Load balancing techniques in cloud computing environment: A review

Dalia Abdulkareem Shafiq, N.Z. Jhanjhi †, Azween Abdullah

School of Computer Science & Engineering (SCE), Taylor’s University, Subang Jaya, Malaysia

Abstract

Cloud Computing is a robust model that allows users and organizations to purchase required services per their needs. The model offers many services such as storage, platforms for deployment, convenient access to web services, and so on. Load Balancing is a common issue in the cloud that makes it hard to maintain the performance of the applications adjacent to the Quality of Service (QoS) measurement and following the Service Level Agreement (SLA) document as required from the cloud providers to enterprises. Cloud providers struggle to distribute equal workload among the servers. An efficient LB technique should optimize and ensure high user satisfaction by utilizing the resources of VMs efficiently. This paper presents a comprehensive review of various Load Balancing techniques in a static, dynamic, and nature-inspired cloud environment to address the Data Center Response Time and overall performance. An analytical review of the algorithms is provided, and a research gap is concluded for the future research perspective in this domain. This research also provides a graphical representation of reviewed algorithms to highlight the operational flow. Additionally, this review presents a fault-tolerant framework and explores the other existing frameworks in the recent literature.

Keywords:
Cloud computing
Virtualization
Task scheduling
Load balancing
Resource optimization
Algorithms

Contents

1. Introduction ....................................................................................................... 00
   1.1. Research highlight & contribution ............................................................ 00
   1.2. Survey plan & organization .................................................................... 00
2. Literature review ................................................................................................ 00
   2.1. Existing review articles ........................................................................... 00
   2.2. Load balancing ....................................................................................... 00
         2.2.1. Load balancing model .................................................................... 00
         2.2.2. Load balancing metrics .................................................................. 00
         2.2.3. Existing common load balancing algorithms ................................. 00
   2.3. Recent literature on load balancing ........................................................... 00
         2.3.1. Load balancing based Throttled algorithm .................................... 00
         2.3.2. Load balancing based Equally Spread current execution .......... 00
         2.3.3. Load balancing based round Robin ............................................ 00
         2.3.4. Load balancing based Weighted round Robin ......................... 00
         2.3.5. Load balancing based Min-Min algorithm .................................. 00
         2.3.6. Load balancing based Honey Bee algorithm ............................... 00
         2.3.7. Load balancing based Genetic algorithm .................................... 00

© 2021 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
1. Introduction

Cloud Computing is the prominent technology that offers services (private and public) for example, accessing data, programs, and files easily across the internet (cloud), scalable storage services online instead of locally stored files on users' machines such as computers or phones. Prof. Rammath Chellappa proposed this well-known technology in the year 1997 (Agarwal and Srivastava, 2017) and it is also known to offer dynamic services such as (Nazir, 2012) cheap, scalable alternatives (Abdalla and Varol, 2019) and various services to clients. It is a technology that enhances businesses worldwide as it aims to reduce hardware costs. The technology utilizes the concept of the Pay-Per-Use model and many of its services are commonly seen in famous technology companies, for example, Google, Microsoft, IBM, and so on. This model allows clients to purchase the services required per their needs, similar to a metered service, or more known as subscriptions. This type of model is widely used in the Software as Service (SaaS) delivery model (Lowe and Galhotra, 2018). A summary of Cloud Computing is provided in Fig. 1 below. All cloud entities work together to handle the cloud environment. For example, cloud auditors act as the police in the cloud ensuring that the services offered by CSPs of high quality and integrity. Cloud carriers make sure that there is a stable connection to transport the services to clients (cloud users). The Data Center in the private cloud is located inside the organization's network whereas for the public, it is on the internet dependent on the Cloud Service Providers (CSPs) and for hybrid, it can be located in both.

In a typical Cloud Computing environment, there are two components: the frontend side and the backend side. The frontend is on the user side where it is accessible through connections over the Internet (Odun-Ayo et al., 2018). Whereas for the backend side, it deals with cloud service models. As can be seen in Fig. 2 below, it consists of a Data Center where multiple physical machines are
stored (known as servers). Upcoming user requests are received from the application dynamically schedule and resources are allocated to clients through the Virtualization. The virtualization technique is used to handle dynamic resources in the cloud as well as balancing the load in the entire system. It is also responsible for scheduling (Jyoti et al., 2019) and efficient allocation of resources.

User sends requests via the internet and these requested are stored in Virtual Machines (VMs) and CSPs in every delivery model have to maintain the QoS by ensuring the requests sent by users can be executed and completed within a specific deadline (Adhikari and Amgoth, 2018). The process of allocating user tasks to appropriate VMs depends on a scheduling policy (Data Broker) which in turn should be able to result in a balanced workload among machines and servers. Efficient scheduling and utilization of resources can be achieved by designing and developing a dynamic load balancer.

1.1. Research highlight & contribution

Our contribution to the study can be summarized as follows: a survey of 58 existing load balancing algorithms, along with their issues and strengths, in addition to the existing review papers, flowcharts of the algorithms, and a compilation of the experimental results is also included based on specific metrics such as Response Time and Processing Time. Furthermore, a proposed framework to address the fault tolerance issue in the Load Balancing algorithm to enhance the migration technique and avoid failures of nodes. Although, this issue has been slightly addressed by researchers in the past, however, most focus on the use of a single load balancer. The proposed fault-tolerant model address failures by using dual Load Balancers and machine learning tools to predict failures in the active Load Balancer. Future research can be done to enhance this model by identifying the suitable predictive model to achieve this. The authors have also compiled the outcome of the comparison of existing review articles based on the number of years and aligning with the analysis factors as can be seen in Fig. 3 below.

1.2. Survey plan & organization

- This survey has been done to include several up-to-date articles in the field of Load Balancing in Cloud Computing. The plan of this survey is summarized in the following points:

  - Selection of the reviewed research articles: articles included in the survey are selected based on Load Balancing in Cloud Computing from various reputed sources such as ScienceDirect, ResearchGate, SpringerOpen, and IEEEXplore. Filtration is done based on the title of the article, then the abstract. Finally, the selection of the articles is carefully done by analyzing the quality of the content, the algorithms with the intent of improving existing common algorithms.

  - Presentation of the survey paper: the paper is organized and presented to make it easier for readers to understand the core concept of Load Balancing in cloud environments. The articles are surveyed as the following:
    - Initially, the core algorithm of the approach is identified, for example, the approach is trying to improve on Round Robin and so on. Doing so can give readers an idea of the purpose of such algorithms and how to utilize them correctly for various objectives of Load Balancing. Then, the objective of the approach is identified along with the parameters considered such as Resource Utilization, Response Time, Makespan etc.
    - Secondly, the operational flow of the algorithm is represented in flowcharts.
    - Finally, the experimental results are given where possible.

In addition to the textual explanations, images and graphical representations are also included to further explain the concepts in clearer and easier ways for readers to understand.

- Analysis of reviewed algorithms: then the reviewed algorithms are examined based on the performance metrics used, the nature of the algorithm (static, dynamic, or nature-inspired), the type of the algorithm (multi-objective or single-objective). This will benefit the readers to choose the specific and suitable algorithms for different objectives in Load Balancing.

- Organization of the paper: The rest of the paper is organized as follows. Section 2 covers the literature review where it includes a review of some existing survey in this field, then the Load Balancing concept is explained along with its model, metrics associated with this concept and used by authors to analyze the articles, the taxonomy of the common load balancing algorithms, and finally, the literature review of the existing algorithms along with their flowcharts and experiment results. Section 3 provides a proposed framework to address one aspect.
As can be seen in Table 1 below, there has been an extensive number of published reviews regarding Load Balancing and the algorithms, highlighting the model, metrics, and existing common algorithms. Leading to recent literature in Load Balancing where proposed algorithms by researchers are explained and analyzed. Followed by new algorithms proposed by recent researchers in the field of Load Balancing, a review of the existing review papers will be provided. Then the experimental results of the reviewed articles based on the simulation results (4), the existing review includes a compilation of the research gap in this literature. Section 4 provides a discussion to induce the research gap. In Section 5, our research is concluded to review the concept and content of the paper, and suggestions for further research in the field of load balancing are included. Finally, in Section 6 suggestion for future topics is provided.

2. Literature review

This section includes the literature review of this paper. Firstly, a review of the existing review articles will be provided. Then the concept of load balancing will be explained highlighting its model, metrics, and existing common algorithms. Leading to recent literature on Load Balancing where proposed algorithms by researchers are explained and analyzed. Followed by new algorithms proposed by recent researchers in the field of Load Balancing. An organization chart for Section 2 is illustrated in Fig. 4 below.

2.1. Existing review articles

In Cloud Computing, it is vital to develop load balancers for equal distribution of nodes in the Data Center which leads to better resource utilization and fewer failures (Shiny, 2013) in the nodes. As can be seen in Table 1 below. There has been an extensive number of published reviews regarding Load Balancing and the algorithms, authors have reviewed thirty (30) of review papers in this field distributed in the past seven years (2014–2020) as can be seen in the comparative analysis in Table 1 below. The purpose of the table is to compare the current review paper with the existing papers as cited in the table.

There are still a few limitations in the existing review papers, for example, authors have reviewed the recent state of art however, it is a limited explanation as it does not include a comparative analysis (10). This is an essential feature when doing reviews as it provides deep analysis of the articles and makes it easy for readers to identify areas for improvement in future research. Most review papers do not include flowcharts (26) and thus it does not provide operational flow of the reviewed algorithms. Many papers lack the explanation and evaluation of the performance metrics (17) used in the reviewed articles which is another contribution of the current review paper. It is found that few authors focused on experimental results (4), the existing review includes a compilation of the experimental results of the reviewed articles based on the simulation platform. This is done to ease the understanding and reduce the hassle to search for such results. The paper is structured to review algorithms based on the underlying common algorithms. This way makes it easier for readers that aim to utilize a specific algorithm such as Round Robin, Genetic Algorithm, and so on.

This review categorizes algorithms based on their research gap addressing three main aspects in load balancing: Response Time, Fault Tolerance, others (such as Makespan, Waiting Time, and so on). Additionally, the paper proposed a framework to address one of the aspects above, which is fault tolerance.

2.2. Load balancing

Load Balancing is a method for optimizing the resources of VMs in the Cloud Computing environment. Load balancing in the cloud environment is one of the important techniques used to make sure there is an equal and dynamic distribution of workload and efficient resource utilization. The efficient balance of workload leads to higher user satisfaction and better resource allocation. In cloud systems, applying Load Balancing reduces delays in sending and receiving data (Kaur and Luthra, 2014) as well as preventing overloaded situations in the nodes that affect the QoS in cloud data centers. Thus, it is important to solve issues regarding Load Balancing and to enhance the performance of cloud-based applications which is discussed in more detail in this section.

2.2.1. Load balancing model

A model that defines the workflow of a load balancer in Cloud Computing is presented in Fig. 5 below. The user request is analyzed and passed to the selected Data Center based on the availability of resources. Those servers (VMs) should not be overloaded or underloaded, there needs to be an equal distribution among them. This is where the Load Balancing concept comes into place. To keep up the performance of cloud applications an efficient Load Balancer must be provided. Unequal load distribution could exist to many factors one of them is Task Scheduling. Without proper task scheduling, the resources will not be efficiently utilized. Load balancing happens at the backend of the cloud. Therefore, the articles reviewed by the authors in this paper are designed and focused on the server-side of Cloud Computing.

