further allocate the tasks to the free slots [20]. Based on the survey work by Andrews and Binu [21] it can be concluded that, the allocation of tasks performance must be measured using computation time for task processing distribution. Also, the parallel research outcome by Chen *et al.* [22] the resource pools accessibility is also one the criteria for performance measure.

TABLE 2. Performance metric for FAIR.

Metric Parameter	Metric Measures
Resource Pool Configuration	This parameter denotes the resource availability for the parallel processing and the replication factors for each resource types.
Job Characteristics	This parameter denotes the scheduled task affinity to each pool satisfying the locality of the resources.

The performance of the FAIR scheduler can be measured by the characteristics of the job pool and task affinity to the job pools [Table – 2].

C. CAPACITY SCHEDULING TECHNIQUE

The enhanced version of the FAIR scheduling technique is noted as the Capacity scheduling algorithm. The FAIR scheduler emphasizes on the fair allocation of the resources in terms of resource pools, in the other hand, capacity scheduling mechanism focuses on allocating the tasks on queue. The work demonstrated by Tiwari [23] have significantly establishes the believe that allocation of task to the queue having the maximum possible resources can outperform the responsiveness of the tasks over the FAIR scheduling technique.

The performance of the capacity scheduler can be measured by the characteristics of the queue and the parallelization of the scheduled tasks [Table – 3].

TABLE 3. Performance metric for capacity.

Metric Parameter	Metric Measures
Queue Characteristics	This parameter focuses on the queue characteristics based on connected clusters to each queue for jobs.
Parallelization of Tasks	This parameter enhances the information gained on the parallelization of the task behaviors.

D. HYBRID SCHEDULING TECHNIQUE

Bounded by the limitations of utilization of the information of either the resource pools or the clustering of resources, the above-mentioned scheduling methods are limited in performance. Hence, Nguyen *et al.* [24] proposed another scheduling mechanism to utilization the awareness of both information and named the scheduler algorithm as Dynamic Priority Based Hybrid scheduler. This algorithm also incorporates the priority of the jobs submitted to the job queue along

TABLE 4. Performance metric for hybrid.

Metric Parameter	Metric Measures
Queue Responsiveness	This parameter denotes the waiting time of the tasks on the processing queue.
Dynamic Priority	This parameter denotes the ever-increasing priority of the jobs based on the waiting time.
Locality-Management	This parameter denotes the percentage of the resources available on the local node.

with the information of resource pools based on locality of resources principle.

The performance of the hybrid scheduler can be measured by the responsiveness of the task processor and the dynamic priority of the scheduled tasks [Table – 4].

E. LATE SCHEDULING TECHNIQUE (LONGEST APPROXIMATE TIME TO END)

Scheduling of jobs based on the priority or the locality of resources can be improved utilizing the task burst time. The work by Zaharia *et al.* [25] have demonstrated the LATE scheduling mechanism or the scheduling method based on Longest Approximate Time to End. The LATE scheduling algorithm does not focus on the resource and job locality principles.

The performance of the LATE scheduler can be measured by the task bust time and the priority of the scheduled tasks [Table – 5].

TABLE 5. Performance metric for LATE.

Metric Parameter	Metric Measures
Remaining Burst Time	This parameter denotes the remaining burst time for the task.
Task Priority	This parameter denotes the priority factors for the task.

F. SAMR SCHEDULING TECHNIQUE (SELF-ADAPTIVE MAPREDUCE)

Despite the best research attempts available for utilizing the resource locality or the task priority or the resource availability, the task processing cannot be changed and be delayed once allocated. Thus, in the search of finding the automated adjustable scheduler, Q. Chen *et al.* [26] have proposed a scheduler algorithm to identify the slower running nodes and further migrate the running tasks to the higher performing node for fast completion of the jobs.

The performance of the SAMR scheduler can be measured by the execution time and the priority of the scheduled tasks [Table – 6].

TABLE 6. Performance metric for SAMR.

Metric Parameter	Metric Measures
Responsiveness	The response time or the completion time for the entire job set can be measured using the responsiveness parameter.
Task Priority	This parameter denotes the priority factors for the task.

G. CASH SCHEDULING TECHNIQUE (CONTEXT-AWARE SCHEDULING FOR HADOOP)

The allocation of tasks to the queues or to the clusters or to the resource pools is not always the most feasible factor. Many of the times, it is been observed that the most obvious allocation is not the most feasible possibility. Hence Kumar *et al.* [27] have proposed a newer method for task allocation based on the context of the execution and the context of the resources, as CASH, are available. This proposed method demonstrates a higher feasibility of task allocation rather than increased waiting time in case of unavailability of the resources.

The performance of the CASH scheduler can be measured by the execution time and the relevancy of the scheduled tasks [Table – 7].

TABLE 7. Performance metric for context-aware.

Metric Parameter	Metric Measures
Responsiveness	The response time or the completion time for the entire job set can be measured using the responsiveness parameter.
Context	This parameter denotes the context awareness of the resources having higher affinity to the tasks.

Further, this section of the work, presents a detailed analysis of the comparison for methods available for Hadoop based job scheduling [Table – 8].

V. UPGRADATION OF THE RESEARCH FOCUS

After the detailed analysis on the background of the research problem, it is natural to realize that all the previous research attempts are bounded by the limitation of available resources. The available resources can easily be exhausted during the surge on the consumer requests. Hence, the limitation of the resources must be overcome using scalability of the resources. The scalability of the resources can be achieved by deploying the resources and the tasks and the manageability feature on cloud computing environment.

The scalability of the applications or the jobs or the tasks submitted to the cloud computing can be achieved in two different architectural patterns as vertical and horizontal. Vertical is often thought of as the ‘‘easier’’ of the two methods. When scaling a system vertically, you add more power

TABLE 8. Comparative analysis for hadoop based schedulers.

Algorithms Metric	FIFO	FAIR	Capacity	Hybrid	LATE	SAMR	CASH
Job Characteristics	✓	✓					
Responsiveness	✓					✓	✓
Resource Pool Configuration		✓					
Queue Characteristics			✓				
Parallelization of Tasks			✓				
Queue Responsiveness				✓			
Dynamic Priority				✓			
Locality-Management				✓			
Remaining Burst Time					✓		
Task Priority					✓	✓	
Context							✓

to an existing instance. This can mean more memory (RAM), faster storage such as Solid-State Drives (SSDs), or more powerful processors (CPUs). The reason this is thought to be the easier option is that hardware is often trivial to upgrade on cloud platforms like AWS, where servers are already virtualized. There is also very little (if any) additional configuration you are required to do at the software level.

In the other hand, the Horizontal scaling is slightly more complex. When scaling your systems horizontally, you generally add more servers to spread the load across multiple machines. With this, however, comes added complexity to your system. You now have multiple servers that require the general administration tasks such as updates, security and monitoring but you must also now sync your application, data and backups across many instances.

The main benefit of scalable architecture is performance and the ability to handle bursts of traffic or heavy loads with little or no notice. A scalable system can help keep your application or online business running during peak times and not end up losing you money or damaging your reputation. Having your system set up into services such as the microservices system architecture can make monitoring, feature updates, debugging and scaling easier.

Supplementary, the desired advancements, which are materialized in this study are listed here:

- Firstly, the traditional algorithms with the focus on job type or application characteristics must include the benefits from the strategies depending on the correlation-based algorithms.
- Secondly, the algorithms with the focus on responsiveness of the applications or the response time of the applications must enhance the strategies using the migration time-based strategies or can be also enhanced using the thresholding-based methods.
- Third, the resource-based existing algorithms can be further modified using the maximum utilization strategies with regression methods.
- Fourthly, the algorithms with the emphasis on parallelization of the applications can be further enhanced using the locality of the thresholding methods.
- Finally, the content or priority focused algorithms can be made more efficient using the advantages of the randomness in the application context.

These advancements of the methodologies are discussed in the further sections of this work.

Realizing the benefits from scalability of the cloud computing environments, this research further considers the recent load balancing strategies on cloud-based data centers.

Henceforth, in the next section of the work, the parallel research outcomes are analyzed.

VI. EXISTING METHODS AND SYSTEMS

In the recent past, motivated by the challenges of limitation of the resources, a large number of research attempts were presented for balancing the load for high performance computing systems. The primary goal of these research attempts is to increase response time for the applications deployed on cloud-based data centers using migrations of virtual machine for balancing the loads. In this section of work, the parallel research outcomes are analyzed.

The primary strategy for identifying the highly loaded and less loaded instance on cloud is the virtual machine consolidation. The consolidation process can be achieved by many possible algorithms as demonstrated by various researchers. One of the prominent attempts by Feller and Morin [28] showcased the consolidation of the virtual machines on a decentralized environment. Nonetheless, the work is criticized for lack of synchronization. Another parallel research attempt by Marzolla and Babaoglu [29] have demonstrated the use of mutual information sharing or gossiping on the information available with the hypervisor for VM consolidation. This work is also criticized for being highly dependent on the hypervisor information update, causing higher workload on the cloud instances and compromises on the response time.

During the virtual machine migration, the cost for migrating the VMs is the energy. Hence many of the research attempts have approached the problem giving higher priority for energy conservation and moderated priority for response time. One of the similar directional research is presented by Murtazaev [30]. This work is one of the notable enhancements on the proposal given by Vogels [31], where the initial direction for further research was established as the action items for research beyond the VM consolidations for distributed systems.

In order to consolidate the virtual machines, the research attempts have to depend on the load or specifically overload detection mechanism for the virtual machines. The initial strategy for overload detection is proposed by Beloglazov and Abawajy [32]. This proposed approached demonstrates that the detection of the overload can be easily performed by utilizing one definite threshold. The key drawback of this approach is the determination of the accurate threshold for the complete system. Motivated by the drawback, this work is further enhanced by the same author, Beloglazov and Buyya [33] with an approach for determination of the load threshold dynamically.

The dynamic detection of the load for calculation of the threshold was a ground-breaking approach and further many researchers have attempted to improve the dynamic detection of load on the VMs. One of the recent research attempts, by F. Farahnakian and Liljeberg [34] and subsequently enhanced by Farahnakia and Pahikkala [35] is to deploy regression method on historical utilization of the computing resources, specifically the CPU utilization.

The CPU utilization analysis was severely criticized by other researchers as the CPU utilization can vary based on the application or job type deployed on the virtual machines. The applications demanding more resources for storage or information processing can rather depend highly on the storage resources rather than the computing resources like CPU. Hence, few of the research attempts can be visualized to approach the problem by calculating the load threshold depending on the service level agreements as demonstrated by Ajiro and Tanaka [36], Wang *et al.* [37] and subsequently by Wood *et al.* [38].

In the other hand, the work by Dorigo and Gambardella [39] have initiated another possible criticism by highlighting the factor that depending one any one parameter for determination of the threshold cannot be the most optimal possible way. This roadmap is also accepted by few of the research attempts and Dorigo *et al.* [40] demonstrated the possibilities on incorporating multiple factors during the load threshold determination. This was is further realized and improved by Harman *et al.* [41] by deploying artificial searching strategies to find the most suitable set of parameters for load detection from job characteristics.

Majority of the researches have adopted the hybrid policy for Threshold Detection as Inter Quartile Range, Local Regression, Median Absolute Deviation, Robust Local Regression & Static Threshold and for VM Consolidation policy as Maximum Correlation, Minimum Migration Time, Minimum Utilization & Random Selection.

Further, in this section of the work, the above-mentioned strategies are analyzed.

This section of the work analyses the threshold computational algorithms as one of the primary tasks to be performed for identification of the highly, moderately and least loaded physical hosts on cloud as demonstrated by Theja and Babu [42].

A. INTER QUARTILE RANGE (IQR) BASED METHODS

The traditional method of data visualization and analytics provide the measure of data distribution as quartile. The quartile measure provides the distribution of the data items over the range and further the range is divided into three major segments based on the median variances.

The mathematical analysis is presented here:

Assuming that T is the set of elements with CPU utilization and each element in the T set is denoted as Ci. Hence the relation can be formulated as:

$$T = \prod_{i=1}^{N} C_i \tag{9}$$

where N is considered to be the length of the set T, after eliminating the possibility of the NULL sets and can be expressed as

$$N = \Phi(T) \tag{10}$$

Henceforth, the median element of the set T can be expressed as,

$$Ck \leftarrow T[\frac{N}{2}] \tag{11}$$

Further, the Q1 as the lower quartile can be formulated as,

$$Q1 \leftarrow \prod_{i=1}^{\frac{(N+1)}{4}} C_i \tag{12}$$

Similarly, the Q3 as the upper quartile can be formulated as,

$$Q3 \leftarrow \prod_{i=1}^{\frac{3(N+1)}{4}} C_i \tag{13}$$

Finally, the IQR can be expressed as,

$$IQR \leftarrow Q3 - Q1 \tag{14}$$

Or,

$$IQR \leftarrow T - (\prod_{i=1}^{\frac{3(N+1)}{4}} C_i - \prod_{i=1}^{\frac{(N+1)}{4}} C_i) \tag{15}$$

The visual representation is furnished here [Fig – 1].