In the cloud, Load Balancing has two important goals (Shafiq et al., 2019): Resource Allocation and Task Scheduling. First, map-
| S/N | Ref. Year | Year | Comparative Analysis | Metrics Identification | Recent State-of-the-Art | Graphical Representation | Flowchart | Taxonomy | Experiment Results | Framework Presented? | No. of References |
|-----|-----------|------|----------------------|------------------------|------------------------|--------------------------|-----------|----------|-------------------|----------------------|------------------|
| 1.  | Our Review Paper | 2021 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 136 |
| 2.  | (Kumar and Chawla, 2020) | 2020 | ✓ | ✗ | U | U | U | U | U | U | U | 21 |
| 3.  | (Mishra et al., 2020) | 2019 | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | 55 |
| 4.  | (Duraisamy, 2019) | 2019 | ✓ | ✓ | x | x | x | x | x | x | x | 14 |
| 5.  | (Kumar and Kumar, 2019) | 2019 | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | 127 |
| 6.  | (Kamlesh et al., 2019) | 2019 | ✓ | ✓ | ✓ | x | x | ✗ | ✗ | ✗ | ✗ | 35 |
| 7.  | (Aziz and Kavitha, 2019) | 2018 | x | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | 71 |
| 8.  | (Kathalkar and Deorankar, 2018) | 2018 | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | 11 |
| 9.  | (Kumar and Raj, 2018) | 2018 | x | x | x | x | x | ✗ | ✗ | ✗ | ✗ | 14 |
| 10. | (Kaur and Verma, 2018) | 2018 | ✓ | x | x | x | x | x | x | x | x | 19 |
| 11. | (Hota et al., 2019) | 2019 | ✓ | x | x | x | x | x | x | x | x | 38 |
| 12. | (Mala and Prasad, 2018) | 2018 | ✓ | x | x | x | x | x | x | x | x | 33 |
| 13. | (Hamadah, 2017) | 2017 | ✓ | x | x | x | x | ✗ | ✗ | ✗ | ✗ | 14 |
| 14. | (Sutagatti and Kulkarni, 2017) | 2017 | x | x | x | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | 15 |
| 15. | (Deepa and Cheethu, 2017) | 2017 | ✓ | x | x | x | x | x | x | x | x | 9 |
| 16. | (Archanas and Sastry, 2017) | 2017 | x | ✓ | x | x | x | x | x | x | x | 15 |
| 17. | (Gupta et al., 2017) | 2017 | x | ✓ | x | x | x | x | x | x | x | 47 |
| 18. | (Thakur and Goraya, 2017) | 2017 | ✓ | ✓ | x | x | x | x | x | x | x | 162 |
| 19. | (Alam and Ahmad Khan, 2017) | 2017 | ✓ | ✓ | x | x | x | x | x | x | x | 31 |
| 20. | (Joshi and Kumari, 2017) | 2017 | x | ✓ | x | x | x | x | x | x | x | 15 |
| 21. | (Singh and S. Bhat J., R. Raju, and R. D. Souza, 2017) | 2017 | x | x | ✓ | x | x | x | x | x | x | 27 |
| 22. | (Jafarnejad Ghomi et al., 2017) | 2017 | ✓ | ✓ | x | x | x | x | x | x | x | 108 |
| 23. | (Milani and Navimipour, 2016) | 2016 | ✓ | ✓ | x | x | x | x | x | x | x | 84 |
| 24. | (Elngomi, 2016) | 2016 | x | x | x | x | x | x | x | x | x | 12 |
| 25. | (Goyal and Verma, 2016) | 2016 | x | x | x | x | x | x | x | x | x | 19 |
| 26. | (Gabi et al., 2015) | 2015 | x | x | x | x | x | x | x | x | x | 25 |
| 27. | (Kartheika and Kanakamal, 2015) | 2015 | ✓ | x | x | x | x | x | x | x | x | 11 |
| 28. | (Aslam and Shah, 2015) | 2015 | ✓ | ✓ | x | x | x | x | x | x | x | 64 |
| 29. | (Kapoor, 2015) | 2015 | ✓ | x | x | x | x | x | x | x | x | 9 |
| 30. | (Mj et al., 2014) | 2014 | x | x | x | x | x | x | x | x | x | 10 |
| 31. | (Sanghavi and Patalia, 2014) | 2014 | x | x | x | x | x | x | x | x | x | 16 |

*(✓) denotes the presence of the feature in the review paper; *(X) denotes the absence of a feature in the review paper.*
ping of tasks to correct VMs should be done such that there should be no overloaded node or an empty node or with less workload. Second, after the allocation process is done, Task Scheduling techniques are applied to enhance the completion process of tasks based on the user requirements to fulfill the specified deadline stated in the SLA document.

### 2.2.2. Load balancing metrics

In this subsection, the metrics considered by most authors when reviewing the recent state of art are provided. These metrics are essential in designing and developing a Load Balancing algorithm. These metrics determine the algorithm's quality in terms of performance in cloud applications. Some of these parameters are used by researchers to evaluate the proposed algorithm and they should be adjusted (Hung and Phi, 2016) for the prevention of imbalance situations in the cloud computing environment. Authors in (Afzal and Ganesh, 2019) emphasized the metrics used in the existing literature and have classified these metrics into two categories: Qualitative and Quantitative as summarized in the taxonomy diagram. Fig. 6 (Afzal and Ganesh, 2019) below has been modified where new parameters based on recent literature have been included to provide an up-to-date version and improve the accuracy of the figure.

This review paper focused on 9 main qualitative and quantitative metrics (Mishra et al., 2020; Thakur and Goraya, 2017) to analyze the recent literature as listed and explained below:

- **Resource Utilization (RU):** The extent of utilizing the resources (e.g. memory, CPU, etc.) in the system. It is to measure the degree of RU in the cloud Data Center. As long as there’s an increase in demand for services, RU is essential. A maximum RU is required for a good performance of the load balancing algorithm.

- **Scalability (S):** Similar to a system, an algorithm should perform well under any unexpected circumstances. Meaning regardless of the increase in the number of tasks and load, the algorithm should remain scalable. Highly scalable for a good performance of the load balancing algorithm.

- **Throughput (TP):** Represents the measure of the number of job requests that have been executed and processed successfully per unit time in the VM. It is the amount of data transferring from one place to another. High TP for a good performance of the load balancing algorithm.

- **Response Time (RT):** Amount of time taken by the algorithm to respond to a task. It takes into account the waiting time, transmission time, and service time. It is how much time is needed to respond to a user inquiry. Minimum RT is required in a good load balancing algorithm.

- **Makespan (MS):** Total completion time required to complete all tasks and allocate resources to users in the system. It is an essential metric in the scheduling process in the cloud environment. It is to measure the time taken to process a set of tasks. Minimum MS is required in a good load balancing algorithm.
**Associated Overhead (AO):** Amount of overhead formed while executing the load balancing algorithm. It could occur due to a high number of task migration and inter-process communication. A balanced load of system results in minimum overhead by the load balancing algorithm.

**Fault Tolerance (FT):** A load balancing should be able to perform well despite the failure of some system elements. Meaning if one VM is overloaded, another available VM should be able to execute tasks. High FT for a good performance of the load balancing algorithm.

**Migration Time (MT):** It is the total amount of time needed to migrate a task from one VM to another. The migration process should occur without affecting the system’s availability. It highly depends on the virtualization concept in the cloud. Low MT for a good performance of load balancing algorithm.

**SLA Violation:** Denotes the number of reductions of SLA violation factors in terms of deadline constraint, priority etc. Violations in SLA occurs due to situations where resources (VMs) are unavailable because they’re overloaded. Minimum SLA for a higher level of user satisfaction.

### 2.2.3. Existing common load balancing algorithms

This subsection explains the taxonomy of Load Balancing techniques. Common static algorithms used in the cloud environment such as Round-Robin are no longer appropriate and efficient due to many limitations such as the uneven distribution of load on nodes where some machines might become overloaded whereas others might be free of load (Adaniya and Paliwal, 2019). In addition to the use of static quantum which causes context switching to happen and thus results in delay and tasks being rejected. This causes improper allocation of tasks and an imbalance in workload.

Thus, many authors have contributed to developing algorithms to enhance the performance of cloud applications through the load balancing concept. These techniques can be implemented by developers either on the user end side which is known as Service Broker Policy or at the Data Center end side (Choudhary and Kothari, 2018). Fig. 7 below illustrates the taxonomy of the common existing Load Balancing algorithms. The purpose of this figure is to classify algorithms based on their underlying approach which is used in this review paper.

There are various common load balancing algorithms used to enhance the performance of CC. These algorithms are often categorized into three main types based on their underlying environment: static, dynamic, and nature-inspired algorithms as discussed below.

- **Static Load Balancing (SLB) Algorithms:** in a static environment, the load balancing algorithms’ process depends on prior knowledge (Alam and Ahmad Khan, 2017) of the system state along with its properties and capabilities. Examples of prior information could include memory, storage capacity, and processing power. Such information represents the load of the system. Static-based algorithms do not take into account dynamic changes to the load during runtime (Mj et al., 2014). Thus, the major drawback of these algorithms is low fault tolerance due to sudden changes in load.

- **Dynamic Load Balancing (DLB) Algorithms:** these types of algorithms are known to be better and adaptable for Load Balancing. Unlike SLB algorithms, in a dynamic environment, the load balancing algorithms take into account the previous state of the system (Alam and Ahmad Khan, 2017). Thus, the major benefit of these algorithms is flexibility although they...
might seem complex and may lead to high overhead on the system thus newly proposed algorithms in this category should avoid such drawbacks.

- **Nature-inspired Load Balancing (NLB) Algorithms:** such algorithms represent biological processes or activities based on human nature (Thakur and Goraya, 2017) such as the genetic process of the searching method of bees to find honey. These processes are modeled mathematically to adopt the natural processes to perform load balancing in CC. The development of these intelligent algorithms perform better for complex and dynamic systems.

Dynamic algorithms are better than static as no prior knowledge is needed and it takes the current state of the system which makes it more efficient for distributed cloud systems (Fatima et al., 2019). Also, dynamic algorithms eliminates the overhead for storing the previous state of the system and these algorithms have higher runtime complexity compared to static (Kamboj and Ghumman, 2016; Haryani and Jagli, 2014). Nature-inspired algorithms are known to be more intelligent as they belong to the metaheuristic class of load balancing algorithms and can resemble natural phenomena explained by natural sciences (Siddique and Adeli, 2015).

### 2.3 Recent literature on load balancing

This section covers the recent approaches that have been reviewed by authors. The following approaches aim to improve the performance of Cloud Computing by providing efficient load balancing techniques. The algorithms are reviewed stating their strengths and weaknesses.

#### 2.3.1 Load balancing based Throttled algorithm

This subsection explains the concept of the Throttled algorithm and how it can be used to solve the load balancing challenge in cloud computing as proposed by researchers.

**Throttled Algorithm (TA)** (Somani and Ojha, 2014): this is known as a dynamic LB algorithm. As can be seen in Fig. 8 below, the purpose of the load balancer is to search for a suitable VM to perform tasks upon receiving a request from a client. TA will keep a list of all VMs along with their index value this is known as the index table or the allocation table where the respective state (e.g., available/busy/idle) of VMs are stored as well. If a VM is available and has enough space, the task is accepted and allocated to the VM. If no available VM is found, then TA returns –1 otherwise the request is queued for fast processing. TA performs better than the Round Robin algorithm, however, It doesn't consider more advanced requirements for load balancing such as Processing Time (Patel et al., 2015; Bhagyalakshmi and Malhotra, 2017). The process of throttled algorithm is further explained in Fig. 9 in the flowchart below:

Similar to the traditional TA algorithm, Modified Throttled in (Nayak and Vania, 2015) maintains an index table of all VMs along with their states. However, the authors improvised on the Response Time and VMs utilization by selecting the VM at the first index if it’s available, a request is assigned and –1 is returned to the Data Center. Moving the VMs after the first index is selected and so on. This is different than traditional TA where VM’s at first index is selected every time there is a request.

Researchers in (Ghosh and Banerjee, 2016) presented a priority approach based on the modified Throttled algorithm (PMTA) with improved execution time to the existing algorithm. It focuses on allocating incoming tasks by using a switching queue to stop low priority tasks from executing to run high priority tasks first and distribute equal workload among several VMs. While the approach improved Response Time and waiting time compared to the existing TA and RR algorithm, it still might cause starvation and high Response Time for low priority jobs.

TMA algorithm (Phi et al., 2018) deals with equal workload distribution by maintaining two tables of VMs with states: available & busy. Unlike the traditional TA, where it keeps one table for all VMs and thus it’s harder to detect if a VM is available or not. The algorithm slightly reduced Response Time from 402.66 to 402.63 (ms). Thus, more optimization is required to enhance the performance of the algorithm.

The comparative analysis in Table 2 below provides a review of the experimental results of studied literature that addressed the load balancing based on the TA approach.

#### 2.3.2 Load balancing based Equally Spread current execution

This subsection explains the concept of the Equally Spread Current Execution algorithm and how it can be combined with the

---

**Table 2**

| No. | Algorithm                        | Parameters for Load Balancing                                                                 | Tool           | Response Time | Processing Time |
|-----|----------------------------------|------------------------------------------------------------------------------------------------|----------------|---------------|-----------------|
| 1.  | TMA (Phi et al., 2018)           | Response time & Processing time                                                                  | CloudAnalyst   | 402.63 ms     | 173.04 ms       |
| 2.  | Modified Throttled (xxxx)        | Response Time                                                                                   | N/A            | N/A           | N/A             |
| 3.  | PMTA (Ghosh and Banerjee, 2016) | Response Time, Processing Time & Data Center Request Serving Time                               | CloudSim       | 0.065 ms      | 1.32 ms         |

* (N/A) denotes No evaluation; Based on the author’s knowledge.
Throttled algorithm to solve the load balancing challenge in cloud computing as proposed by researchers.

**Equally Spread Current Execution (ESCE):** this is a type of a dynamic algorithm in load balancing. It considers the size of the job as the priority then distributes the workload to a VM with a light load randomly. It is also known as the Spread Spectrum technique since it spreads the workload to different nodes (Moharana et al., 2013). ESCE depends on the use of a queue (Falisha et al., 2018) to store the requests and distribute load to VMs if there exists a VM that is overloading. A common problem of ESCE is that it could cause overhead when updating the index table due to the communication that occurs between the Data Center controller and the load balancer (Lamba and Kumar, 2014). The process of ESCE algorithm is further explained in Fig. 10 in the flowchart below:

A hybrid approach that combines both Equally Spread Concurrent Execution (ESCE) and TA is proposed by authors in (Sachdeva and Kakkar, 2017) to reduce the Response Time in CC. The Hybrid LB (TA & ESCE) keeps a HashMap list to keep the user requests and scans through this list to find the available VM. Unlike, TA where it returns – 1 if VMs are busy, ESCE is used to look for a machine with a minimum load to assign the task to it. Another similar approach to (Sachdeva and Kakkar, 2017) is proposed by authors in (Subalakshmi and Malarvizhi, 2017). The Enhanced LB (TA & ESCE) is to further optimize the resources. Instead of keeping a queue in case of VMs being busy, they aim to use a create host function to reduce the waiting time in the queue. Both algorithms are good to reduce the Response Time, however, both do not take into consideration the migration of VM in case if any fault occurs.

Another hybrid approach using TA and ESCE is proposed in (Rathore et al., 2018) to reduce the amount of waiting when task number is increased, unlike the approaches in (Sachdeva and Kakkar, 2017; Subalakshmi and Malarvizhi, 2017). Load Balancing Hybrid Model (LBHM) keeps a threshold limit for each VM, identified based on the capacity of the VM. However, it is unable to perform well in case of failure of nodes occurs.

A similar approach to Rathore et al. (2018) is proposed in Aliyu and Souley (2019). The Hybrid Approach (TA & ESCE) is proposed to keep a threshold value as the priority for each VM for equal workload distribution. Besides the reduction in Response Time, it is also good for low cost.

Another approach combining both TA and ESCE is presented in Khanchi and Tyagi (2016) as a Hybrid Virtual Machine Load Balancing (Hybrid VM LB). It considers the current workload and state of the VM to decide whether to allocate the task to it or not. By keeping a list of the cloudlets assigned, they can determine overloaded and underloaded VMs. This algorithm can reduce Response Time however, the Data Center Processing Time has not improved a lot compared to traditional ESCE.

In Babu et al. (2017) authors have also proposed a hybrid technique combining (TA & ESCE) algorithms to resolve the underutilization of resources problem in cloud computing by achieving the optimal consumption value. The algorithm is divided into three cases to address each status of the VM if it’s busy, available, and so on. The algorithm can reduce waiting time, turnaround time, and processing cost however, it has low fault tolerance in case of failure of VMs.

MET proposed in Alamin et al. (2017) is again a merge of TA and ESCE. Utilizing the features of both algorithms, this load balancer

---

**Table 3: Performance Comparison of Hybrid (Throttled + Equally Spread Current Execution) Algorithm Based Simulation Tool.**

| No. | Algorithm | Parameters for Load Balancing | Tool | Response Time | Processing Time |
|-----|-----------|-------------------------------|------|---------------|----------------|
| 1.  | Enhanced LB (TA & ESCE) (Subalakshmi and Malarvizhi, 2017) | Response Time & Machine Cost | CloudAnalyst | 558.00 ms | N/A |
| 2.  | LBHM (Rathore et al., 2018) | Response Time, Processing Time & Algorithm Cost | | 206.14 ms | 0.19 ms |
| 3.  | TA (Somani and Ojha, 2014) | Response Time, Processing Time & Data Center Request | | 202.55 ms | 1.32 ms |
| 4.  | Hybrid VM LB (Khanchi and Tyagi, 2016) | Request Servicing Time, Response Time | | 92.45 ms | 0.31 ms |
| 5.  | MET (Alamin et al., 2017) | Response Time, Processing Time, Cost | | 55 ms | 2.16 ms |
| 6.  | Hybrid LB (TA & ESCE) (Sachdeva and Kakkar, 2017) | Response Time, Processing Time, Earliest Finish Time & Turnaround Time | CloudSim | 151.00 ms | N/A |
| 7.  | Hybrid Approach (TA & ESCE) (Aliyu and Souley, 2019) | Response Time, Processing Time, Cost | | 315.68 ms | 7.57 ms |

* (N/A) denotes No evaluation; Based on the author’s knowledge.
keeps a list of available VMs and counts the number of requests allocated to it. Then it compares whether the new request size is equal to the available VMs found after scanning for the allocation process. If not equal, then it will look for the least load size VM. Else the request queued thus might result in high waiting time if the request is not served.

Another hybrid algorithm of TA and ESCE is presented in Mishra and Tondon (2016). The Proposed Hybrid (TA & ESCE) is similar to the approach above of distributing the workload of VMs in an overloaded situation to a VM with the least load. However, it will also create a new VM if no free VM is found to be allocated to the coming user request. Using the technique called the unfold spectrum the load is spread to different nodes using the ESCE algorithm.

Comparative analysis in Table 3 above provides a review of the experimental results of studied literature that addressed the load balancing based on the hybrid approach of TA and ESCE algorithms.

### 2.3.3. Load balancing based Round Robin

This subsection explains the concept of the Round Robin algorithm and how it can be used to solve Load Balancing challenges in Cloud Computing as proposed by researchers.

**Round Robin (RR) Algorithm (Kaurav and Yadav, 2019):** works in a circular and ordered procedure where each process is assigned a fixed time slot without any priority. This is a very common algorithm and is often used due to its simplicity in implementation. A common problem in load balancing is that after user requests, the allocation state of the VM is not saved and updated. Another common problem of this static algorithm is illustrated in the compiled Fig. 11 (Villanueva, 2015; Round Robin Load Balancing, 2020) below. As can be seen in the figure, in the first section, let us say 100 user requests want to use the application, the load balancer distribute the load firmly among the two servers. Now let us say there’s an additional 50 user requests as seen in Section 2, since the RR algorithm is used, it will work in a cyclic manner, where it will add up 50 more user requests and the total will be 100 for server A and B and 50 for server C thus, RR fails to firmly distribute the load among servers.

In Tailong and Dimri (2016), the authors proposed a Modified Optimize Response Time algorithm to modify the existing Response Time service broker policy in CloudAnalyst. The algorithm calculates the Response Time and waiting time for each process then decides on the scheduling process. It can reduce the Response Time; however, it did not address the problem of time quantum in RR which makes the algorithm less suitable for dynamic cloud environments.

An enhancement to RR using the Genetic Algorithm (GA) approach is presented in Kaurav and Yadav (2019). It aims to provide an efficient load balancing which can improve the capability of Data Centers. To resolve the issues of RR, the approach allocates requests by scanning through the hash map that efficiently contains all VMs. If a VM is available, the task is allocated otherwise, the best VM will be chosen by analysing the best-fitted tasks using GA. Results show that the algorithm reduces the Response Time of servers significantly. However, GA tends to be complex when the search space increases (Katyal and Mishra, 2013).

Researchers in paper Issawi et al. (2015), have improved on the QoS in cloud applications by considering the burstiness workload problem. It is a problem in load balancing that occurs due to a sudden increase of users in cloud services, hence, Cloud Computing should consider both situations. Thus, an Adaptive Load Balancing (Adaptive LB (RR + Random)) algorithm is proposed to provide equal distribution of received tasks to VMs efficiently under heavy (bursty) workload by swapping between random (if workload is normal) and RR (if workload is bursty) task scheduling policies. Adaptive LB results show a reduction in Response Time however since RR is used, a static quantum is applied which leads to more waiting time.