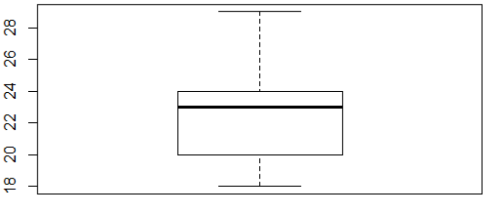


FIGURE 1. IQR analysis.

This principle is applied to measure the threshold detection as demonstrated in the following algorithm.

Algorithm 1 Existing Threshold Detection Using Inter Quartile Range (IQR)

```
Step-1. Accept the list of active nodes
Step-2. For each active node accept the list of
        Virtual Machine
            a.  Initialize the CPU utilization and sort in
                ascending order
            b.  Build the quartiles for each segment
            c.  Calculate IQR as Q3 - Q1
Step-3. Calculate the threshold, Q as 1 – Safety_Range
        ∗ IQR
Step-4. If the host CPU utilization > Q
    a.  Then mark the host as overloaded
Step-5. Else,
    a.  Mark the host node as safe node
Step-6. Report the final overloaded list of hosts
```

B. LOCAL REGRESSION (LR) BASED METHODS

Yet another traditional approach for predicting one component of the research based on another component the regression method. The local regression method defines a more sophisticated approach for determining the progression of the testing dependent variable based on another local variable. The local regression method enables the higher accuracy of the prediction process.

The mathematical analysis is presented here:

Assuming that T is the set of elements with CPU utilization and each element in the T set is denoted as Ci. Hence the relation can be formulated as:

$$T = \prod_{i=1}^{N} C_i \tag{16}$$

where N is considered to be the length of the set T, after eliminating the possibility of the NULL sets and can be expressed as

$$N = \Phi(T) \tag{17}$$

Considering L is the driving load variable in the data center, the dependency equation can be realized as

$$\oint (L) \rightarrow T \tag{18}$$

Further, the weight functions, CW and LW as CPU capacity weight and the Load weight functions respectively, calculation can be visualized as

$$CW(t) \leftarrow \frac{1}{(1 - C[]^3)^3} \tag{19}$$

$$LW(t) \leftarrow \frac{1}{(1 - L[]^3)^3} \tag{20}$$

The visual representation is furnished here [Fig – 2].

This principle is applied to measure the threshold detection as demonstrated in the following algorithm.

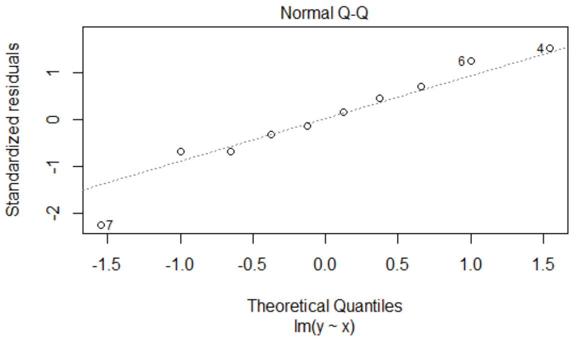


FIGURE 2. LR analysis.

Algorithm 2 Existing Threshold Detection Using Local Regression (LR)

```
Step-1. Accept the list of active nodes
Step-2. For each active node accept the list of Virtual
        Machine
            a.  Initialize the CPU utilization and sort in
                descending order as C[]
            b.  Initialize the increase in load pattern as P[]
            c.  Calculate the CPU utilization weight function
                as CW = 1/((((1 - C[]>^(C[] - 1])^3)^3)
            d.  Calculate the load pattern weight function as
                LW = 1/((1 - P[]>^(P[] - 1])^3)^3)
Step-3. Calculate the threshold, R as (C[]*CW +
        P[]*LW)* Safety_Range
Step-4. If the host CPU utilization > R
            a.  Then mark the host as overloaded
Step-5. Else,
            a.  Mark the host node as safe node
Step-6. Report the final overloaded list of hosts
```

C. MEDIAN ABSOLUTE DEVIATION (MAD) BASED METHODS

The traditional method for calculating the change pattern in the time series is deviation process. The deviations can be calculated from the mean, median or from the mode by calculating the difference measures. Nevertheless, the most accurate measure can be observed from the median calculation aspects. Once the deviation for all the elements are calculated, further mean must be calculated to achieve the final objective as median absolute deviation.

The mathematical analysis is presented here:

Assuming that T is the set of elements with CPU utilization and each element in the T set is denoted as Ci. Hence the relation can be formulated as:

$$T = \prod_{i=1}^{N} C_i \tag{21}$$

where N is considered to be the length of the set T, after eliminating the possibility of the NULL sets and can be expressed as

$$N = \Phi(T) \tag{22}$$

Hence, calculating the median element from the set T is formulated as following, where M denotes the median element,

$$M \leftarrow T[{}^{(N+1)}\!/_{2}] \tag{23}$$

Further, calculation of the median deviation, denoted as DM, can be calculated as,

$$DM[] \leftarrow \left| M - \int_{i=1}^{N} C_i \right| \tag{24}$$

It is natural to realize that, the number of elements in set T will be equal to the set DM.

Further calculation of median absolute deviation, denoted as R, can be presented as,

$$R \leftarrow DM[{}^{(N+1)}\!/_{2}] \tag{25}$$

The visual representation is furnished here [Fig – 3].

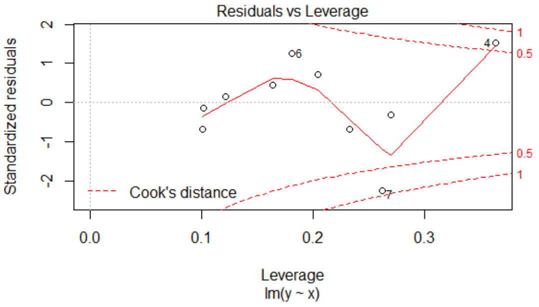


FIGURE 3. MAD analysis.

This principle is applied to measure the threshold detection as demonstrated in the following algorithm.

Algorithm 3 Existing Threshold Detection Using Median Absolute Deviation (MAD)

```
Step-1. Accept the list of active nodes
Step-2. For each active node accept the list of
        Virtual Machine
        a.  Initialize the CPU utilization and sort
            in descending order as C[]
        b.  Calculate the Median of the C[]
        c.  Calculate the Deviation Sum D[] as
            (Median - C[])
Setp-3. Calculate the deviation median as DM
        as Median(D[])
Setp-4. Calculate the threshold, R as
        (1 - DM∗Safety_Range)
Setp-5. If the host CPU utilization > R
        a.  Then mark the host as overloaded
Setp-6. Else,
        a.  Mark the host node as safe node
Setp-7. Report the final overloaded list of hosts
```

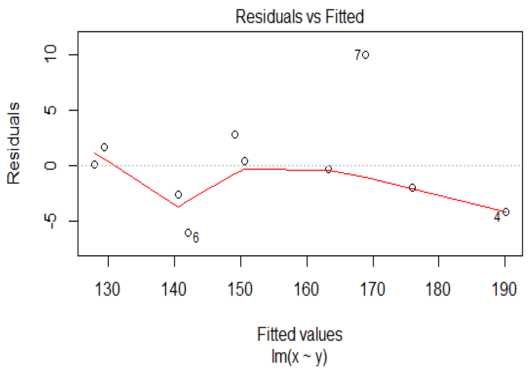


FIGURE 4. LRR analysis.

D. ROBUST LOCAL REGRESSION (LRR) BASED METHODS

Yet another enhancement to the traditional approach for predicting one component of the research based on another component the regression method. The local regression method defines a more sophisticated approach for determining the progression of the testing dependent variable based on another local variable. The local regression method enables the higher accuracy of the prediction process. The enhancements made to the local regression method is to reduce the errors using the weight function combined for both the variables.

The mathematical analysis is presented here:

Assuming that T is the set of elements with CPU utilization and each element in the T set is denoted as Ci. Hence the relation can be formulated as:

$$T = \prod_{i=1}^{N} C_i \tag{26}$$

where N is considered to be the length of the set T, after eliminating the possibility of the NULL sets and can be expressed as

$$N = \Phi(T) \tag{27}$$

Considering L is the driving load variable in the data center, the dependency equation can be realized as

$$\int (L) \rightarrow T \tag{28}$$

Further, the weight functions, CW and LW as CPU capacity weight and the Load weight functions respectively, calculation can be visualized as

$$CW(t) \leftarrow \frac{1}{(1 - C[]^3)^3} \tag{29}$$

$$LW(t) \leftarrow \frac{1}{(1 - L[]^3)^3} \tag{30}$$

Finally, the combined weight function, CLW can be calculated as,

$$CLW(t) \leftarrow \frac{1}{(1 - \{CW[].LW[]\}^3)^3} \tag{31}$$

The visual representation is furnished here [Fig – 4].

This principle is applied to measure the threshold detection as demonstrated in the following algorithm.

Algorithm 4 Existing Threshold Detection Using Robust Local Regression (LRR)

```
Step-1. Accept the list of active nodes
Step-2. For each active node accept the list of
        Virtual Machine
          a.  Initialize the CPU utilization and sort
              in descending order as C[]
          b.  Initialize the increase in load pattern as P[]
          c.  Calculate the CPU utilization weight function
              as CW = 1/((((1 - C[]>^(C[] - 1])^3)^3)
          d.  Calculate the load pattern weight function as
              LW = 1/((1 - P[]>^(P[] - 1])^3)^3)
          e.  Calculate the combined weight function as
              CLW = 1/((1 - CW[]>^(CW[] - 1])^3)^3)
              * 1/((1 - LW[]>^(LW[] - 1])^3)^3)
Step-3. Calculate the threshold, R as (C[]*CW + P[]*LW
        + CLW[] ) *Safety_Range
Step-4. If the host CPU utilization > R
Step-5. Then mark the host as overloaded
Step-6. Else,
          a.  Mark the host node as safe node
Step-7. Report the final overloaded list of hosts
```

E. STATIC THRESHOLD (THR) BASED METHODS

Lastly, the static threshold-based methods are subjected to the assumption of the load and CPU utilization. These methods are highly criticized for making questionable and independent assumptions to compute the threshold values for identification of the overloaded virtual machines and physical cloud-based hosts.

The details of the other algorithms are analyzed and discussed in the further sections of this chapter.

VII. CLASSIFICATION OF ALGORITHMS FOR VIRTUAL MACHINE CONSOLIDATIONS

This section of the work mathematically analyses the existing methods and algorithms for selecting or consolidating the virtual machines. Virtual machine selection is one of the second most important tasks after the thresholds are identified. As showcased in the work of Perla Ravi Theja et al [42] and subsequently by SK. Khadar Babu et al. [43], the classifications are presented here.

A. MAXIMUM CORRELATION (MC) BASED METHODS

The method for detecting the correlation is used for determining the dependencies between multiple research parameters. The relation between two parameters can be identified easily using the any domain function. Nonetheless, the correlation between more variables for any research problem demands higher computations parameters to be included.

The mathematical analysis is presented here.

Assuming the set of parameters to be considered for finding the correlation is P and every parameter in the set is assumed as Ai. Hence, the basic formulation can be realized as,

$$P \leftarrow \sum_{i=1}^{K} A_i \tag{32}$$

Here, K is assumed to be the total number of elements in set P.

Further, the correlation set C denotes the all possible correlations between the attributes in the set P and can be formulated as,

$$\int (\sum_{i=1}^{K^K} C_i) \leftarrow \int P \tag{33}$$

Henceforth, finding the maximum correlated set will produce the desired outcome for the analysis and can be represented as, MC, as,

$$MC \leftarrow \Im[\int (\sum_{i=1}^{K^K} C_i)] \tag{34}$$

The visual representation is furnished here [Fig – 5].

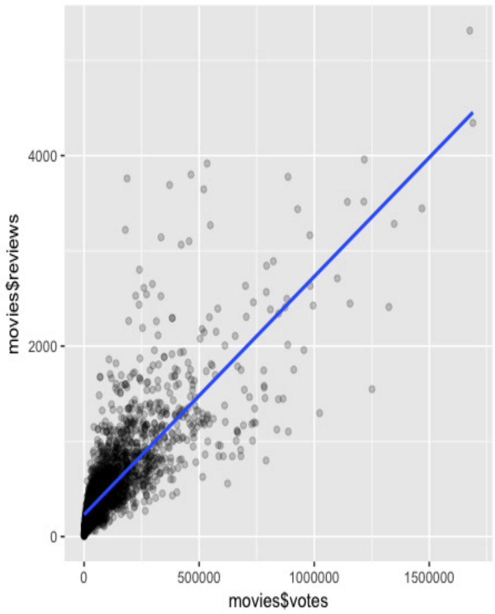


FIGURE 5. BC analysis.