In Richhariya et al. (2015) and Richhariya et al. (2015), the proposed hybrid approach is a combination of the priority policy and RR algorithm. Improved RR is an approach consisting of two processors: a small processor where it is used to calculate the time slice for each process and a main processor where the processes are arranged ascendingly based on their burst time which is taken as the priority value. The approach aims to reduce the Response Time however there is no performance testing to deduce the results.

Table 4 above provides a review of the experimental results of studied literature that addressed the load balancing based on the RR approach.

| No. | Algorithm | Parameters for Load Balancing | Tool                | Response Time | Processing Time |
|-----|-----------|--------------------------------|---------------------|---------------|-----------------|
| 1   | Modified Optimize Response Time (Tailong and Dimri, 2016) | Response time & Processing Time | CloudAnalyst | 152.72          | 1.66 ms         |
| 2   | Genetic Algorithm (GA) (Kaurav and Yadav, 2019) | Response time & Processing Time | CloudAnalyst | 161.43 ms       | 0.55 ms         |
| 3   | Improved RR (Pasha et al., 2014) | Response time & Processing Time | CloudSim | 300.9 ms        | 1.21 ms         |
| 4   | Adaptive LB (RR + Random) (Issawi et al., 2015) | Response time & Processing Time | CloudSim | 370 ms          | 315 ms          |
| 5   | WMaxMin (Khatavkar and Boopathy, 2017) | Response Time & Average Waiting Time | CloudSim | 202.55 sec      | 1.32 sec        |
2.3.4. Load balancing based Weighted round Robin

This subsection explains the concept of the Weighted version of the Round Robin algorithm and how it can be used to solve load balancing challenges in cloud computing as proposed by researchers.

Weighted Round Robin (WRR): similar approach to traditional RR, however, this algorithm considers the weight of each node. This weight is assigned by the developer (Moly et al., 2019) and it's based on the VM with the highest capacity. Although this algorithm is great for estimating the waiting time, however, it does not consider different lengths of tasks to allocate for the appropriate VM (Mayur and Chaudhary, 2019). This algorithm results in 694.82 ms Response Time (James and Verma, 2012) which is quite high.

A task-based Load Balancing is presented in Pasha et al. (2014) which is based on the RR approach. The proposed algorithm is an Improved RR that utilizes a hash map to store the last entry given from a user base to reduce overall Response Time in cloud applications. In Khatavkar and Boopathy (2017), an efficient load-balancing algorithm is proposed based on the combination of Weighted RR and Max-Min (WMMaxMin) to reduce the waiting time and Response Time. It uses Max-Min to select tasks having maximum processing time and assigns it to VM having the highest processing capabilities (such as RAM, MIPS etc.) using weighted RR. Both approaches are not suitable for dynamic environments.

Authors in Manaseer et al. (2019) proposed a method known as the MEMA technique where it takes into account requests with priority. Thus, it divides the original load balancer into two parts: normal and urgent requests. It utilizes a Weighted Round Robin (WRR) algorithm where each VM sends its weight to the server (which is known as the load balancer) to determine the number of requests that can be allocated. This approach has a limitation of high running time compared to WRR and it assumes that all requests are on the same level of priority.

Another approach proposed based on Improved Weighted Round Robin (IWRR) is presented in Mishra and Scholar (2016). This approach aims to solve the fallbacks in selecting the weights for each server in the WRR method. It improves the performance by migrating tasks to balance the load by gathering resource information. It also estimates the length of the tasks to be transferred however, the simulation experiment indicates that all tasks have the same length.

In Manikandan and Pravin (2018) authors attempted to further improvise on the WRR algorithm by considering the tasks’ execution time in addition to the weight of the server. The Improved WRR works by finding the server with the maximum weight and assigning the task having the maximum execution time to it. Hence the approach aims to minimize the Response Time however, it does not take into account the state (busy/available etc.) of the VMs before assigning a task to them.

Another approach that utilizes WRR and clustering technique is proposed in Chen et al. (2017). This Cloud Load Balancing (CLB) algorithm can cluster VMs based on CPU, memory and assign different weight values to each VM. Results show that the Response Time is reduced in a virtual web server when there is the same number of connections. This technique manages to resolve problems of access failures.

2.3.5. Load balancing based Min-Min algorithm

This subsection explains the concept of the Min-Min algorithm and how it can be used to solve the load balancing challenges in cloud computing as proposed by researchers.

Min-Min (MM) algorithm (Arshad Ali et al., 2019): This is an algorithm where it considers the least/minimum completion time for scheduling and balancing. Several shortcomings of MM include: inability of running tasks simultaneously, the algorithm gives high priority to smaller tasks (Patel et al., 2015) which leads to starvation for larger tasks and in turn results in imbalanced VM load. The steps of the algorithm are illustrated in Fig. 12 below.

A Resource Aware Min-Min (RAMM) approach is proposed in Arshad Ali et al. (2019) which is designed to enhance the
traditional Min-Min algorithm. The proposed algorithm provides a matrix to store tasks. It maps the tasks to its resource (VM) taking into account the completion and execution time. Scheduling of jobs is done depending on two parameters: minimum expected execution and Completion time (CT) of tasks on the VM. However, a limitation to this proposed approach is that it does not consider the current and updated load of the VMs in the task allocation process.

In Patel et al. (2015) authors proposed another Resource Load-Balanced Min-Min (ELBMM) algorithm. It finds the tasks with the minimum execution time then it assigns to the VM/resources that have the least completion time which reduces the cost of utilization and throughput of the system however, it does not consider task priorities and Response Time is still high of 80 secs.

Authors in Shanthan and Arockiam (2018) proposed a Resource-based Load balanced Min-Min (RBLMM) algorithm which is designed to reduce Makespan and balance the workload on VMs. It calculates the value of the Makespan after it is allocated to the resource then this value is considered as the threshold value. Results shows that the Makespan value for the traditional MM algorithm is 10 secs and the maximum completion time (Make-span) for the proposed RBLMM algorithm is 7 secs which is greatly reduced however, it doesn’t take into account tasks’ priority and any important parameters related to QoS such as deadline.

### 2.3.6. Load balancing based Honey Bee algorithm

This subsection explains the concept of the Honey-Bee algorithm and how it can be used to solve load balancing challenges in cloud computing as proposed by researchers.

**Honey Bee Algorithm (HB):** The idea of this is that a group of bees known as the foraging bees, spread to look for food sources and send location information to the other bees. This is known to solve decision making and classification problems with more robust and flexible ways (Kiritbhai and Shah, 2017). As can be seen in Fig. 13 from the flowchart below, the fitness of the bees is evaluated every time, and the searching for food process is repeated (Yuce et al., 2013).

Similarly, in a cloud environment, there are various changes of demand on the servers, and services are allocated dynamically and VMs should be utilized to the maximum limit reducing the waiting time. Honey bee behavior can be mapped to a cloud environment as can be seen in Table 5 below. The problem with this algorithm is that tasks with low priority tasks will be waiting in the queue.

Authors in (Babu et al., 2015) have utilized Bee Colony to propose a load balancing algorithm (Proposed LB based Bee Colony). It finds the load of VMs and decides on the load balancing, group VMs and transfer tasks from one VM to another. Load of VM is calculated based on the size and rate. Balancing is required when the capacity of the Data Center is greater than the threshold. The algorithm improves on the QoS by considering the priority of tasks.

A similar approach to Nair et al. (2019) was proposed in Hashem et al. (2017), Load Balancing Algorithm Honey Bee (LBA_HB) also considers current job count to allocate tasks, however in addition to the previous algorithm, the VMs are given a priority value. Additionally, VM overloading situations is avoided. The authors apply the concept of Honey Bee behavior, VMs will be updated when the task is allocated and it will notify the other tasks. Tasks are put in the waiting queue based on First Come First Serve (FCFS) algorithm. Tasks should be allocated to a VM with the minimum load, else it is delayed until the next available VM occurs. Results show that the LBA_HB algorithm increases Response Time by 50% compared to modified TA and RR. However, since FCFS is used, no priority is enforced on tasks.

A similar approach to Khatavkar and Boopathy (2017) in assigning weights to VMs is proposed by George et al. (2017) known as an enhanced HB Behavior Load Balancing (Enhanced HBBLB) algorithm. High priority of task is considered before assigning to a suitable VM. Results show that Enhanced HBBLB improves response time by 0.15 ms. However, it may increase waiting time for tasks with low priority.

An enhancement to the HB algorithm is made in Ehsanmoghadam and Effatparvar (2018) where priority of job is also considered. Honey Bee Behavior Load Balancing (HBB-LB) algorithm is designed to reduce the search time for allocating tasks. The public cloud is sectioned into three sectors known as underloaded, balanced, and overloaded after the load of each VM has been calculated. It then calculates the difference between the loads, based on if it’s more than zero then no movement of VM from one sector to another is required. It also considers speed and cost as a priority value for VMs however, it does not present a solution for the case of two similar priorities.

A hybrid approach of HB and RR (HB + RR) algorithms is proposed by authors in Kiritbhai and Shah (2017). It explains how this evolutionary algorithm can be used for efficient load balancer in the context of bees’ behavior. The approach aims to solve the priority issue of the HB algorithm, however, using RR will still raise issues regarding using many tasks due to the static quantum used in this algorithm.

In Gundu and Anuradha (2019), authors proposed a combination of Round Robin, Throttled, Equally Spread Current Execution, And Honey Bee (RTEH) With Artificial Bee Colony (ABCO)

| Honey-Bee Hive | Cloud Environment |
|----------------|--------------------|
| Honey-Bee Food source | Task (Cloudlet) |
| Searching (foraging) for food source | Virtual Machine |
| Honey-Bee getting crowded (depleted) near the food source | Overloading state of the Virtual Machine |
| A new food source is found. | Task is removed from overload Virtual Machine and scheduled to another, underloaded Virtual Machine. |

Table 5

How to Map Bee Colony with Cloud Environment (Babu et al., 2015).

![Genetic Algorithm](image_url)
Algorithms. To improve on the overhead limitation of RTEH which is when the index table is updating in the algorithm, authors have used ABCO to improve the process searching technique. It greatly reduced Makespan (292.46 s) compared to RTEH (295.35 s).

2.3.7. Load balancing based Genetic algorithm

This subsection explains the concept of the Genetic algorithm and how it can be used to solve the load balancing challenge in cloud computing as proposed by researchers.

**Genetic Algorithm (GA)** (Sharma et al., 2019): this approach is based on the evolution in nature. The algorithm works well since it does not focus on a single point and it resolves the resource deficiency issue resulting in multi-objective optimization. However, the GA algorithm tends to be complex when search space is increased which makes it time-consuming (Katyal and Mishra, 2013; Abdullah and Othman, 2012).

As can be seen in the compile Fig. 14 above, GA consists of five main phases (Gandhi, 2020). The process of the below steps continues until a satisfying result is obtained and a stopped point turns to true state:

- **Initialization**: a group of random individuals is initialized having different characteristics known as genes. They are usually depicted by binary numbers and a collection of them is known as a chromosome.
- **Fitness Function**: an experiment can be done, for example, to select the best fit individuals. These are given fitness values which helps in the next phase (Overview of Genetic Algorithm, 2020), in selection.
- **Selection**: this phase is a preparation for reproduction in the next phase, crossover. Best fit individuals are selected to pass on their genes to the next generation.
- **Crossover**: similar to the mating process where genes are transferred between the selected individuals. The genes of the parents are split into half and recombined this is known as one-point crossover.
- **Mutation**: after the mating process, the genes of the chosen offspring can be modified to produce the best child. This process makes sure there’s a diversity in the population (Overview of Genetic Algorithm, 2020).

In Dam et al. (2015) authors proposed to combine GA with Gravitational Emulation Local Search (GLS) algorithm making a hybrid approach known as GA-GEL. GLS algorithm demonstrates the gravitational attractions in a space where search is involved. Using the velocity calculated value of chromosome, this algorithm initiates the Population for GA. Similar to GA, where fitness is used for the selection, crossover, and mutation are applied. The approach fairly reduced the response time however authors do not consider any priority for the request.

2.3.8. Load balancing based Particle Swarm algorithm

This subsection explains the concept of the Particle Swarm Optimization algorithm and how it can be used to solve the load balancing challenge in cloud computing as proposed by researchers.

**Particle Swarm Optimization (PSO)** (Parmesivan et al., 2018): this algorithm demonstrates the natural gathering of the populace. For example, simulating the foraging behaviour of birds or “ducks gathering to find food”, in here the populace is known as the swarm and the particles are the ducks, representing individuals in the swarm. These particles will search globally using a set velocity and thus can alter and refresh the situation. This type of optimizer is proved to be very useful in Neural Network applications. It is similar to the Genetic Algorithm, however, PSO has simpler rules than GA as it doesn’t perform any mutation or crossover operations (Xiao et al., 2018). PSO aims for the optimal solution by doing iteration and uses fitness function to evaluate the quality of the solution as seen in the flowchart in Fig. 15 below.

In Yadav (2015), the authors proposed a hybrid approach combining PSO with Equally Spread Current Execution Load (ESCEL) algorithm (Hybrid PSO & ESCEL). PSO is used to optimize the jobs in the cloud server before assigning them. Then the server assigns the task using the ESCEL approach. This approach aims to optimize the resources and produce faster response time however there is no evaluation experiment to prove that.

2.3.9. Load balancing based Ant Colony Optimization Algorithm

This subsection explains the concept of the Ant Colony Optimization algorithm and how it can be used to solve load balancing challenges in cloud computing as proposed by researchers.

**Ant Colony Optimization (ACO)**: The motivation behind this algorithm is based on the behaviour of ants during their hunt for food. Ants travel randomly searching for food and when they return, it disposes an amount of chemicals known as pheromone (Verma et al., 2017). Such amount determines the shortest path from the nest to the food source for the other ants to follow. Although this approach is good for resource optimization, however, it results in slow Response Time and performance (Aslam and Shah, 2015). The behaviour of ants in ACO is illustrated in Fig. 16 below.

To make an effective scheduling and even distribution among servers, authors in (Kumar and Prashar, 2015) presented a hybrid algorithm combining ACO with priority ABC algorithm (Hybridized ACO & Priority–based Bee Colony) for higher performance rate. The priority is assigned to tasks based on the shortest job criteria. Similar to the behaviour of ants and bees the shortest distance is communicated among the nodes. Using CloudAnalyst, the proposed algorithm reduces response time and cost however it requires further enhancement to reduce the Data Center Processing Time.

A novel Ant Colony based strategy (LB strategy based on AC) was proposed in Dam et al. (2014) to search for underloaded nodes to make load balancing. Requests are allocated to VM using the FCFS algorithm and an index table is used for storage. Ants are used to travel randomly to find the optimal VM for allocation. The algorithm is designed to produce high QoS for customer’s jobs however it assumes that all jobs have equal priority which may in turn cause failures in the system.
Another approach where ACO is utilized for dynamic load balancing (Dynamic Novel Approach with ACO) is proposed in Selvakumar and Gunasekaran (2017) ants are generated in case of underloaded or overloaded situations in the cloud and then a searching procedure is applied to find the best candidate node for balancing. It considers the physical resources of VMs such as CPU, internal and external storage, I/O interface. The algorithm ensures high QoS in the cloud however, there is no performance testing to guarantee this.

HACOBEE Singh and Vivek (2016) is a hybrid algorithm combining ACO and Artificial Bee Colony where ants determine the load and bee colony finds the best fit VM for task allocation. It also uses the shortest job first to make priority of tasks in the population initialization stage. The algorithm is good for reducing response time however, it may not work in a dynamic environment.

2.3.10. Other existing recent load balancing algorithms

As can be seen in Subsection 2.2.1–2.2.8, there have been many proposed algorithms that focused on improvising the traditional common load balancing algorithms such as Round Robin, Min-Min, and so on. In this subsection, other new proposed algorithms are explained which aims to solve the challenge of balancing the load in a Cloud Computing environment as presented by researchers.

- **Cloud Partition Based Load Balancing** (Chaturvedi and Agrawal, 2017): this strategy makes use of the main controller to partition the cloud into four partitions resulting in optimized scheduling and load balancing. It consists of two algorithms: Partition Based Load Balancing Algorithm which is applied for searching of the finest partition for job allocation and secondly is Determination of Refresh Period algorithm where it is used to determine the best refresh period for the system to update its load. The approach aims to reduce execution time however it doesn’t set specific rules for the partition.

- **Weighted Active Monitoring Load Balancing (WAMLB)** (Singh and Prakash, 2018): this approach works well for heterogeneous environment where the weights of each VM is calculated using its bandwidth, number of processors, and the speed (see Fig. 17). The VM with the highest weight is selected and the Id is sent to the Data Center and the allocation table is updated. This approach aims to reduce response time in the cloud however it does not consider the utilization of VMs.

- **Central Load Balancer (CLB)** (Soni and Kalra, 2014): this algorithm is presented to address the priority of VMs. The priority is calculated by CLB by considering VMs values such as memory and CPU speed. The approach is great as it connects to all users and utilizes the VMs to their maximum capacities however the priority assigned to the VM is fixed which may not work efficiently when there are too many changes in the system's servers where urgent priority should be considered. This may also result in congestion issues in the system.

- **Dynamic Load Management** (Panwar and Mallick, 2016): for response time reduction, this algorithm ensures balancing of load is done based on the present status of VM. It checks for the best suited VM, then the index is removed after allocation of requests to ensure the VM is busy. The proposed algorithm obtains a better response compared to the optimal VM load balancer however, it may not work for static environments therefore more resource optimization is still required.

- **Improved Central Load Balancer** (Kaur and Sharma, 2018): this algorithm makes use of the priority system to reduce the resource wastage in the cloud Data Center. This would allow the system to select the best VM to service the coming requests. It reduces response time along with cost compared to ACO however authors share no information about how priority selection works in this approach

- **Proposed LB Strategy**: A load balancer that aims to allocate tasks based on the most powerful VM is presented in (Haidri et al., 2014). Once tasks are submitted, the proposed algorithm selects the most powerful VM in terms of capacities, allocates the tasks, and updates the power of each VM using a formula. Then it calculates the earliest finish time and response time of the task. Simulation results show that the approach can efficiently utilize the remaining capacities of VMs as the number of tasks is increased continuously. Since FCFS is used, the approach may result in high waiting time.

- **A Dynamic Load Balancing Algorithm (Dynamic LBA)** (Kumar and Sharma, 2017): This approach is designed to calculate and store the value of all VMs in queues. The algorithm makes use
of the concept of elasticity which is applied if the workload of the upcoming request becomes large and slows down the Data Center. Otherwise, it proceeds to look for VMs status, if the VM is overloaded, the task is transferred to the appropriate VM. It considers the task length which is randomly generated. The allocation process is performed in the First Come First Serve (FCFS) manner. The algorithm is adaptable in real-time environments which is the main advantage. It also reduces Makespan value. However, it may be time-consuming since it manually identifies VMs condition.

- **Dynamic Load Balancing** (Kumar and Sharma, 2017): this algorithm aims to minimize the Makespan time and increase the utilization of resources. It uses a bubble sort algorithm to sort tasks based on their length and processing speed. Then using FCFS manner the tasks are allocated to VMs. When this is done, load balancing is applied where VMs load is monitored and calculated. The approach is good for optimizing the resources however since FCFS is used, no priority is applied.

- **Heuristic-based Load Balancing Algorithm (HBLBA)**: A proposed heuristic algorithm for the IaaS model known as HBLBA is presented in (Adhikari and Amgoth, 2018). It aims to solve improper allocation of tasks to VMs by configuring servers based on the number of tasks, size, and VM compatibility to increase efficiency and find an appropriate VM. The approach works great for a small number of tasks, thus may be inefficient when there’s a large number of tasks in the system. Also using additional configuration information may slow down the process.

- **“My Load Balancer” algorithm** (Nair et al., 2019): this is for the allocation of tasks by computing the current count of jobs on a particular VM selected randomly. If the count value is less than its average, the task will be allocated to that VM. The algorithm is a great example of DBL however it only keeps count and does not consider important task parameters.

- **QoS-Based Cloudlet Allocation Strategy**: to improve on the system quality of service authors in (Bananjee et al., 2015) proposed a new cloudlet allocation policy with a better load balancing technique to reduce the cloudlet completion time. VMs Makespan, and host Makespan. The cloudlet is allocated to the VM with maximum load capacity. It is great to keep the system active and balanced however, it still suffers from high Makespan value for VMs and hosts. Also, it is not scalable in a large-scale environment as the experiment is built for 3 VMs only.

- **Least Frequently Used**: in (Kumar and Parthiban, 2014), authors proposed a new CloudAnalyst policy using the least frequently used VM mechanism to distribute the load evenly among VMs in the Data Center. The results of the approach reveal that it performs well on an average compared to other existing load balancing algorithms in CloudAnalyst. However, there’s no performance testing with other techniques rather than the consistent technique in this platform.

- **Modified Load Balancing Method**: The main idea behind the model proposed in (Patel et al., 2017) is the utilization of host resources and migration. Based on the geographical location, it selects the nearest Data Center and then calculates its utilization. It divides the host into two parts: overloaded and underloaded. The approach aims to reduce energy consumption by shutting down the underutilized host after process migration. However, there is no performance testing.

- **Roulette Wheel Selection Algorithm Load Balancing (RWSALB)** (Al-Marhabi et al., 2014): the weight value of each VM is considered in this algorithm for balancing the load. It first collects information about the VM then assigns requests to the VM with the highest weight. VMs are allocated a selection probability value based on the weight. Otherwise, requests are queued for processing (see Fig. 18). The algorithm can reduce data transfer cost however response time and Request servicing time are still high.

- **Dynamic Cost-Load Aware Service Broker (DCLASB)** (Rekha and Dakshayini, 2016): to ensure high QoS, the authors in this research proposed a dynamic algorithm that considers network latency for balancing the load. VMs are arranged based on their speed, the VM is chosen based on the request length. Data Center with the least processing time is selected, latency is compared and Data Center with the least cost is considered. The algorithm can achieve effective performance in the cloud however, it does not take into account the priorities of user requests.