This principle is applied to measure the virtual machine selection or consolidation as demonstrated in the following algorithm.

B. MINIMUM MIGRATION TIME (MMT) BASED METHODS

The detection of highly loaded virtual machines can be helpful in determining which virtual machine is to be migrated to less loaded node or physical instance. However, many of the situation can be observed where the many virtual

Algorithm 5 Existing Virtual Machine Consolidation Using Maximum Correlation (MC)

```
Step-1. Accept the list of overloaded virtual machines
Step-2. For each virtual machine
        a.  Initialize the utilization matrix with Available
            and Utilized Resources as C[][]
Step-3. Identify the transpose of the matrix C[][] and
        denote as C'[][]
Step-4. For each element in C[][] and C'[][]
        a. Identify the correlation
Step-5. Report the virtual machines with highest
        correlation
```

machine or physical hosts will be under similar load situations. Thus, finding the specific virtual machine is a real challenging task. Considering the energy complications, this minimum migration time algorithm proposes identifies the VM to migrate based on the least migration time and concerning to the this, the less energy consuming VM.

This principle is applied to measure the virtual machine selection or consolidation as demonstrated in the following algorithm.

Algorithm 6 Existing Virtual Machine Consolidation Using Minimum Migration Time (MMT)

```
Step-1. Accept the list of overloaded virtual machines
Step-2. Calculate the network transfer speed as NS
Step-3. For each virtual machine
        a.  Calculate the size of the RAM as SR
        b.  Calculate the size of the Application as SA
        c.  Find the transper time as $(SR * SA)/NS$ and
            denote as TT
        d.  Find the total migration time MT as
            $Shutdown\_Time + Startup\_Time + TT$
Step-4. Report the virtual machines with lowest MT
```

C. RANDOM SELECTION (RS) BASED METHODS

In computing randomness defines a larger problem space. The selection in any random order from a definite selection space can be highly complex as the selection must not follow any specific pattern. Nevertheless, the randomness in selection of the research item does not depend on any external knowledge, hence the possibilities of higher accuracy is questionable. However, the time complexity can be very less in case of the best possible selection during the best-case scenarios.

This principle is applied to measure the virtual machine selection or consolidation as demonstrated in the following algorithm.

Algorithm 7 Existing Virtual Machine Consolidation Using Random Selection (RS)

```
Step-1. Accept the list of overloaded
        virtual machines
Step-2. Calculate the network transfer speed
        as NS
Step-3. For each virtual machine under the
        loaded nodes
        a.  Migrate the virtual machine to less
            loaded node
            i.  If the load is balanced
                1.  Continue with the migration
            ii. Else
                1.  Reverse the migration
Step-4. Report the final migration
```

TABLE 9. Possible combinations for load balancing [45].

Algorithm Combination	Threshold Detection Policy	VM Selection / Consolidation Policy
IQR MC	Inter Quartile Range	Maximum Correlation
IQR MMT	Inter Quartile Range	Minimum Migration Time
IQR MU	Inter Quartile Range	Minimum Utilization
IQR RS	Inter Quartile Range	Random Selection
LR MC	Local Regression	Maximum Correlation
LR MMT	Local Regression	Minimum Migration Time
LR MU	Local Regression	Minimum Utilization
MAD MC	Median Absolute Deviation	Maximum Correlation
MAD MMT	Median Absolute Deviation	Minimum Migration Time
MAD MU	Median Absolute Deviation	Minimum Utilization
MAD RS	Median Absolute Deviation	Random Selection
LRR MC	Robust Local Regression	Maximum Correlation
LRR MMT	Robust Local Regression	Minimum Migration Time
LRR MU	Robust Local Regression	Minimum Utilization
LRR RS	Robust Local Regression	Random Selection
THR MC	Static Threshold	Maximum Correlation
THR MMT	Static Threshold	Minimum Migration Time
THR MU	Static Threshold	Minimum Utilization
THR RS	Static Threshold	Random Selection

D. MINIMUM UTILIZATION (MU) BASED METHODS

The final classifications of the virtual machine migration algorithms focus on the generating less overhead for the longer processing durations of the jobs or the applications. The minimum utilization policy ensures to migrate less loaded virtual machines to moderately loaded nodes to ensure balancing of the loads by utilizing the principle of energy consumption reduction.

Henceforth, with the detailed analysis of the possible virtual machine load detection and migration algorithms, this work furnishes the final possible combinations of the algorithms to be utilized for load balancing. This summarization is motivated by the work of Heidari and Buyya [45] and presented here [Table – 9].

In the further sections of this work, the achieved results are analyzed and discussed.

VIII. ANALYSIS OF THE DATASETS

This work dependably analyzes the algorithms types on various public datasets. The dataset characteristics are important for the research as the parametric information in the dataset governs the deciding factors of the research.

Hence, in this section of the work, the analysis of the possible datasets is carried out with description of the parameters. In the domain of cloud-based datacenter load balancing research, the Planet Lab provides the majority of the prominent datasets. The dataset is maintained by Planet Lab organization and can be obtained freely. The description of the dataset is provided by Sevinc [46]. Thus, this research focuses primarily on the PlanetLab datasets.

The description of the parameters is discussed here:

A. DATASET NAME

The name or the dataset id is the unique identifier of the dataset, which ensures appropriate references to the actual collection of the research findings. The dataset name utilized by PlanetLab is the based on the timestamp of the process, when the collection of the data started.

B. NUMBER OF TRACES

The dataset provided by PlanetLab contains multiple trace files from different datacenters or from different listening ports. The trace files contained primary events of the datacenter directing to decide on the load balancing factors. More number of traces helps in deciding the threshold values or the identification of the virtual machines more accurately.

C. LOCATION

The location of the traces not only makes the dataset diversified, rather also helps in identifying different parametric analytics as region-based load balancers performances, pick time load analysis and deployed application types. The PlanetLab dataset revels the primary location of the trace files without intimating the personal information of the application owner or application consumers.

D. EVENT ID

In a cloud-based datacenter environment, the traces are created based on the any of the events. The type of the events can generically be referred as one of the remote methods calls by the application developers from the application consumer end. The traces generated by the event calls can further help in evaluating the performance of the load balancer in less, moderate or high load situations. The PlanetLab dataset provides the unique event ids for the record events and based on the event ids, the details of the events can be mined further.

E. NODE ID

During a distributed datacenter task processing situation, the applications and the resources can be distributed over heterogeneous or homogeneous nodes in different locations. The PlanetLab dataset provides the unique node id for each trace record and the detailed configuration of the nodes can further be mined based on the node id.

F. FAULT CODE

The fault codes are used to denote the status of the application consumers API calls. Most of the PlanetLab datasets provides the fault codes for identification of the trace errors and further enhancements of identification of the reason for failure caused by application or the load balancer.

G. MESSAGE

The PlanetLab dataset traces also contains the optional messages generated by the load balancer after completion of the virtual machine migration process.

Hence, this work analyses the capabilities of the available dataset based on the mentioned parameters here [Table – 10].

In the next section of this chapter, the performance analyzing factors for the standard algorithms are discussed.

IX. PERFORMANCE ANALYSIS

In this section of the chapter, the performance analysis of the standard load balancing algorithms is carried out. The metric for the performance analysis is also explained in this part of the work.

Firstly, this analyses the metric for the performance analysis.

A. METRIC FOR NUMBER OF HOSTS

The number of hosts in the cloud computing environment can be important to justify the load distribution for the applications. The hosts are generically the pre-defined physical server configurations on any datacenter, primarily profiled using reference to the physical configurations denoted as P[], compatibility with other peripherals denoted as F[], confirmation to the access security denoted as S[], backup configuration denoted as B[]. Hence the final configuration of the host set H[] can be represented as

$$H[] = \begin{pmatrix} P[] & S[] \\ F[] & B[] \end{pmatrix} \tag{35}$$

Also, the number of hosts denoted by N can be further formulated as,

$$N = \exists(H[]) \tag{36}$$

B. METRIC FOR NUMBER OF VMs

The virtual machines for each data center solely depend on the physical hosts. The equal distribution or the on-demand distribution of the of the physical resources builds the virtual machines. The calculation of the number of virtual machines can be formulated as

$$\exists(VM[]) = \frac{\delta}{\delta \frac{D}{dH}}(H[]) \tag{37}$$

where, VM[] and D denote the virtual machine set and load characteristics.

C. METRIC FOR TOTAL SIMULATION TIME

The total simulation time is the duration of the experiment to understand the behavioral pattern of the load balancing

TABLE 10. Dataset analysis.

Dataset Name	Availability of the Information					
	Number of Traces	Location Information	Event Id	Node Id	Fault Code	Message
20110303	1052	Oxford, Tsinghua (China), Parris (France), CA (USA), LA (USA), Ukraine	✓	✓	×	×
20110306	898	Oxford, USA	✓	✓	×	
20110309	1061	Oxford, Tsinghua (China), LA (USA), Ukraine	✓	✓	×	×
20110322	1516	Oxford, Tsinghua (China), Parris (France), CA (USA), LA (USA), Ukraine	✓	✓	✓	✓
20110325	1078	Oxford, Tsinghua (China), Parris (France)	✓	×	×	×
20110403	1463	Oxford, Parris (France), CA (USA), LA (USA), Ukraine	✓	×	✓	✓
20110409	1358	Oxford, Tsinghua (China), Parris (France)	✓	✓	✓	×
20110411	1233	Parris (France), CA (USA), LA (USA), Ukraine	✓	×	✓	×
20110412	1054	Parris (France), CA (USA), LA (USA)	✓	×	×	×
20110420	1033	Oxford, CA (USA), LA (USA), Ukraine	✓	✓	×	×

algorithm in due course of time. It is natural to realize that in the simulation environment, the actual time, denoted as T is scaled with the simulated time, denoted as K, by the pre-decided time scaling factor, denoted as S and can be formulated as,

$$K = \overset{\lim\limits_{T \to Max}}{\underset{\lim\limits_{S \to Min}}{\left[{}^{T}\!/_{S}\right]}} \tag{38}$$

D. METRIC FOR ENERGY CONSUMPTION (kWh)

The energy consumption is one of the prime deciding factors to select any virtual machine migration algorithm. Motived by the green computing, majority of the data center are adopting the VM migration strategies, which can operate in low energy mode. The calculation of the energy consumption can be done with the traditional formulation of power consumed by the data center infrastructure denoted as P and the duration of the consumption denoted as T,

$$E = P.T \tag{39}$$

E. METRIC FOR NUMBER OF VM MIGRATIONS

The number of virtual machine migration can be the deciding factor for the best service level agreement satisfiable load balancing algorithm. The virtual machine migration from host to destination causes the shutdown time and the migration time to increase and finally results in increase of response time. Hence, calculating the number of virtual machine migrations are one of the most important metrics in load balancing. The number, N, can be decided by performing the correlation between the set of identified loaded VM, L[], and the migrated VM, M[], as,

$$N = Corr\,[L[] \odot M[]] \tag{40}$$

F. METRIC FOR SLA PERFORMANCE DEGRADATION DUE TO MIGRATION

The service level agreement is the measure for any application to decide the responsiveness of the application. The consumers of the applications demand the higher service level agreements to ensure less down time for the services. It is been observed that the service levels are bound to be compromised due to the migration of the virtual machine. Nonetheless, the migration process also ensures less waiting time for the consumers. Hence, the violation of SLA is less. The SLA violation or degradation can be measured using the VM shutdown time (ST), VM transfer time (VT) and startup time (SAT) and can be formulated as,

$$SLA\_Degrade(\%) = \frac{SLA - (ST + VT + SAT)}{SLA} * 100 \tag{41}$$

G. METRIC FOR NUMBER OF NODE SHUTDOWNS

The number of hosts shutdown is the measure for the data center infrastructure efficiency or DCiE. The number of hosts shutdown can easily be measured from the dead nodes (DN)

TABLE 11. Comparative analysis for problem formulation.