- **Flexible Load Sharing (FLS) algorithm** (Bhatt and Bheda, 2016): to resolve the scheduling and load balancing issue in a cloud distributed environment, the authors in this research proposes a new algorithm that shows the basic technique for grouping VMs. Nodes in the cloud are sharing the load and exchange information which in turn reduces the overloaded VMs. This technique consumes less time for VMs and shows a better effect on performance in the cloud.

- **Starvation Threshold–Based Load Balancing (STLB)** (Semmoud et al., 2019): this proposed algorithm makes sure that load balancing happens when there is at least one free VM (starvation state) and this helps in reducing the number of task migrations from one VM to another and avoid additional overhead cost as it only deals with direct nodes for workload balancing. However, the algorithm is suitable only for independent tasks.

- **Improved LB algorithm (Priority Basis)** (Kaur and Ghumman, 2017): this is algorithm focuses on categorizing tasks in three leases: Cancellable, suspendable and Non-Preemptable, in case of two or more jobs having the same priority then the algorithm considers the type of the task. High priority is given to tasks that are cancellable to run first. The advantage of this algorithm reducing the response and waiting time, and processing cost however it may have a high risk of failure.

- **K-Means Clustering LB** (Kamboj and Ghumman, 2016): a similar approach to (Kaur and Ghumman, 2017) is proposed in this research however, this algorithm clustering and sorting tech-
niques at both sides, client and server is applied where both user requests and VMs are clustered. The algorithm aims to reduce overhead when VM scanning occurs and implements priority depending on the cost, higher cost means higher priority. The algorithm manages to efficiently utilize resources and reduce response time however it may have a high risk of failures.

- **Cluster-Based LB (Kamboj and Ghumman, 2016):** authors proposed a new algorithm that utilizes the space sharing policy. In case of a situation where a VM is unavailable to handle the job, the job can be shared among VMs and thus reduce response time. This algorithm applies k-means clustering however, clustering is done only for VMs and not tasks. The cluster identifies the min and max capacities of VMs. The algorithm reduces the overhead of scanning the entire list in VMs which makes it more dynamic.

- **VM-Assign Load Balancing (Domanal and Reddy, 2014):** unlike the Active load balancer, the authors in this study proposed a load balancer that checks the previous assigned request in the least loaded VM before the process of allocation of user request start. Then the VM is assigned a request and its id is returned to Data Center, else searches for the next least loaded VM. The algorithm can resolve the problem of inefficient utilization of resources however it may not be suitable for a dynamic cloud environment.

3. Proposed framework

- This section explains the proposed framework for improvised load balancing in Cloud Computing Environment. The main goal of the proposed framework is to provide a high availability cloud environment whereby it avoids system failures and recovering of user tasks which in turn enhances the security in Cloud Computing applications. This framework resolves fault tolerance issues in the cloud by using dual load balancers in the cloud environment and applying migration technique which has not been addressed in the previous literature up to the authors’ knowledge. As illustrated in Fig. 19 below, the proposed framework consists of two layers:

- **Top Layer:** deals with requests from multiple different clients (application’s users) of both mobile and desktop. Users can access the internet using multiple different devices to send requests to the cloud. In Cloud Computing, Data Center (DC) can be described as a big storage for cloud servers and data. DC receives requests and sends them to the active load balancer. In the top layer of the framework, there are two types of Load Balancers: Active Load Balancer and Passive Load Balancer (Heidi, 2020). The purpose of providing a passive load balancer is in case of failure in the main/active load balancer. Failures could exist for many reasons such as overloading the VMs with many requests or unnecessary migration of requests.

- **Bottom Layer:** deals with allocation of user requests to VMs. As can be seen from the figure, in the primary VMs batch, VM3’s status is overloaded. Thus, a migration technique should be applied to transfer the failed requests to another available VM found in the secondary VMs batch. This will then turn off the active Load Balancer and declares it as unavailable which can cause severe downtime of the system in the cloud. However, in this case, passive Load Balancers can take over and continue to re-allocate requests to available VMs. The allocation table is then updated whenever a VM becomes available, overloaded, idle and the number of requests it has been allocated.

The proposed framework makes use of replication concepts. Providing a standby load balancer can address fault tolerance issues in Cloud Computing which has been identified as a research gap in this literature. In the main/active Load Balancer, there may exist some misconfiguration which could lead to disruptions in the cloud services and eventually cause system downtime. The above framework will be able to detect and recover from such anomalies. A passive
Load Balancer must be available to take over when there is a failure therefore it must be configured similarly to the active load balancer. VMs should have the same workload to function properly and in case of an overloading situation, a migration technique must be applied to transfer requests as can be seen in the above figure. An Optimal Migration Algorithm will be applied in the Load Balancer. The purpose of the algorithm is to balance the load in VMs with minimal data movement. It computes and applies a Migration Threshold. The threshold can be calculated based on host utilization (Razali et al., 2014):

$$\text{Utilization}_{VM} = \frac{\text{Total Requested MIPS}}{\text{Total MIPS for that VM}}$$

In some cases, the active Load Balancer can make unnecessary migrations, this can be detected by training a model. The load balancer has a list of health status in 2 clusters to indicate whether it’s working properly:

- **OK**: all VMs are balanced and the algorithm is working per configuration.
- **CRITICAL**: number of migrations exceeded the threshold.

The model is trained to learn the optimal number of migrations. If the load balancer is in a critical state, it will then become unavailable and a Passive load balancer should be activated to handle the requests with high availability. To achieve this, a dataset must be used which can be obtained from the simulation result of the Optimal Migration Algorithm. The output of this algorithm can be saved in a CSV file. Then following the general process of Machine Learning below (see Fig. 20) to predict the limited number of migration:

After collecting the dataset, it will be pre-processed to remove any outliers and correct other mistakes. Then a Machine Learning model will be identified for the prediction of the number of migrations required, if it is beyond the limit, the status of the active Load Balancer will be CRITICAL. This framework will be modified in future research to identify the best Machine Learning tools to be used to achieve a complete fault-tolerant architecture in Cloud Computing for the improvement of performance and availability of services.

4. Discussion

In this section, a thorough investigation of the algorithms is discussed whereby it includes tables and figures to summarize the reviewed algorithms, the available implementation tools based on review, and finally, suggestions for future research.

4.1. Summary of the reviewed algorithms

After the analysis of the algorithms presented in the literature review sections is done, the authors have categorized the literature based on related research gaps. The research gap has been identified to highlight the future work that can be done by other researchers for more improvements. The comparative analysis is later compiled in tables to represent algorithms’ names (as proposed by previous researchers), the advantages, the disadvantages, the evaluation tools used by researchers for simulation, and finally the authors’ names, and year of publication. Existing algorithms are categorized based on four aspects: Response Time, Fault Tolerance, QoS and priority, and others (such as Makespan, Waiting Time, etc.)

### Table 6: An Overview of Reviewed Load Balancing Algorithms Based on Response Time

| S/N | Algorithm | Strengths | Weaknesses | Tool |
|-----|-----------|-----------|------------|------|
| 1.  | Modified Throttled (Nayak and Vania, 2015) | More flexible in servicing requests & selection of VMs. Improved response & process time; Reduce no. of queued requests. | Doesn't fully utilize VMs resources; No performance testing. No weakness based on the literature review. | CloudSim |
| 2.  | TMA (Phi et al., 2018) | Keeps a threshold value of VM; uses physical memory & storage capacity; Reduce costs. Avoid starvation; Distribute load evenly on servers; Considers tasks’ execution time & size; Reduce response, processing time & cost. | It may not function well in case of a failure in the system. Do not consider the current state of VMs before assigning tasks to them. | Eclipse framework |
| 3.  | Hybrid Approach (TA & ESCE) (Aliyu and Souley, 2019) | Reduces response time; Make use of dynamic VM set; Takes less time for request allocation. | Requests can still be assigned to overloaded VM; Not suitable for static environment. | CloudAnalyst |
| 4.  | Improved WRR (Manikandan and Pravin, 2018) | Considers priority of tasks; Eliminates overloading situations. | May increase time for tasks with low priority. | N/A |
| 5.  | Dynamic Load Management (Panwar and Malik, 2016) | Reduces response time; | Avg. response time is increased with more connections. | GridSim |
| 6.  | Enhanced HBBBLB (George et al., 2017) | Suitable for VMs with the same number of connections only. | May not be stable in cloud environments; No evaluation of response or completion time; Consider less parameters | CloudSim |
| 7.  | RWSALB (Al-Marhabi et al., 2014) | Consider less parameters | | |
| 8.  | CLB (Chen et al., 2017) | Compute and apply a Migration Threshold. The output of this algorithm can be saved in a CSV file. | | |
| 9.  | Hybrid PSO & ESCEL (Yadav, 2015) | | | |

* (N/A) denotes No evaluation; Based on the author’s knowledge.
An Overview of Reviewed Load Balancing Algorithms Quality of Service & Priority.

| S/N | Algorithm | Strengths | Weaknesses | Tool |
|-----|-----------|-----------|------------|------|
| 1   | Hybrid LB (TA & ESCE) (Sachdeva and Kakkar, 2017) | Decrease average response time; Avoid overloading situations. | Doesn’t consider migration of VM in case of failures. | CloudSim |
| 2   | Hybrid Technique (TA & ESCE) (Babu et al., 2017) | Reduces waiting time, turnaround time & processing cost; Efficient resource utilization | Doesn’t apply any VM migration technique | |
| 3   | RTEAH (Gundu and Anuradha, 2017) | Reduces waiting time, Makespan & burst time. | May result in high latency. | |
| 4   | STL (Sennoud et al., 2019) | Minimize response time; Maximize utilization rate; Decrease overall migration cost; Maintain system stability | Only suitable for independent tasks; | |
| 5   | Improved LB algorithm (Priority Basis) (Kaur and Ghumman, 2017) | Suitable for heterogeneous environment; Consider cost & priority of tasks; Increase profit for cloud | No comparison with other algorithms; No migration techniques applied & consumes high processing power. | |
| 6   | K-Means Clustering LB (Babu et al., 2017) | Cluster tasks based on cloudlet length & cost; Reduce processing cost, turnaround, waiting, & execution time. | Doesn’t address the same priority value; No migration techniques & consumes high processing power. | |
| 7   | Improved RR (Pasha et al., 2014) | Requires less time for VM scheduling. | Static algorithm; doesn’t consider tasks with different configuration. | |
| 8   | Enhanced LB (TA & ESCE) (Subalakshmi and Malarvizhi, 2017) | Minimize response time; Avoid overloading; Maximize services; Reduce machine cost. | No weakness based on the literature review. | CloudAnalyst |
| 9   | GA-GEL (Dam et al., 2015) | Minimization of VMs & Makespan | Ignores the priority of the job | |
| 10  | LBHM (Rathore et al., 2018) | Reduce avg. response time & processing time; Considers capacity & size of VM. | Doesn’t consider the current workload of VM; | |
| 11  | LB strategy based on AC (Dam et al., 2014) | Ensure high QoS of customer’s job; Reduce response time | Assumes all jobs have the same priority | |
| 12  | VM-Assign Load Balancing (Domanal and Reddy, 2014) | Efficient utilization of the resources. | Not suitable in a dynamic environment; | |
| 13  | Modified Optimize Response Time (Tailong and Dimri, 2016) | Apply mapping & sorting technique; reduces response time. | | |
| 14  | WAMLB (Singh and Prakash, 2018) | Can be applied for social networking sites such as Facebook etc.; Effectively utilize VM resources | Only considers RAM, Bandwidth & speed & not CPU for weight calculation; static weight used. | |

Table 8
An Overview of Reviewed Load Balancing Algorithms Quality of Service & Priority.

| S/N | Algorithm | Strengths | Weaknesses | Tool |
|-----|-----------|-----------|------------|------|
| 1   | PMTA (Ghosh and Banerjee, 2016) | Equal distribution of workload unlike RR; Reduce response time & improve service quality. | May cause starvation & high response time for low priority tasks. | CloudSim |
| 2   | HBB-LB (Ehsaninomoghadam and Effatparvar, 2018) | Reduce searching process for task allocation; Apply VM migration; Suitable for heterogeneous environment; | Doesn’t consider if two VMs have the same priority value; Higher Completion time compared to the honey bee algorithm. | |
| 3   | LBA_HB (Hasheem et al., 2017) | Able to handle remaining tasks by updating of allocation status; Avoid imbalanced workload. | Ignores task prioritization; doesn’t calculate transfer time of tasks & algorithm’s completion time. | |
| 4   | Dynamic Load Balancing (Kumar and Sharma, 2017) | Proper utilization of resources; Reduces Makespan; Provides elasticity. | FCFS for request allocation, no priority | |
| 5   | Hybrid VM LB (Khanchi and Yagi, 2016) | Reduce response time; Makes an efficient & effective management of CSP’s resources. | No threshold is applied to avoid allocation overloaded machine; No priority is enforced. | CloudAnalyst |
| 6   | Genetic Algorithm(GA) (Kaurav and Yadav, 2019) | Minimum time taken to finish user requests; Efficient scheduling among different VMs. | Less efficient when search space is increased; Doesn’t take into account task priority. | |
| 7   | Dynamic LBA (Kumar and Sharma, 2017) | Reducing Makespan; balancing of workload among VMs; higher resource utilization ratio. | No priority for tasks (depends on arrival time); can be time-consuming. | |
| 8   | Central Load Balancer (Soni and Kalra, 2014) | Achieve better load distribution in large-scale environment; Considers VMs priority. | Priority value of VMs is fixed; Does’t consider urgent priority; May cause congestion. | |
| 9   | Improved Central Load Balancer (Manikandan and Pravin, 2018) | No wastage of resource; Reduce response & processing time; Cost-effective solution | No information on the priority process; No performance testing. | |
| 10  | My Load Balancer (Nair et al., 2019) | Faster allocation of tasks to VMs. | Only keeps count of tasks & doesn’t consider deadline parameters, size, or completion time. | CloudAnalyst |
| 11  | DCLASB (Rekha and Daksheyini, 2018) | Considers the network latency & bandwidth; Reduce response time, processing & VM cost | Does not consider priority or user application constraint; Results in security issues | CloudSim |
| 12  | ELBMM (Patel et al., 2015) | Improve utilization of resources; Reduce Makespan. | No task priority | N/A |
| 13  | RBLLM (Shantan and Arooskan, 2018) | | | |
| 14  | RAMM (Arshad Ali et al., 2019) | Suitable for small tasks; Reduce completion time. | Do not consider the capacity of VMs; No priority. | MATLAB |
| 15  | MEMA Technique (Manaseer et al., 2019) | Considers urgent & normal messages; Fair distribution among VMs; Reduce response time & wasted requests. | Assumes that all urgent messages have the same level of priority; Higher runtime than WRR algorithm. | NetBeans |

* (N/A) denotes No evaluation; Based on the author’s knowledge.
environments such as applying the replication concept (Semmoud et al., 2019). For example, creating a backup load balancer to ensure the availability of resources. Based on the literature, this is still an issue and an interesting research area in this field. Still, many algorithms did not consider this capability as represented by the reviewed algorithms in Table 7 below.

### 4.1.4. Others

Other open issues related to Load Balancing are concluded from the literature such as Waiting Time, Makespan, Processing cost, and these are summarized in the algorithms in Table 9 below.

Table 9 below provides a review of the studied literature that addressed the load balancing quantitative and qualitative performance metrics. Also, we have classified them based on their underlying environment. This classification is carried out based on the information from Tables 6–9 and it is useful to identify whether the algorithm is single objective or multi-objective. To the best of the authors’ knowledge, the following algorithms (TMA, Hybrid LB (TA & ESCE), Hybrid Technique (TA & ESCE), LBHM, Modified Optimize Response Time, My Load Balancer, Proposed LB based Bee Colony, Improved RR, Improved WRR, WMMaxMin, Enhanced HBLBA, Hybridized ACO & Priority-based Bee Colony algorithm, HACOBEE, Dynamic Novel Approach with ACO, Modified Load Balancing Method RWSALB, Least Frequently Used, DCLASB) in the table are single-objective whereas the rest are multi-objective. From the table, we can see that majority of the authors focused on one main metric which is the response time as this metric can highly affect the performance and the quality of

| S/N | Algorithm | Strengths | Weaknesses | Tool |
|-----|-----------|-----------|------------|------|
| 1.  | TA (Somani and Ojha, 2014) | Reduce the response time & request serving time. | Uses RR, may take a long time to allocate VM due to context switching. | CloudSim |
| 2.  | Proposed LB based Bee Colony (Babu et al., 2015) | Improve QoS; Consider priority of tasks; Reduce the number of task migrations. | May result in high cost & low speed of VMs. | CloudSim |
| 3.  | IWRRLB (Mishra and Scholar, 2016) | Avoid overloading; Reduces waiting time; Uses symmetric load balancing to predict cloud load. | Assumes that all tasks have similar size; Results in high Makespan. | CloudSim |
| 4.  | Modified RR (Richhariya et al., 2015) | Eliminates the drawback of the RR approach where static time slice is used. | Doesn't use any performance parameters for evaluation such as response time etc. | CloudSim |
| 5.  | WMMaxMin (Khatavkar and Boopathy, 2017) | Reduces waiting time & response time; Balanced load among VMs. | Static algorithm; May increase waiting time for smaller tasks to execute. | CloudSim |
| 6.  | LB Strategy (Hadi et al., 2014) | Distributing equal workload; Avoid resource wastage & provides high availability. | Implements FCFS for the scheduling which results in higher waiting & turnaround time. | CloudSim |
| 7.  | Cloud Partition Based Load Balancing (Chaturvedi and Agrawal, 2017) | Improves system throughput; Efficient resource utilization; Use refresh period policy for better job scheduling. | Doesn't work well in a heterogeneous environment; No predefined partitioning rules | CloudSim |
| 8.  | QoS-Based Cloudlet Allocation Strategy (Banerjee et al., 2015) | Efficient resource utilization of the system; Improved Makespan & Quality of Service; Keeps a balanced & active system; Reduces cloudlet completion time. | Less scalable, only 3 VMs are used; Still results in a high Makespan value of the system; Does not address the overlapping situations of VMs | CloudSim |
| 9.  | Cluster-Based LB (Kambo and Ghuman, 2016) | Results in high data processing time. | No performance testing or comparison with other algorithms | CloudSim |
| 10. | TLS (Bhatt and Bherda, 2016) | Avoid overloading of nodes; increase the performance ratio. | Doesn't consider the cost of Data Center; Not suitable for heterogeneous environment; No testing. | CloudSim |
| 11. | Proposed Hybrid (TA & ESCE) (Mishra and Tondon, 2016) | Reduces search space in traditional TA; Evenly distribute load among nodes. | Doesn't consider the cost of Data Center; Not suitable for heterogeneous environment; No testing. | CloudSim |
| 12. | MET (Alamin et al., 2017) | Minimize response time & cost; Avoid underutilized resources. | May increase Processing Time & waiting time | CloudSim |
| 13. | Adaptive LB (RR + Random) (Issawi et al., 2015) | Improve QoS in case of peak in user demands; Reduces response & processing time. | Shortcomings of RR algorithm where static quantum is used thus increasing waiting time. | CloudSim |
| 14. | Hybridized ACO & Priority-based Bee Colony (Kumar and Prashar, 2015) | Efficient utilization & mapping of resources in Data Center; Reduces response time. | Results in high data processing time | CloudSim |
| 15. | HACOBEE (Singh and Vivek, 2016) | Efficient utilization & mapping of resources in Data Center; Reduces response time. | Unstable Performance; High processing cost; Not suitable for dynamic environment | CloudSim |
| 16. | Least Frequently Used (Kumar and Parthiban, 2014) | Distribute workload evenly in the Data Center. | No performance comparison of response time with other techniques | CloudSim |
| 17. | HBLBA (Adhiyarkar and Angoth, 2018) | Minimizes completion time of tasks & Makespan; Provides better resource utilization. | Not suitable for a larger number of tasks (200); Dependency on extra information e.g. host configuration may slow down the processing time. | CloudSim |
| 18. | HB + RR (Kirimihai and Shah, 2017) | Better performance, resource utilization & response time. | Not suitable for a large number of tasks. | CloudSim |
| 19. | Dynamic Novel Approach with ACO (Selvakumar and Gunasekaran, 2017) | Suitable for dynamic environment; Aim to ensure high QoS | No performance testing | CloudSim |
| 20. | Modified Load Balancing Method (Patel et al., 2017) | Reduce energy consumption; Distribute workload evenly across nodes. | No performance evaluation; little/no information shared about the methods used | CloudSim |

* (N/A) denotes No evaluation; Based on the author’s knowledge.
Load Balancing Algorithms Based on Their Underlying Environment and Their Corresponding Performance Metrics.