Reference Number	Title & Author	Journal & Year	Proposed Solution	Limitations & Takeaways (Identified)
[47]	Storage-Tag-Aware Scheduler for Hadoop Cluster Nawab Muhammad Faseeh Qureshi ; Dong Ryeol Shin ; Isma Farah Siddiqui ; Bhawani Shankar Chowdhry	IEEE Access, July 2017	A novel endorsement of MapReduce job with storage media tag.	1. The applicability of the storage media tag is limited to homogeneous storage structures. 2. The heterogeneous storage is highly popular for the modern content storages for cloud.
[48]	Big Data 2.0 Processing Systems Taxonomy and Open Challenges Fuad Bajaber · Radwa Elshawi · Omar Batarfi · Abdulrahman Altalhi · Ahmed Barnawi · Sherif Sakr	Springer, June 2016	A taxonomy and detailed analysis of the state-of-the-art in Big Data 2.0 processing systems.	1. This is a survey work elaborating Big data processing 2. Nevertheless, the high level queuing design is ignored and must be addressed
[49]	Apache Hadoop Yarn Parameter configuration Challenges and Optimization Bhavin J. Mathiya ; Vinodkumar L. Desai	IEEE, October 2015	Customizing parameter configurations setting for performance tuning of Apache Hadoop jobs and better utilization of available resources	1. The parametric optimizations are important to obtain optimal performance 2. The heterogeneity of the jobs are overlooked 3. The heterogeneity of the cloud services are ignored
[50]	Review of Apriori based algorithms on MapReduce framework S. Singh, G. Rakhi, P. K. Mishra	Cornell University Press, Feb 2017	Discusses and analyzes the various implementations of Apriori on MapReduce framework	1. The summarization of the job types demands accurate identification of job and resource demand mapping 2. Frequency mapping for the job types are not identical to frequent item set mining
[51]	Heterogeneity in mobile cloud computing Taxonomy and open challenges Zohreh Sanaei ; Saeid Abolfazli ; Abdullah Gani ; Rajkumar Buyya	IEEE Communications Society, 2014	The impacts of heterogeneity in MCC are investigated	1. The heterogeneity dependency of the mobile cloud computing environment is well elaborated
[52]	Efficient Load-Balancing Aware Cloud Resource Scheduling for Mobile User Li Chunlin ; Zhou Min ; Luo Youlong	IEEE The Computer Journal (Volume 60, Issue 6, June 2017)	The approach augments local cloud service pools with public cloud to increase the probability of meeting the service level agreements	1. This approach decreases the SLA violation 2. Energy awareness is low
[53]	DGLB Distributed Stochastic Geographical Load Balancing over Cloud Networks	IEEE Transactions on Parallel and Distributed Systems (Volume 28, Issue 7, July 1	A systematic approach to designing energy-aware traffic-efficient geographical	1. Highly applicable for distributed data

TABLE 11. *(Continued.)* **Comparative analysis for problem formulation.**

	Tianyi Chen ; Antonio G. Marques ; Georgios B. Giannakis	2017)	load balancing scheme	centers 2. Security issues for location specific loads are ignored
[54]	An adaptive approach to better load balancing in a consumer-centric cloud environment Qi Liu ; Weidong Cai ; Jian Shen ; Xiaodong Liu ; Nigel Linge	IEEE Transactions on Consumer Electronics (Volume 62, Issue 3, August 2016)	Proposed an adaptive scheme is offered to achieve time and space efficiency in a heterogeneous cloud environment	1. The proposed space management is an improvement 2. The increased time complexity can be handled better
[55]	A Load-Balanced Call Admission Controller for IMS Cloud Computing Ahmadreza Montazerolghaem ; Mohammad Hossein Yaghmaee ; Alberto Leon-Garcia ; Mahmoud Naghibzadeh	IEEE Transactions on Network and Service Management (Volume 13, Issue 4, Dec. 2016)	A virtual load balanced call admission controller (VLB-CAC) for the cloud-hosted SIP servers	1. Improvement over agent based load balancing methods
[56]	A Heuristic Clustering-Based Task Deployment Approach for Load Balancing Using Bayes Theorem in Cloud Environment Jia Zhao ; Kun Yang ; Xiaohui Wei ; Yan Ding ; Liang Hu ; Gaochao Xu	IEEE Transactions on Parallel and Distributed Systems (Volume 27, Issue 2, Feb. 1 2016)	A heuristic approach called Load Balancing based on Bayes and Clustering (LB-BC)	1. The supervised classification of the loads is a unique method 2. Nevertheless, the lazy learning degradations must be overcome
[57]	A systematic review of nature inspired load balancing algorithm in heterogeneous cloud computing environment Pankaj Jain ; Sanjay Kumar Sharma	IEEE Information and Communication Technology (CICT), 2017	Surveyed the nature inspired load balancing algorithm such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), and Genetic Algorithm (GA), BAT	1. Survey work highlighting the advantages and bottlenecks of the works
[58]	A load balancing algorithm for virtual machines scheduling in cloud computing Li Liu ; Zhe Qiu ; Jie Dong	IEEE Modelling, Identification and Control (ICMIC), 2017	A model of virtual machine load based on task execution time span	1. This approach decreases the SLA violation 2. Energy awareness is low
[59]	Load Balancing of Nodes in Cloud Using Ant Colony Optimization Kumar Nishant ; Pratik Sharma ; Vishal Krishna ; Chhavi Gupta ; Kuwar Pratap Singh ; Nitin ; Ravi Rastogi	IEEE Computer Modelling and Simulation (UKSim), 2012	The ants continuously update a single result set rather than updating their own result set	1. Enhancement to the popular ACO 2. Further improvements can be addressed with adversarial comprehensive optimization
[60]	Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization Kun Li ; Gaochao Xu ; Guangyu Zhao ; Yushuang Dong ; Dan Wang	IEEE Chinagrid Conference (ChinaGrid), 2011	A cloud task scheduling policy based on Load Balancing Ant Colony Optimization (LBACO) algorithm	1. Enhancement to the popular ACO
[61]	Ant colony optimization for routing and load-balancing survey and new directions Kwang Mong Sim ; Weng Hong Sun	IEEE Transactions on Systems, Man, and Cybernetics - Part A Systems and Humans (Volume 33, Issue 5, Sept. 2003)	Surveying and comparing three major research in applying ACO in routing and load-balancing	1. Considering this paper as scope for the research paper
[62]	Cloud Computing Fitness for E-Government Implementation Importance-Performance Analysis	IEEE Access (Volume 6), 2017	Importance-performance matrix analysis	1. Importance of the performance evaluation matrix is

TABLE 11. ***(Continued.)*** **Comparative analysis for problem formulation.**

	Fathey Mohammed ; Ahmed Ibrahim Alzahrani ; Osama Alfarraj ; Othman Ibrahim			elaborated 2. Focused on the specific types of applications
[63]	Factors Influencing the Adoption of Cloud Computing by Small and Medium Size Enterprises (SMEs) Shima Ramezani TehraniFarid Shirazi	Springer, Human Interface and the Management of Information. Information and Knowledge in Applications and Services, 2016	Decision maker's knowledge about cloud computing is the main influential factor in decision making about its adoption	1. The demand from the enterprise applications cannot be fit into this proposal
[64]	Diffusion and Acceptance of Cloud Computing in SMEsTowards a Valence Model of Relevant Factors Mark Stieninger, Dietmar Nedbal	47th Hawaii International Conference on System Science, 2014	Individual, organizational, technical, and environmental factors influencing the diffusion and acceptance of Cloud Computing	1. Considering this paper as scope for the research paper
[65]	An analysis of the factors affecting the adoption of cloud consumer relationship management in the machinery industry in Taiwan Hsin-Pin Fu, Tsung-Sheng Chang	SAGE Journal, 2015	Fuzzy analytical hierarchy process (FAHP) was then used to calculate the weights of the factors, based on the returned questionnaires	1. Significantly connects to the proposed method in this work 2. Nonetheless, the enhancements of the weight measures must be realistic
[66]	Exploring the factors influencing the cloud computing adoption a systematic study on cloud migration Rashmi Rai, Gadadhar Sahoo and Shabana Mehfuz	Springer, 2015	Existing research work in the area of legacy to cloud migration	1. Important aspects of the recent research outcomes are discussed
[67]	QoS-Driven Power Management of Data Centers via Model Predictive Control Qiu Fang ; Jun Wang ; Qi Gong	IEEE Transactions on Automation Science and Engineering (Volume 13, Issue 4, Oct. 2016)	Formulate a constrained nonlinear optimal control problem to minimize the energy consumption of both information technology system and cooling technology system	1. Smart Power Grid techniques for overall power reduction 2. Cloud data centers component optimization is ignored 3. VM migration hot startup and shutdowns are reduced is ignored
[68]	Integrated Power Management of Data Centers and Electric Vehicles for Energy and Regulation Market Participation Sen Li ; Marco Brocanelli ; Wei Zhang ; Xiaorui Wang	IEEE Transactions on Smart Grid (Volume 5, Issue 5, Sept. 2014)	A systematic design of both the tracking control and market planning problems	
[69]	A Survey on Geographic Load Balancing Based Data Center Power Management in the Smart Grid Environment Ashikur Rahman ; Xue Liu ; Fanxin Kong	IEEE Communications Surveys & Tutorials (Volume 16, Issue 1, First Quarter 2014)	Survey on power management methodologies based on geographic load balancing (GLB) have recently been proposed to effectively utilize several features of smart grid	
[70]	Integrated Approach to Data Center Power Management Lakshmi Ganesh ; Hakim Weatherspoon ; Tudor Marian ; Ken Birman	IEEE Transactions on Computers (Volume 62, Issue 6, June 2013)	Algorithm which is reducing disk and server power consumption	
[71]	Data Center Peak Power Management with Energy Storage Devices Baris Aksanli	IEEE Internet Computing (Volume 21, Issue 4, 2017)	Compares ESDs based on their ability to manage data center peak power	

TABLE 12. Metric for number of hosts analysis.

Algorithms	Dataset wise Description									
	20110303	20110306	20110309	20110322	20110325	20110403	20110409	20110411	20110412	20110420
IQR MC	50	50	50	50	50	50	50	50	50	50
IQR MMT	50	50	50	50	50	50	50	50	50	50
IQR MU	50	50	50	50	50	50	50	50	50	50
IQR RS	50	50	50	50	50	50	50	50	50	50
LR MC	50	50	50	50	50	50	50	50	50	50
LR MMT	50	50	50	50	50	50	50	50	50	50
LR MU	50	50	50	50	50	50	50	50	50	50
LRR MC	50	50	50	50	50	50	50	50	50	50
LRR MMT	50	50	50	50	50	50	50	50	50	50
LRR MU	50	50	50	50	50	50	50	50	50	50
LRR RS	50	50	50	50	50	50	50	50	50	50
MAD MC	50	50	50	50	50	50	50	50	50	50
MAD MMT	50	50	50	50	50	50	50	50	50	50
MAD MU	50	50	50	50	50	50	50	50	50	50
MAD RS	50	50	50	50	50	50	50	50	50	50
THR MC	50	50	50	50	50	50	50	50	50	50
THR MMT	50	50	50	50	50	50	50	50	50	50
THR MU	50	50	50	50	50	50	50	50	50	50
THR RS	50	50	50	50	50	50	50	50	50	50

TABLE 13. Metric for number of VMs analysis.

Algorithms	Dataset wise Description									
	20110303	20110306	20110309	20110322	20110325	20110403	20110409	20110411	20110412	20110420
IQR MC	50	50	50	50	50	50	50	50	50	50
IQR MMT	50	50	50	50	50	50	50	50	50	50
IQR MU	50	50	50	50	50	50	50	50	50	50
IQR RS	50	50	50	50	50	50	50	50	50	50
LR MC	50	50	50	50	50	50	50	50	50	50
LR MMT	50	50	50	50	50	50	50	50	50	50
LR MU	50	50	50	50	50	50	50	50	50	50
LRR MC	50	50	50	50	50	50	50	50	50	50
LRR MMT	50	50	50	50	50	50	50	50	50	50
LRR MU	50	50	50	50	50	50	50	50	50	50
LRR RS	50	50	50	50	50	50	50	50	50	50
MAD MC	50	50	50	50	50	50	50	50	50	50
MAD MMT	50	50	50	50	50	50	50	50	50	50
MAD MU	50	50	50	50	50	50	50	50	50	50
MAD RS	50	50	50	50	50	50	50	50	50	50
THR MC	50	50	50	50	50	50	50	50	50	50
THR MMT	50	50	50	50	50	50	50	50	50	50
THR MU	50	50	50	50	50	50	50	50	50	50
THR RS	50	50	50	50	50	50	50	50	50	50

from the set of physical hosts (H[]) and can be formulated as,

$$DN = \prod_{Node\_Status=Dead} H[] \tag{42}$$

H. METRIC FOR MEAN TIME BEFORE A HOST SHUTDOWN (Sec)

The running time (T[]) for the physical host is the primary factor for calculating the mean time before host shutdown, denoted as TS, and can be formulated as,

$$TS = \frac{\sum_{i=1}^{n} T_i}{n} \tag{43}$$

Considering n is the total number of elements in set T[].

I. METRIC FOR MEAN TIME BEFORE A VM MIGRATION (Sec)

The mean time before a VM migration, denoted as TVM, is the combination of the mean time before a host shutdown (TS) and time to identify the loaded virtual machine (LT) and can be formulated as,

$$TVM = TS + LT \tag{44}$$

Or,

$$TVM = \frac{\sum_{i=1}^{n} T_i}{n} + LT \tag{45}$$

TABLE 14. Metric for total simulation time analysis.

Algorithms	Dataset wise Description									
	20110303	20110306	20110309	20110322	20110325	20110403	20110409	20110411	20110412	20110420
IQR MC	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
IQR MMT	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
IQR MU	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
IQR RS	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
LR MC	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
LR MMT	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
LR MU	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
LRR MC	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
LRR MMT	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
LRR MU	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
LRR RS	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
MAD MC	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
MAD MMT	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
MAD MU	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
MAD RS	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
THR MC	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
THR MMT	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
THR MU	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400
THR RS	86400	86400	86400	86400	86400	86400	86400	86400	86400	86400

TABLE 15. Metric for energy consumption analysis.

Algorithms	Dataset wise Description									
	20110303	20110306	20110309	20110322	20110325	20110403	20110409	20110411	20110412	20110420
IQR MC	46.86	46.86	46.86	46.86	46.86	46.86	46.86	46.86	46.86	46.86
IQR MMT	47.85	47.85	47.85	47.85	47.85	47.85	47.85	47.85	47.85	47.85
IQR MU	49.32	49.32	49.32	49.32	49.32	49.32	49.32	49.32	49.32	49.32
IQR RS	47.17	47.3	47.12	46.94	47	47.05	46.62	46.97	47.01	47.49
LR MC	34.35	34.35	34.35	34.35	34.35	34.35	34.35	34.35	34.35	34.35
LR MMT	35.37	35.37	35.37	35.37	35.37	35.37	35.37	35.37	35.37	35.37
LR MU	35.38	35.38	35.38	35.38	35.38	35.38	35.38	35.38	35.38	35.38
LRR MC	34.35	34.35	34.35	34.35	34.35	34.35	34.35	34.35	34.35	34.35
LRR MMT	35.37	35.37	35.37	35.37	35.37	35.37	35.37	35.37	35.37	35.37
LRR MU	35.38	35.38	35.38	35.38	35.38	35.38	35.38	35.38	35.38	35.38
LRR RS	34.42	34.12	34.58	34.23	34.23	34.32	34.03	34.27	33.98	34.4
MAD MC	44.99	44.99	44.99	44.99	44.99	44.99	44.99	44.99	44.99	44.99
MAD MMT	45.61	45.61	45.61	45.61	45.61	45.61	45.61	45.61	45.61	45.61
MAD MU	47.36	47.36	47.36	47.36	47.36	47.36	47.36	47.36	47.36	47.36
MAD RS	44.38	44.95	45.1	44.88	44.74	44.89	44.74	44.76	44.99	44.79
THR MC	40.85	40.85	40.85	40.85	40.85	36.81	40.85	40.85	40.85	40.85
THR MMT	41.81	41.81	41.81	41.81	41.81	40.85	41.81	41.81	41.81	41.81
THR MU	44.08	44.08	44.08	44.08	44.08	41.81	44.08	44.08	44.08	44.08
THR RS	41.17	40.92	41.26	41.59	41.44	44.08	41.15	41.12	41.08	41.41

J. METRIC FOR EXECUTION TIME (Sec)

The execution time for any application submitted to the data center can be calculated as the submission time (SubT) of the job or the task and the completion time (ComT) of the tasks. Regardless to mention, the implications of the simulation time scaling factor, as demonstrated in Eq. 1.38, must be considered to calculate the actual time for execution of the tasks. The calculation of the execution Time, T, can be furnished as,

$$T = ComT.S - SubT.S \tag{46}$$

In the further section of this chapter the comparative analysis discussions for each algorithm type on each listed dataset are performed and analyzed.

X. OPEN CHALLENGES AND BOTTLENECKS IN VM MIGRATION FOR LOAD BALANCING

The never-ending research in the field of virtual machine migrations for load balancing motivates number of researchers to produce, significant works in this research fields. To formulate the problem for the research, this work again considers few more parallel researches outcomes, where the bottleneck is clearly observable [Table -11].

With the detailed understanding of the existing methods for load balancing using virtual machine migrations, the following short coming are identified and the proposed set of solutions.

In this section of the work, the demand or the scopes for the research is identified.

TABLE 16. Metric for VM migration analysis.

Algorithms	Dataset wise Description									
	20110303	20110306	20110309	20110322	20110325	20110403	20110409	20110411	20110412	20110420
IQR MC	5085	5085	5085	5085	5085	5085	5085	5085	5085	5085
IQR MMT	5502	5502	5502	5502	5502	5502	5502	5502	5502	5502
IQR MU	5789	5789	5789	5789	5789	5789	5789	5789	5789	5789
IQR RS	5043	5057	5093	5066	5134	4958	4973	5135	5036	5013
LR MC	2203	2203	2203	2203	2203	2203	2203	2203	2203	2203
LR MMT	2872	2872	2872	2872	2872	2872	2872	2872	2872	2872
LR MU	2808	2808	2808	2808	2808	2808	2808	2808	2808	2808
LRR MC	2203	2203	2203	2203	2203	2203	2203	2203	2203	2203
LRR MMT	2872	2872	2872	2872	2872	2872	2872	2872	2872	2872
LRR MU	2808	2808	2808	2808	2808	2808	2808	2808	2808	2808
LRR RS	2353	2322	2499	2283	2292	2363	2244	2375	2228	2363
MAD MC	4778	4778	4778	4778	4778	4778	4778	4778	4778	4778
MAD MMT	5265	5265	5265	5265	5265	5265	5265	5265	5265	5265
MAD MU	5628	5628	5628	5628	5628	5628	5628	5628	5628	5628
MAD RS	4697	4873	4979	4862	4816	4854	4824	4746	4897	4704
THR MC	4392	4392	4392	4392	4392	2638	4392	4392	4392	4392
THR MMT	4839	4839	4839	4839	4839	4392	4839	4839	4839	4839
THR MU	5404	5404	5404	5404	5404	4839	5404	5404	5404	5404
THR RS	4482	4266	4395	4578	4424	5404	4500	4439	4571	4537

TABLE 17. Metric for SLA degradation analysis.

Algorithms	Dataset wise Description									
	20110303	20110306	20110309	20110322	20110325	20110403	20110409	20110411	20110412	20110420
IQR MC	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
IQR MMT	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23
IQR MU	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
IQR RS	0.25	0.25	0.25	0.26	0.26	0.25	0.25	0.25	0.25	0.25
LR MC	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
LR MMT	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13
LR MU	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13
LRR MC	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
LRR MMT	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13
LRR MU	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13
LRR RS	0.14	0.15	0.15	0.14	0.14	0.14	0.14	0.14	0.14	0.14
MAD MC	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
MAD MMT	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23
MAD MU	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
MAD RS	0.25	0.25	0.26	0.26	0.25	0.26	0.26	0.25	0.26	0.25
THR MC	0.27	0.27	0.27	0.27	0.27	0.15	0.27	0.27	0.27	0.27
THR MMT	0.23	0.23	0.23	0.23	0.23	0.27	0.23	0.23	0.23	0.23
THR MU	0.28	0.28	0.28	0.28	0.28	0.23	0.28	0.28	0.28	0.28
THR RS	0.27	0.26	0.26	0.28	0.27	0.28	0.27	0.27	0.27	0.28

- Firstly, most of the proposed researches are focused on the specific service models. Hence a wide range of service models needs to be considered in order to prove the performance enhancements.
- Secondly, the Virtual Machine based implementation are demonstrated in parallel research outcomes, however the power demands of the virtual machines are increasing, which needs to be addressed.
- Third, the most ignored fragment in the parallel researches are the power management and power aware algorithms not considering the smart power grid into the architecture to improve the power consumption.
- Fourthly, there are a wide range of researches are being executed focusing on specific domain data like medical domain data or tactical networks. The generic characteristics of the services may lead to the situation in variation of load balances.
- Fifth, the implementations of the parallel researches majorly concentrate on the public cloud-based load balancing. Nevertheless, the implantations of the load balancing algorithms will demonstrate the different outcomes in case of hybrid cloud.

Finally, none of the parallel researches demonstrates the use of predictive performance evaluation matrix using machine learning for enhancement.

XI. EXPERIMENTAL ANALYSIS AND DISCUSSION

This section of the chapter discussed the performance of the existing load balancing strategies over the performance comparison metrics as discussed in the earlier section of this work.

A. METRIC FOR NUMBER OF HOSTS ANALYSIS

The number of hosts analysis is presented here [Table – 12].

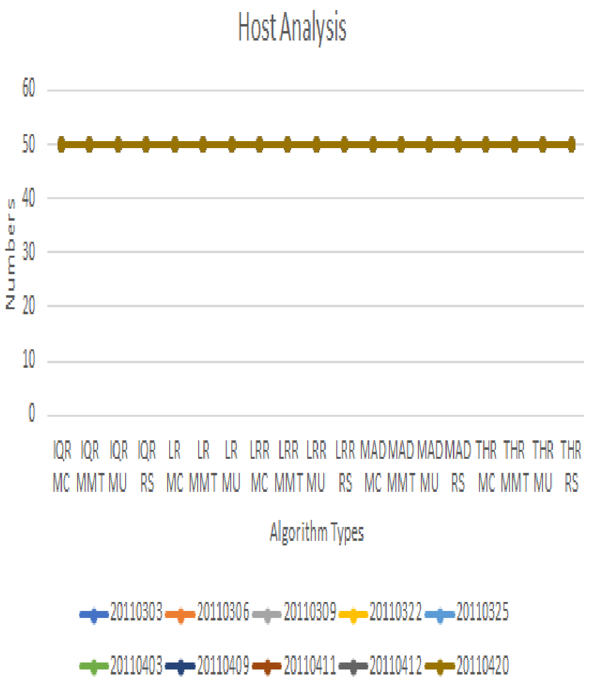


FIGURE 6. Host metric analysis.

The result is visualized graphically here [Fig – 6].

The initial condition for the experiments with all the algorithms and on each dataset, must be similar and thus the initial numbers of physical host nodes on the datacenter are same.

B. METRIC FOR NUMBER OF VMs ANALYSIS

The number of VM analysis is presented here [Table – 13] and the result is visualized graphically here [Fig – 7].

The initial condition for the experiments with all the algorithms and on each dataset, must be similar and thus the initial numbers of machines on each physical node of the datacenter are same.

C. METRIC FOR TOTAL SIMULATION TIME ANALYSIS

The number of total simulation time analysis is presented here [Table – 14] and the result is visualized graphically here [Fig – 8].

The total simulation time is the measure of the task completion time for each algorithm type. This measure is considered under similar situations and thus, provides a standard comparison of the algorithms to be called as most time efficient, moderately time efficient and finally least time efficient.

The duration of the simulation is one of the most conclusive factors on various metrics such as energy consumption or the number of virtual machines been migrated or number of physical hosts to be shutdown, hence for a neutral analysis, the simulation time for all the experiments are kept similar, though few of the algorithms demonstrates static behavior after a specific point of time.

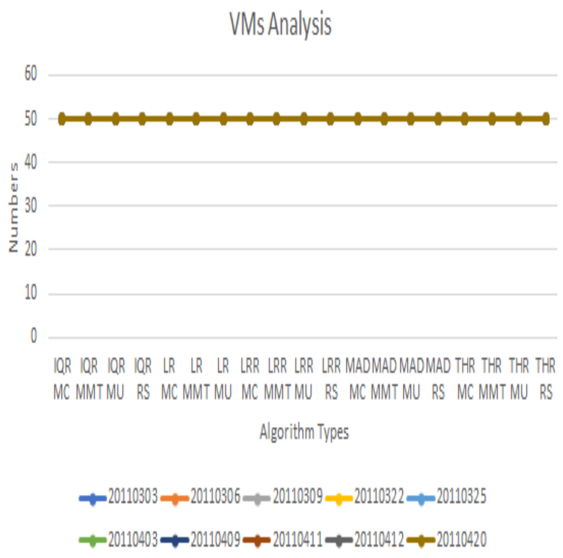


FIGURE 7. VM metric analysis.

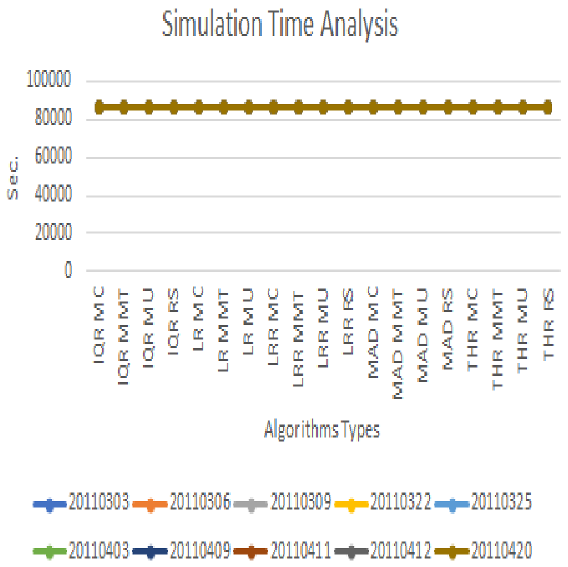


FIGURE 8. Simulation time metric analysis.

D. METRIC FOR ENERGY CONSUMPTION (kWh) ANALYSIS

The Energy Consumption Analysis is presented here [Table – 15] and the result is visualized graphically here [Fig – 9].

The performance of robust local regression method for threshold determination and minimum migration time for the VM selection performs the best under this metric analysis. The elaborated reason is furnished in the previous section of this work with detailed mathematical analysis of the algorithm types.

E. METRIC FOR NUMBER OF VM MIGRATIONS ANALYSIS

In this section of the work, the number of VM migration analysis is carried out [Table – 16] and the result is visualized graphically here [Fig – 10].

TABLE 18. Metric for node shutdown analysis.

Algorithms	Dataset wise Description									
	20110303	20110306	20110309	20110322	20110325	20110403	20110409	20110411	20110412	20110420
IQR MC	1517	1517	1517	1517	1517	1517	1517	1517	1517	1517
IQR MMT	1549	1549	1549	1549	1549	1549	1549	1549	1549	1549
IQR MU	1622	1622	1622	1622	1622	1622	1622	1622	1622	1622
IQR RS	1497	1500	1504	1519	1518	1499	1467	1523	1506	1493
LR MC	685	685	685	685	685	685	685	685	685	685
LR MMT	806	806	806	806	806	806	806	806	806	806
LR MU	816	816	816	816	816	816	816	816	816	816
LRR MC	685	685	685	685	685	685	685	685	685	685
LRR MMT	806	806	806	806	806	806	806	806	806	806
LRR MU	816	816	816	816	816	816	816	816	816	816
LRR RS	704	703	740	690	704	707	686	708	677	721
MAD MC	1468	1468	1468	1468	1468	1468	1468	1468	1468	1468
MAD MMT	1528	1528	1528	1528	1528	1528	1528	1528	1528	1528
MAD MU	1632	1632	1632	1632	1632	1632	1632	1632	1632	1632
MAD RS	1453	1481	1487	1486	1456	1486	1467	1466	1474	1448
THR MC	1389	1389	1389	1389	1389	795	1389	1389	1389	1389
THR MMT	1424	1424	1424	1424	1424	1389	1424	1424	1424	1424
THR MU	1578	1578	1578	1578	1578	1424	1578	1578	1578	1578
THR RS	1381	1345	1378	1420	1390	1578	1375	1372	1390	1402

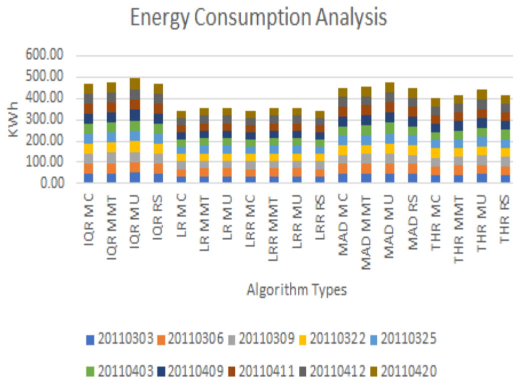


FIGURE 9. Energy consumption metric analysis.

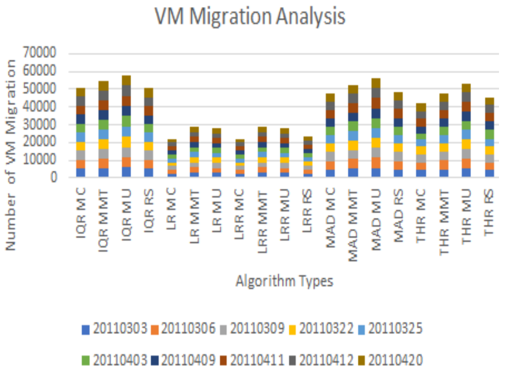


FIGURE 10. VM migration metric analysis.

The performance of local regression method for threshold determination and maximum correlation for the VM selection performs the best under this metric analysis. The elaborated reason is furnished in the previous section of this work with detailed mathematical analysis of the algorithm types.

F. METRIC FOR SLA PERF DEGRADATION DUE TO MIGRATION ANALYSIS

This section of the work analyses the SLA performance degradation for various algorithms under various datasets [Table – 17] and the result is visualized graphically here [Fig – 11].

A service-level agreement is a dedication between a service supplier and a customer. Aspects of the service – quality, availability, duties – are agreed between the service supplier and the service client.

The performance measures of local regression method for threshold determination and minimum migration time for the VM selection performs the best under this metric analysis.

G. METRIC FOR NUMBER OF NODE SHUTDOWNS ANALYSIS

The analysis for physical node shutdown is presented here [Table – 18] and the result is visualized graphically here [Fig – 12].

The performance of local regression method for threshold determination and maximum correlation for the VM selection performs the best.

H. METRIC FOR MEAN TIME BEFORE A HOST SHUTDOWN (Sec) ANALYSIS

The analysis for mean time before a host shutdown is presented here [Table – 19].

The result is visualized graphically here [Fig – 13].

TABLE 19. Metric for mean time before VM migration analysis.

Algorithms	Dataset wise Description									
	20110303	20110306	20110309	20110322	20110325	20110403	20110409	20110411	20110412	20110420
IQR MC	20.33	20.33	20.33	20.33	20.33	20.33	20.33	20.33	20.33	20.33
IQR MMT	17.62	17.62	17.62	17.62	17.62	17.62	17.62	17.62	17.62	17.62
IQR MU	20.38	20.38	20.38	20.38	20.38	20.38	20.38	20.38	20.38	20.38
IQR RS	20.3	20.15	20.33	20.2	20.24	20.24	20.14	20.34	20.27	20.18
LR MC	20.35	20.35	20.35	20.35	20.35	20.35	20.35	20.35	20.35	20.35
LR MMT	16.6	16.6	16.6	16.6	16.6	16.6	16.6	16.6	16.6	16.6
LR MU	20.06	20.06	20.06	20.06	20.06	20.06	20.06	20.06	20.06	20.06
LRR MC	20.35	20.35	20.35	20.35	20.35	20.35	20.35	20.35	20.35	20.35
LRR MMT	16.6	16.6	16.6	16.6	16.6	16.6	16.6	16.6	16.6	16.6
LRR MU	20.06	20.06	20.06	20.06	20.06	20.06	20.06	20.06	20.06	20.06
LRR RS	20.17	20.38	20.34	20.37	20.36	20.36	20.19	20.34	20.12	20.51
MAD MC	20.35	20.35	20.35	20.35	20.35	20.35	20.35	20.35	20.35	20.35
MAD MMT	17.17	17.17	17.17	17.17	17.17	17.17	17.17	17.17	17.17	17.17
MAD MU	20.18	20.18	20.18	20.18	20.18	20.18	20.18	20.18	20.18	20.18
MAD RS	20.17	20.27	20.41	20.26	20.38	20.38	20.38	20.43	20.21	20.26
THR MC	20.47	20.47	20.47	20.47	20.47	20.47	20.47	20.47	20.47	20.47
THR MMT	16.82	16.82	16.82	16.82	16.82	16.82	16.82	16.82	16.82	16.82
THR MU	20.23	20.23	20.23	20.23	20.23	20.23	20.23	20.23	20.23	20.23
THR RS	20.16	20.34	20.27	20.31	20.42	20.42	20.36	20.19	20.23	20.35

TABLE 20. Metric for execution/task completion analysis.

Algorithms	Dataset wise Description									
	20110303	20110306	20110309	20110322	20110325	20110403	20110409	20110411	20110412	20110420
IQR MC	0.00309	0.00243	0.00246	0.00245	0.00244	0.00246	0.00249	0.00246	0.00243	0.00247
IQR MMT	0.00251	0.00238	0.00251	0.00249	0.00241	0.00241	0.00248	0.00251	0.00243	0.00245
IQR MU	0.00278	0.00258	0.00264	0.00262	0.00272	0.00256	0.00267	0.00268	0.00259	0.00262
IQR RS	0.00234	0.00222	0.00228	0.00224	0.00233	0.00218	0.00231	0.00228	0.00223	0.00219
LR MC	0.00169	0.00183	0.00184	0.00189	0.00189	0.00188	0.00193	0.0019	0.00192	0.00184
LR MMT	0.00162	0.00209	0.00206	0.00201	0.00205	0.00202	0.00205	0.00207	0.00208	0.00202
LR MU	0.00163	0.00203	0.00202	0.00206	0.00196	0.00201	0.00202	0.00202	0.00203	0.00198
LRR MC	0.00194	0.00162	0.00161	0.00159	0.00163	0.00158	0.00162	0.00162	0.00162	0.00162
LRR MMT	0.00202	0.00168	0.00167	0.00168	0.00167	0.00167	0.00167	0.00172	0.00168	0.00178
LRR MU	0.00197	0.00174	0.00168	0.00166	0.0016	0.00163	0.00164	0.00161	0.0017	0.0016
LRR RS	0.00175	0.00148	0.00153	0.00148	0.0015	0.00149	0.00152	0.00153	0.00149	0.00144
MAD MC	0.00296	0.00294	0.00296	0.00299	0.00301	0.00298	0.00289	0.00297	0.00293	0.00295
MAD MMT	0.00308	0.00296	0.00304	0.00318	0.00301	0.00306	0.0031	0.00299	0.00309	0.00308
MAD MU	0.00338	0.00331	0.00339	0.00332	0.0033	0.00334	0.00337	0.0034	0.00327	0.00335
MAD RS	0.00277	0.00282	0.0028	0.00275	0.00279	0.00276	0.00276	0.00267	0.0028	0.00277
THR MC	0.00176	0.00164	0.0016	0.00159	0.00176	0.00156	0.00168	0.00164	0.00162	0.00162
THR MMT	0.00155	0.00163	0.00145	0.00152	0.00149	0.00163	0.00153	0.00147	0.00157	0.00149
THR MU	0.00182	0.00169	0.00171	0.00168	0.00168	0.00148	0.00175	0.00178	0.00171	0.00165
THR RS	0.00134	0.00141	0.00138	0.00145	0.0014	0.00178	0.00145	0.0014	0.00141	0.00143

The outcomes from the Median Absolute Deviation method for threshold determination and minimum utilization for the VM selection achieves the best results for host shutdown scenarios.

I. METRIC FOR MEAN TIME BEFORE A VM MIGRATION (Sec) ANALYSIS

The analysis for mean time before a VM migration is presented here [Table – 20].

Post-duplicate VM migration is started by suspending the VM at the source. With the VM suspended, an insignificant subset of the execution condition of the VM is exchanged to the objective. The VM is then continued at the objective. Simultaneously, the source effectively pushes the rest of the memory pages of the VM to the objective - an action known as pre-paging. At the objective, if the VM endeavors to get to a page that has not yet been exchanged, it creates a page-shortcoming. These shortcomings, known as system

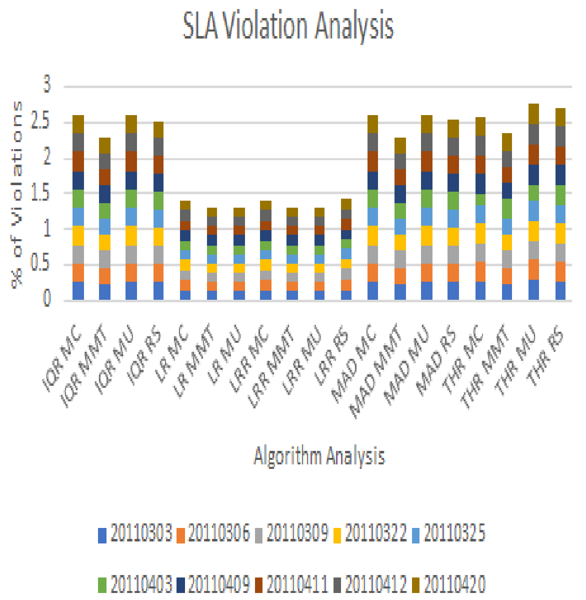


FIGURE 11. SLA violation metric analysis.

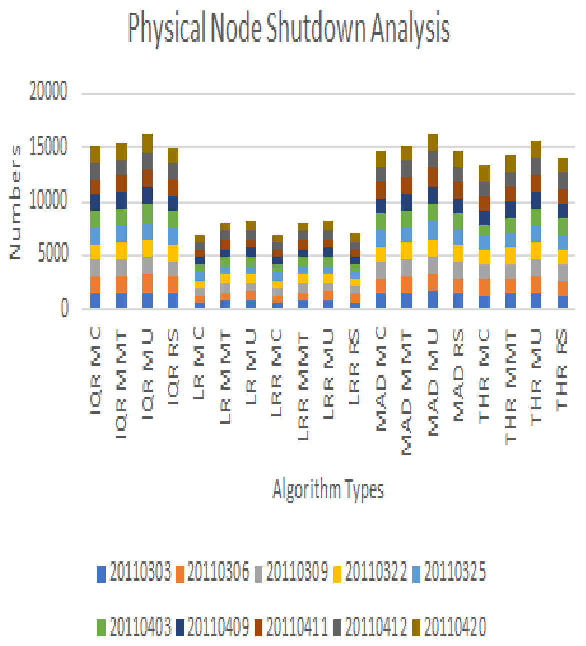


FIGURE 12. Node shutdown metric analysis.

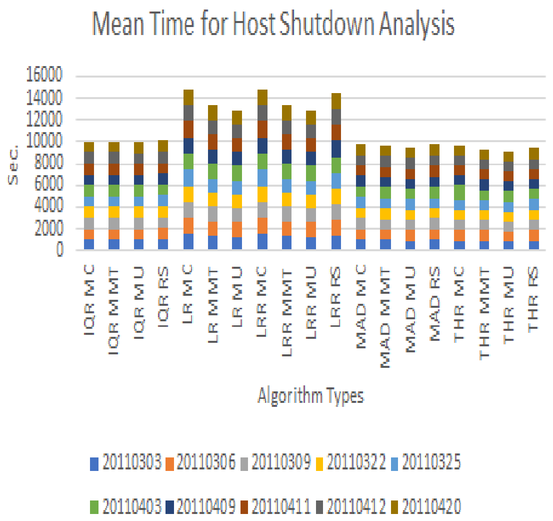


FIGURE 13. Mean time for host shutdown metric analysis.

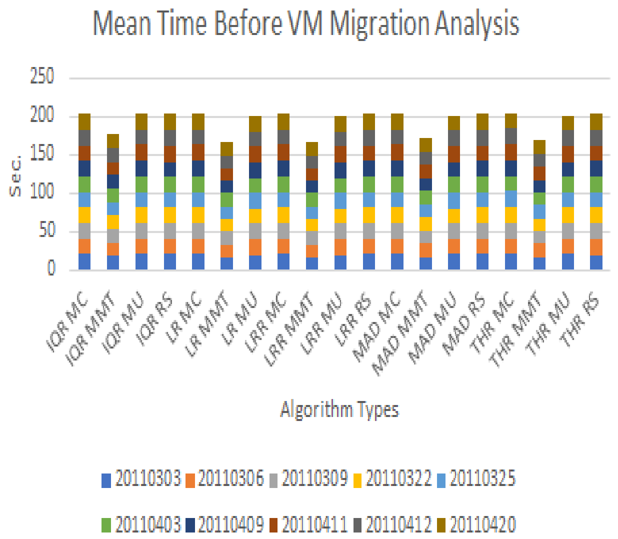


FIGURE 14. Mean time before VM migration metric analysis.

flaws, are caught at the objective and diverted to the source, which reacts with the blamed page. Too many system issues can debase execution of utilizations running inside the VM. Consequently, pre-paging can progressively adjust the page transmission request to arrange blames by effectively pushing pages in the region of the last flaw. A perfect pre-paging plan would veil vast greater part of system deficiencies, in spite of the fact that its exhibition relies on the memory get to example of the VM's remaining burden. Post-duplicate sends each page precisely once over the system. Conversely, pre-duplicate can exchange a similar page on different occasions if the page is dirtied over and again at the source amid migration.

The result is visualized graphically here [Fig – 14].

Performance of local regression method for threshold determination and minimum migration time for the VM selection produces the finest outcomes for the VM migrations and reduces the time complexity to a significant extend.

J. METRIC FOR EXECUTION TIME (Sec) ANALYSIS

The final section of the experimental analysis presents the job or task completion time for the analyzed algorithms [Table – 21].

The result is visualized graphically here [Fig – 15].

The results for the robust local regression method for threshold determination and random selection for the virtual

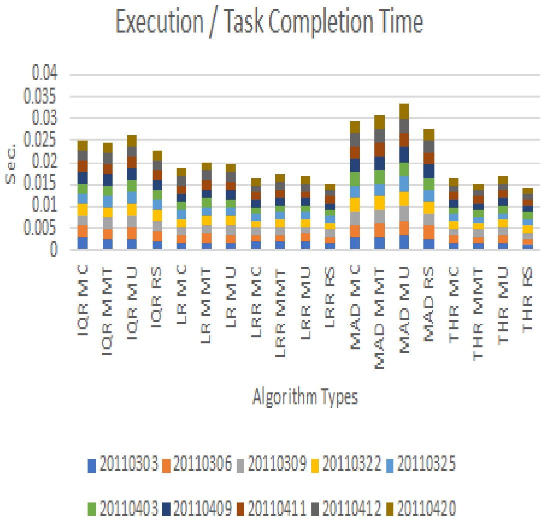


FIGURE 15. **Execution/task completion time metric analysis.**

machines demonstrates the best performance and significantly reduces the time complexity for execution or task completion times with better satisfiability of the SLA.

XII. CONCLUSION

In the due course of the literature survey, the observation is to that the initial research attempts where been made on the Hadoop framework and the initial strategies were showcased as FIFO, FAIR, Capacity, Hybrid, LATE, SAMR and Context aware allocation of tasks or jobs. This work analyzes the characteristics of these algorithms with parametric metric. Further, realizing the demand to migrate the load balancing strategies to the cloud, this work analyses the Threshold Detection Policy as Inter Quartile Range, Local Regression, Median Absolute Deviation, Robust Local Regression and Static Threshold. In the same continuation, VM Consolidation Policies are also analyzed as Maximum Correlation, Minimum Migration Time, Minimum Utilization and Random Selection. The mathematical analysis demonstrates the advantages and bottlenecks of each algorithm categories. The work is intended to analyses the possible effects of these algorithm type combinations on different datasets, hence nearly 10 datasets are analyzed using Number of Traces, Location Information, Event Id, Node Id, Fault Code and Message as parameters for comparisons. Furthermore, this work deploys a standard metric for performance comparisons of the standard algorithms and presents the experimental results on various datasets using few metrics as Number of hosts, Number of VMs, Total simulation time, Energy consumption, Number of VM migrations, SLA degradation due to migration, Number of node shutdowns, mean time before a host shutdown, Mean time before a VM migration and Execution time. Finally, this work concludes few of the major bottlenecks to be taken under the further research attempts as Firstly, demonstration of cloud-based service type variations on Hadoop based multiple queue load balancing. Secondly, identification of the existing optimal load balancing matching for specific application types. Thirdly, proposing a novel generative adversarial method combining the genetic algorithms for load balancing. Fourthly, proposing a novel predictive performance evaluation metric for migrating applications and finally, this study concludes, that the performance of robust local regression method for threshold determination and minimum migration time for the VM selection are proven to be the best for energy conservation, the number of migration of the virtual machines are least for the local regression method for threshold determination and maximum correlation for the VM selection, the service level agreement or SLA can be best satisfiable with the performance measures of local regression method for threshold determination and minimum migration time for the VM selection, the number of physical host shutdowns can increase the effect on the environment and reboot process as the lease can be noted for local regression method for threshold determination and maximum correlation for the VM selection, Median Absolute Deviation method for threshold determination and minimum utilization for the VM selection achieves the best results for host shutdown scenarios, local regression method for threshold determination and minimum migration time for the VM selection produces the finest outcomes for the VM migrations and reduces the time complexity to a significant extend and the robust local regression method for threshold determination and random selection for the virtual machines demonstrates the best performance and significantly reduces the time complexity for execution or task completion times with better satisfiability of the SLA.

REFERENCES

[1] A. Wylie, W. Shi, J. P. Corriveau, and Y. Wang, "A scheduling algorithm for Hadoop mapreduce workflows with budget constraints in the heterogeneous cloud," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, May 2016, pp. 1433–1442.

[2] J. Dean and S. Ghemawat, "Mapreduce simplified data processing on large clusters," *Oper. Syst. Des. Implement.*, vol. 51, no. 1, pp. 107–113, 2013.

[3] S. T. Leung, S. T. Leung, and S. T. Leung, "The google file system," in *Proc. 19th ACM Symp. Oper. Syst. Principles*, 2003, pp. 29–43.

[4] J. Tan, X. Meng, and L. Zhang, "Coupling task progress for mapreduce resource-aware scheduling," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1618–1626.

[5] H. Mao, S. Hu, Z. Zhang, L. Xiao, and L. Ruan, "A load-driven task scheduler with adaptive DSC for mapreduce," in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun.*, Aug. 2011, pp. 28–33.

[6] M. Konar, R. Evans, T. Graves, E. Baldeschwieler, and E. Baldeschwieler, "Apache Hadoop yarn yet another resource negotiator," in *Proc. Symp. Cloud Comput.*, 2013, p. 5.

[7] Y. Liu, Y. Zeng, and X. Piao, "High-responsive scheduling with mapreduce performance prediction on Hadoop YARN," in *Proc. IEEE Int. Conf. Embedded Real Time Comput. Syst. Appl.*, Aug. 2016, pp. 238–247.

[8] P. Shu-Jun, Z. Xi-Min, H. Da-Ming, L. Shu-Hui, and Z. Yuan-Xu, "Optimization and research of Hadoop platform based on fifo scheduler," in *Proc. Int. Conf. Measuring Technol. Mechatronics Autom.*, 2015, pp. 727–730.

[9] M. Yong, N. Garegrat, and S. Mohan, "Towards a resource aware scheduler in Hadoop," Comput. Sci. Eng., Univ. Michigan, Ann Arbor, MI, USA, Dec. 2009.

[10] J. S. Sarma, J. S. Sarma, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling a simple technique for achieving locality and fairness in cluster scheduling," in *Proc. Eur. Conf. Comput. Syst.*, 2010, pp. 265–278.

[11] X. Dai and B. Bensaou, "Scheduling for response time in Hadoop mapreduce," in *Proc. IEEE Int. Conf. Commun.*, May 2016, pp. 1–6.

[12] A. M. Elkholy and E. A. H. Sallam, "Self adaptive Hadoop scheduler for heterogeneous resources," in *Proc. Int. Conf. Comput. Eng. Syst.*, 2015, pp. 427–432.

[13] D. Nayak, V. S. Martha, D. Threm, and S. Ramaswamy, "Adaptive scheduling in the cloud SLA for Hadoop job scheduling," in *Proc. Sci. Inf. Conf.*, 2015, pp. 832–837.

[14] X. Xu, L. Cao, and X. Wang, "Adaptive task scheduling strategy based on dynamic workload adjustment for heterogeneous Hadoop clusters," *IEEE Syst. J.*, vol. 10, no. 2, pp. 471–482, Jun. 2017.

[15] G. K. Archana and V. D. Chakravarthy, "HPCA A node selection and scheduling method for Hadoop mapreduce," in *Proc. Int. Conf. Comput. Commun. Technol.*, 2015, pp. 368–372.

[16] Y. Mao, H. Zhong, and L. Wang, "A fine-grained and dynamic mapreduce task scheduling scheme for the heterogeneous cloud environment," in *Proc. Int. Symp. Distrib. Comput. Appl. Bus. Eng. Sci.*, 2015, pp. 155–158.

[17] Y. Yao, J. Tai, B. Sheng, and N. Mi, "LsPS: A job size-based scheduler for efficient task assignments in Hadoop," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 411–424, Oct./Dec. 2015.

[18] B. Moseley, A. Dasgupta, and R. Kumar, "On scheduling in map-reduce and flow-shops," in *Proc. ACM Symp. Parallelism Algorithms Architectures*, 2011, pp. 289–298.

[19] *Hadoop*. Accessed: 2019. [Online]. Available: https//hadoop.apache.org/

[20] *Hadoop's Fair Scheduler*. Accessed: 2019. [Online]. Available: https//hadoop.apache.org/docs/r1.2.1/fair_scheduler

[21] B. P. Andrews and A. Binu, "Survey on job schedulers in Hadoop cluster," *IOSR J. Comput. Eng.*, vol. 15, no. 1, pp. 46–50, Sep./Oct. 2013.

[22] J. Chen, D. Wang, and W. Zhao, "A task scheduling algorithm for Hadoop platform," *J. Comput.*, vol. 8, no. 4, pp. 929–936, Apr. 2013.

[23] N. Tiwari, "Scheduling and energy efficiency improvement techniques for Hadoop mapreduce state of art and directions for future research," Ph.D. dissertation, Dept. Comput. Sci. Eng., Indian Inst. Technol., Mumbai, India.

[24] P. Nguyen, T. Simon, M. Halem, D. Chapman, and Q. Le, "A hybrid scheduling algorithm for data intensive workloads in aMapReduce environment," in *Proc. IEEE/ACM 5th Int. Conf. Utility Cloud Comput. IEEE Comput. Soc. (UCC)*, Washington, DC, USA, Nov. 2012, pp. 161–168.

[25] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," in *Proc. OSDI 8th USENIX Symp. Oper. Syst. Design Implement.*, 2008, pp. 29–42.

[26] Q. Chen, D. Zhang, M. Guo, Q. Q. Deng, and S. Guo, "SAMR: A self-adaptive MapReduce scheduling algorithm in heterogeneous environment," in *Proc. 10th Int. Conf. Comput. Inf. Technol.*, Jul. 2010, pp. 43–2736.

[27] K. A. Kumar, V. K. Konishetty, K. Voruganti, and G. Rao, "CASH context aware scheduler for Hadoop," in *Proc. Int. Conf. Adv. Comput., Commun. Inform.*, New York, NY, USA, 2012, pp. 52–61.

[28] E. Feller and C. Morin, "A case for fully decentralized dynamic VM consolidation in clouds," in *Proc. 4th IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Taipei, Taiwan, Dec. 2012, pp. 26–33.

[29] M. Marzolla and O. Babaoglu, "Server consolidation in Clouds through gossiping," in *Proc. 12th IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, Lucca, Italy, Jun. 2011, pp. 1–6.

[30] A. Murtazaev and S. Oh, "Sercon server consolidation algorithm using live migration of virtual machines for green computing," *IETE Tech. Rev.*, vol. 28, no. 3, pp. 212–231, 2011.

[31] W. Vogels, "Beyond server consolidation," *ACM Queue*, vol. 6, no. 1, pp. 20–26, 2008.

[32] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[33] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Comput., Pract. Exper.*, vol. 24, no. 13, pp. 1397–1420, Sep. 2012.

[34] F. Farahnakian and P. Liljeberg, "Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers," in *Proc. 39th Euromicro Conf. Softw. Eng. Adv. Appl.*, Santander, Spain, 2013, pp. 357–364.

[35] F. Farahnakian, T. Pahikkala, P. Liljeberg, and J. Plosila, "Energy aware consolidation algorithm based on K-nearest neighbor regression for cloud data centers," in *Proc. 6th IEEE/ACM Int. Conf. Utility Cloud Comput.*, Dresden, Germany, Dec. 2013, pp. 256–259.

[36] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *Proc. Int. Conf. Comput. Meas. Group*, San Diego, CA, USA, 2007, pp. 399–407.

[37] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. 30th IEEE Int. Conf. Comput. Commun.*, Shanghai, China, Apr. 2011, pp. 71–75.

[38] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Comput. Netw.*, vol. 53, no. 7, pp. 2923–2938, 2009.

[39] M. Dorigo and L. Gambardella, "Ant colony system a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[40] M. Dorigo, G. Caro, and L. Gambardella, "Ant algorithms for discrete optimization," *Artif. Life*, vol. 5, no. 2, pp. 137–172, Apr. 1999.

[41] M. Harman, K. Lakhotia, J. Singer, D. R. White, and S. Yoo, "Cloud engineering is search based software engineering too," *J. Syst. Softw.*, vol. 86, no. 9, pp. 2225–2241, 2013.

[42] P. R. Theja and S. K. Babu, "An evolutionary computing based energy efficient VM consolidation scheme for optimal resource utilization and QoS assurance," *Indian J. Sci. Technol.*, vol. 8, no. 26, pp. 1–11, 2015.

[43] P. R. Theja and S. K. Babu, "An adaptive genetic algorithm based robust QoS oriented green computing scheme for VM consolidation in large scale cloud infrastructures," *Indian J. Sci. Technol.*, vol. 8, no. 27, pp. 1–13, 2015.

[44] P. R. Theja and S. K. Babu, "Evolutionary computing based on QoS oriented energy efficient VM consolidation scheme for large scale cloud data centers," *Cybern. Inf. Technol.*, vol. 16, no. 2, pp. 97–112, 2016.

[45] S. Heidari and R. Buyya, "Quality of service (QoS)-driven resource provisioning for large-scale graph processing in cloud computing environments graph processing-as-a-service (GPaaS)," in *Future Generation Computer Systems (FGCS)*, vol. 96. Amsterdam, The Netherlands: Elsevier, Jul. 2019, pp. 490–501.

[46] S. Sevinc. *PlanetLab Data Sets*. Accessed: 2019. [Online]. Available: https//www.planet-lab.org/datasets

[47] N. M. F. Qureshi, D. R. Shin, I. F. Siddiqui, and B. S. Chowdhry, "Storage-tag-aware scheduler for Hadoop cluster," *IEEE Access*, vol. 5, pp. 13742–13755, Jul. 2017.

[48] F. Bajaber, R. Elshawi, O. Batarfi, A. Altalhi, A. Barnawi, and S. Sakr, "Big data 2.0 processing systems: Taxonomy and open challenges," *J. Grid Comput.*, vol. 14, no. 3, pp. 379–405, Jun. 2016.

[49] J. B. Mathiya and L. V. Desai, "Apache Hadoop yarn parameter configuration challenges and optimization," in *Proc. ICSNS*, Oct. 2015, pp. 1–6.

[50] S. Singh, G. Rakhi, and P. K. Mishra, *Review of Apriori Based Algorithms on MapReduce Framework*. Ithaca, NY, USA: Cornell Univ. Press, Feb. 2017.

[51] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing taxonomy and open challenges," *IEEE Commun. Soc.*, vol. 16, no. 1, pp. 369–392, 1st Quart., 2014.

[52] L. Chunlin, Z. Min, and L. Youlong, "Efficient load-balancing aware cloud resource scheduling for mobile user," *Comput. J.*, vol. 60, no. 6, pp. 925–939, Jun. 2017.

[53] T. Chen, A. G. Marques, and G. B. Giannakis, "DGLB: Distributed stochastic geographical load balancing over cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 1866–1880, Jul. 2017.

[54] Q. Liu, W. Cai, J. Shen, X. Liu, and N. Linge, "An adaptive approach to better load balancing in a consumer-centric cloud environment," *IEEE Trans. Consum. Electron.*, vol. 62, no. 3, pp. 243–250, Aug. 2016.

[55] A. Montazerolghaem, M. H. Yaghmaee, A. Leon-Garcia, and M. Naghibzadeh, "A load-balanced call admission controller for IMS cloud computing," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 806–822, Dec. 2016.

[56] J. Zhao, K. Yang, X. Wei, Y. Ding, L. Hu, and G. Xu, "A heuristic clustering-based task deployment approach for load balancing using Bayes theorem in cloud environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 305–316, Feb. 2016.

[57] P. Jain and S. K. Sharma, "A systematic review of nature inspired load balancing algorithm in heterogeneous cloud computing environment," in *Proc. IEEE Inf. Commun. Technol.*, Nov. 2017, pp. 1–7.

[58] L. Liu, Z. Qiu, and J. Dong, "A load balancing algorithm for virtual machines scheduling in cloud computing," in *Proc. IEEE Modelling, Identificat. Control (ICMIC)*, Jul. 2017, pp. 471–475.

[59] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, Nitin, and R. Rastogi, "Load balancing of nodes in cloud using ant colony optimization," in *Proc. IEEE Comput. Modelling Simulation (UKSim)*, Mar. 2012, pp. 3–8. doi: 10.1109/UKSim.2012.11.

[60] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *Proc. IEEE Chinagrid Conf. (ChinaGrid)*, Aug. 2011, pp. 3–9.

[61] K. M. Sim and W. H. Sun, "Ant colony optimization for routing and load-balancing: Survey and new directions," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 33, no. 5, pp. 560–572, Sep. 2003.

[62] F. Mohammed, A. I. Alzahrani, O. Alfarraj, and O. Ibrahim, "Cloud computing fitness for E-government implementation importance-performance analysis," *IEEE Access*, vol. 6, pp. 1236–1248, 2017.

[63] S. R. Tehrani and F. Shirazi, "Factors influencing the adoption of cloud computing by small and medium size enterprises (SMEs)," in *Human Interface and the Management of Information. Information and Knowledge in Applications and Services*. Cham, Switzerland: Springer, 2014, pp. 631–642.

[64] M. Stieninger and D. Nedbal, "Diffusion and acceptance of cloud computing in SMEs: Towards a valence model of relevant factors," in *Proc. 47th Hawaii Int. Conf. Syst. Sci.*, 2014, pp. 3307–3316.

[65] H.-P. Fu and T.-S. Chang, "An analysis of the factors affecting the adoption of cloud consumer relationship management in the machinery industry in Taiwan," *SAGE J.*, vol. 32, no. 5, pp. 1741–1756, 2015.

[66] R. Rai, G. Sahoo, and S. Mehfuz, "Exploring the factors influencing the cloud computing adoption: A systematic study on cloud migration," *SpringerPlus*, vol. 4, p. 197, Dec. 2015.

[67] Q. Fang, J. Wang, and Q. Gong, "QoS-driven power management of data centers via model predictive control," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 4, pp. 1557–1566, Oct. 2016.

[68] S. Li, M. Brocanelli, W. Zhang, and X. Wang, "Integrated power management of data centers and electric vehicles for energy and regulation market participation," *IEEE Trans. Smart Grid*, vol. 5, no. 5, pp. 2283–2294, Sep. 2014.

[69] A. Rahman, X. Liu, and F. Kong, "A survey on geographic load balancing based data center power management in the smart grid environment," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 214–233, 1st Quart., 2014.

[70] L. Ganesh, H. Weatherspoon, T. Marian, and K. Birman, "Integrated approach to data center power management," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1086–1096, Jun. 2013.

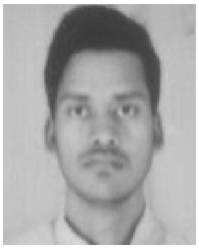
[71] B. Aksanli, "Data center peak power management with energy storage devices," *IEEE Internet Comput.*, vol. 21, no. 4, pp. 26–33, Jul. 2017.

[72] A. K. Singh and J. Kumar, "Secure and energy aware load balancing framework for cloud data centre networks," *Electron. Lett.*, vol. 55, no. 9, pp. 540–541, 2019.

[73] A. Murk, W. Malik, I. Mahmood, N. Ahmed, and Z. Anwar, "Big data in motion: A vehicle-assisted urban computing framework for smart cities," *IEEE Access*, vol. 7, pp. 55951–55965, May 2019.

[74] D. Shen, J. Luo, F. Dong, and J. Zhang, "VirtCo: Joint coflow scheduling and virtual machine placement in cloud data centers," *Tsinghua Sci. Technol.*, vol. 24, no. 5, pp. 630–644, Apr. 2019.

NILADRI SEKHAR DEY received the B.Tech. degree from the West Bengal University of Technology, Kolkata, and the M.Tech. degree in computer science and engineering from Jawaharlal Nehru Technological University, Hyderabad. He is currently pursuing the Ph.D. degree with Koneru Lakshmaiah Educational Foundation at Green Fields, India. He is recognized as the AWS Cloud Faculty Ambassador, in 2019. He is currently an Assistant Professor with the B. V. Raju Institute of Technology, Hyderabad. His areas of interest include cloud computing, data centre optimization, virtualization, process optimization, and machine learning.



T. GUNASEKHAR received the B.Tech. degree in information technology and the M.Tech. degree in computer science and engineering from Jawaharlal Nehru Technological University, Anantapur, and the Ph.D. degree for notable contributions on insider attack detections for cloud-based infrastructures from the Koneru Lakshmaiah Educational Foundation. He is having rich research experience on privacy preservation for data centric and resource centric frameworks. His research works also extend into cloud security, machine learning, network security, TPM-based computing, and cryptographic techniques. He is currently an Associate Professor and a Research Guide for many research scholars with Koneru Lakshmaiah Educational Foundation at Green Fields, India.