Table 10

| No. | Algorithm                                      | Environment | Performance Metrics |
|-----|-----------------------------------------------|-------------|---------------------|
| 1   | TA (Somani and Ojha, 2014)                    | Static      | RU     |
| 2   | Modified Throttled (Nayak and Vania, 2015)     | Static      | S      |
| 3   | PMTA (Ghosh and Banerjee, 2016)               | Dynamic     | TP      |
| 4   | TMA (Pil et al., 2018)                        | Static      | RT      |
| 5   | Hybrid LB (TA & ESCE) (Sachdeva and Kakkar, 2017) | Static      | MS      |
| 6   | Hybrid Approach (TA & ESCE) (Aliyu and Souley, 2019) | Static      | AO      |
| 7   | Proposed Hybrid (TA & ESCE) (Mishra and Tondon, 2016) | Dynamic     | FT      |
| 8   | Hybrid Technique (TA & ESCE) (Rabu et al., 2017) | Static      | SLA     |
| 9   | MET (Alamin et al., 2017)                     | Static      |         |
| 10  | Enhanced LB (TA & ESCE) (Subalakshmi and Malarvizhi, 2017) | Static      |         |
| 11  | LBHM (Rathore et al., 2018)                   | Static      |         |
| 12  | Hybrid VM LB (Khanchi and Tyagi, 2016)         | Static      |         |
| 13  | Modified Optimize Response Time (Tailong and Dimri, 2016) | Dynamic     |         |
| 14  | Genetic Algorithm (GA) (Kaurav and Yadav, 2019) | Dynamic     |         |
| 15  | Dynamic LBA (Kumar and Sharma, 2017)          | Static      |         |
| 16  | ELBMM (Patel et al., 2015)                    | Dynamic     |         |
| 17  | RBLMM (Shantan and Arocksam, 2018)            | Static      |         |
| 18  | RAMM (Arshad Ali et al., 2019)                | Static      |         |
| 19  | HBLBA (Adhikari and Ambgoth, 2018)            | Static      |         |
| 20  | My Load Balancer (Nair et al., 2019)          | Static      |         |
| 21  | Proposed LB based Bee Colony (Rabu et al., 2015) | Static      |         |
| 22  | LBA_HB (Hashem et al., 2017)                  | Static      |         |
| 23  | HBB-LB (Ehsanimmohghadam and Effatparvar, 2018) | Static      |         |
| 24  | Adaptive LB (RR + Random) (Issawi et al., 2015) | Static      |         |
| 25  | MEMA Technique (Manaseer et al., 2019)        | Static      |         |
| 26  | Improved WRR (Manikanand and Pravin, 2018)    | Static      |         |
| 27  | IWRLB (Mishra and Scholar, 2016)              | Static      |         |
| 28  | Improved RR (Pasha et al., 2014)              | Static      |         |
| 29  | Modified RR (Richriana et al., 2015)          | Static      |         |
| 30  | CLB (Chen et al., 2017)                       | Static      |         |
| 31  | WMaxMin (Khatavkar and Boopathay, 2017)       | Static      |         |
| 32  | Enhanced HBLLB (George et al., 2017)          | Static      |         |
| 33  | LB Strategy (Haidri et al., 2014)             | Static      |         |
| 34  | HB + RR (Kirthibai and Shah, 2017)            | Static      |         |
| 35  | RTFAH (Guundo and Anuradha, 2019)             | Static      |         |
| 36  | Hybridized ACO & Priority-based Bee Colony algorithm (Kumar and Prashar, 2015) | Static      |         |
| 37  | HACOBEE (Singh and Vivek, 2016)               | Static      |         |
| 38  | LB strategy based on AC (Dam et al., 2014)    | Static      |         |
| 39  | Dynamic Novel Approach with ACO (Selvakumar and Gunasekaran, 2017) | Static      |         |
| 40  | Cloud Partition Based Load Balancing (Chaturvedi and Agrawal, 2017) | Static      |         |
| 41  | WAMLB (Singh and Prakash, 2018)               | Static      |         |
| 42  | Central Load Balancer (Son and Kalra, 2014)   | Static      |         |
| 43  | Dynamic Load Management (Panwar and Malik, 2016) | Static      |         |
| 44  | Dynamic Load Balancing (Kumar and Sharma, 2017) | Static      |         |
| 45  | QoS-Based Cloudlet Allocation Strategy (Banerjee et al., 2015) | Static      |         |
| 46  | Modified Load Balancing Method (Patel et al., 2017) | Static      |         |
| 47  | RWALB (Ali-Marhabi et al., 2014)              | Static      |         |
| 48  | Improved Load Balancer (Rabu et al., 2015)    | Static      |         |
| 49  | Least Frequently Used (Kumar and Parthiban, 2014) | Static      |         |
| 50  | Hybrid PSO & ESCEL (Yadav, 2015)              | Static      |         |
| 51  | GA-GEL (Damp et al., 2015)                    | Static      |         |
| 52  | DCLASB (Retha and Dakshayini, 2018)           | Static      |         |
| 53  | Cluster-Based LB (Kamboj and Ghumman, 2016)   | Static      |         |
| 54  | FLS (Bhuri and Bhada, 2016)                   | Static      |         |
| 55  | STLBB (Semmboud et al., 2019)                 | Static      |         |
| 56  | Improved LB algorithm (Priority Basis) (Kaur and Ghumman, 2017) | Static      |         |
| 57  | K-Means Clustering LB (Kamboj and Ghumman, 2016) | Static      |         |
| 58  | VM-Assign Load Balancing (Domanal and Reddy, 2014) | Static      |         |

* (√) denotes the presence of the feature in the review paper; *(X) denotes the absence of the feature in the review paper.

cloud-based applications. However, no work has been done using associated overhead and fault tolerance parameters in the past few years which can be a new research area in this topic to enhance the overall performance of cloud applications.

Now we discuss some statistics based on the reviewed articles. Fig. 21(a) shows the distribution of the reviewed articles by the year of publication from 2014 until 2019. In the figure we see the number of articles in each year inside each corresponding slice; for example, the highest number of articles is in 2017 which 16, and the second-highest is in 2015 at 12. The lowest number of reviewed articles is in the years 2016 and 2019 which is 7. This research focused on reviewing articles from the past six years only to ensure a new and accurate research gap can be identified to proceed further for future research in the field of Load Balancing.

Load balancing is one of the important aspects to cover in Cloud Computing to enhance the performance of its applications. As can
be seen in Fig. 21(b), a good load balancing algorithm should be DLB or NLB and it should consider user requirements. In this paper, this research presented various load balancing algorithms, along with their underlying environment, we’ve concluded that these algorithms have their advantages and disadvantages, for example, SLB algorithms can perform well in a homogenous environment where the load is known before execution and the runtime is stable having almost same configuration settings for all VMs in the Data Center. SLB algorithms may fail in a heterogeneous cloud computing environment where the load is constantly and dynamically changing all the time however, the implementation is easier than Dynamic and nature-inspired algorithms. Although DLB algorithms are challenging to implement, handle and test but they are most suited in a heterogeneous cloud environment and proven to have higher performance and accuracy than SLB algorithms. This paper shows that new research is aiming for the use of NLB algorithms which are known to be metaheuristic techniques that can solve complex optimization problems such as scheduling of tasks, load balancing, neural network. As can be seen in Fig. 21(c), Load Balancing algorithms can be single objective, meaning it focuses on one performance parameter whereas multi-objectives can focus on more than one parameter. Most algorithms reviewed in this literature are multi-objective.

Literature reveals that response time is one of the essential metrics to be considered to evaluate the proposed algorithm as can be seen in Fig. 22 below. One of the most important tasks for the CSPs is to guarantee the service quality of their applications which is a vital component in the SLA document shared between users and CSPs. These quality attributes include are response time, task queuing length, waiting time, throughput, priority, and so on. Therefore, it has been used by most researchers in this literature. The second highest used parameter is resource utilization as it is an important goal of achieving an equal balance in the cloud Data Centers. This paper aims to help the forthcoming researchers studying in the field of Load Balancing, especially for reducing the Response Time in the cloud applications. RT is one of the challenges in cloud workload balancing and other distributed systems. It indicates the total amount of time to respond to a request coming from the user or client. In the LB algorithms, it is concluded that algorithms in a dynamic environment consumes higher response time than static algorithms (Alam and Ahmad Khan, 2017).

4.2. Available implementation tools

Simulation creates a virtual environment for testing and verifying research experiments for efficient and better solutions for the application. It is a scientific technique to make a model or a real-time system (Pakize et al., 2014). Thus it eliminates the need and expenses of computing facilities (Nazeer and Banu, 2015) for performance evaluation and modeling the research solution. This literature distributed the articles based on two main cloud simulation tools, CloudSim and CloudAnalyst (see Fig. 23). Other tools include

![Fig. 21. (a) Year of Publications of Reviewed Articles; (b) Nature of Reviewed Algorithms; (c) Type of Reviewed Algorithms.](image)

![Fig. 22. Evaluation Metrics of Reviewed Articles.](image)
Gridsim. Some of these simulation tools can be integrated with software such as Eclipse or NetBeans for programming the proposed algorithm. As concluded from Tables 6–9, 22 researchers used the CloudSim tool to enhance the performance of Cloud Computing and for modeling of load balancing. The second highest is CloudAnalyst which is an extension of the CloudSim tool. 15 researchers used this. For testing and simulating performance parameters such as VM migration and response time, CloudAnalyst is better than CloudSim. It also extracts the results in PDF or XML format (Byrne, 2017) which makes the workload description more clear and detailed.

4.3. Suggestions for future research

Based on this literature, it is concluded that there are still open research issues to be addressed in the future. The study has identified areas of improvement in the load balancing algorithms for researchers to consider in their future research in this field to further optimize the resources in Cloud Computing. These are listed below:

- Most reviewed algorithms here are not fault tolerance. This is one of the important factors that can be considered to design a stable algorithm to function well in the dynamic cloud environment. Failures in the algorithm could exist for many reasons such as sudden addition of nodes in the cloud Data Center, a high priority task waiting for execution, sudden change in the workload and configuration setting of VMs.
- Still, there are open challenges in Task migration since very few authors focused on the Migration Time parameter as can be seen in Tables 6–9.
- Some approaches still result in higher waiting times due to the underlying static algorithms used such as RR or FCFS. To resolve this, researchers can study how intelligent algorithms can be used to mimic the cloud environment such as nature-inspired algorithms that can solve complex optimization problems.

The above points conclude a new research gap that could be considered by researchers interested in this area to further optimize and enhance the performance of Cloud Computing applications. As the load balancing is helpful for the cloud optimization, energy aware task allocation (Mishra, 2020), which help different cloud application, mainly to the health related (Ali, 2020) critical applications.

5. Conclusion

Load Balancing is an important aspect in the field of Cloud Computing to enhance the workload distribution and utilize the resources efficiently which in return reduces the overall response time of the system. Plenty of approaches and algorithms have been proposed to solve issues related to load balancing such as: scheduling of tasks, migration, resource utilization, and so on. This study presented several approaches related to the vital challenge in Cloud computing which is load balancing. The problems associated with load balancing were discussed through comparative analysis of the proposed algorithms by researchers in the past six years. Several approaches were proposed, however, there are still exists some issues in the cloud environment such as migration of VMs, fault tolerance issues not fully addressed yet.

This review paper provided rich scope for researchers to develop intelligent and efficient load balancing algorithms for cloud environments. This study will be helpful for researchers to identify research problems related to load balancing, especially to further reduce the response time and avoid failures in the server as it included a summary of existing and available load balancing techniques.

6. Future work

In this paper, we described and reviewed numerous load balancing techniques in three major different environments: static, dynamic, and nature-inspired. Researchers can continue to further research on how to design algorithms that are more dynamic and intelligent and can solve fault tolerance issues to further enhance the performance of cloud computing. In the future, authors will aim to review nature-inspired and intelligent algorithms such as the application of machine learning or clustering techniques.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to send many thanks and gratitude to Taylor’s University, School of Computer Science & Engineering for giving us the great opportunity to carry out this research.

References

Abdalla, P.A., Varol, A., 2019. ‘Advantages to disadvantages of cloud computing for small-sized business. 7th Int Symp. Digit. Forensics Secur. ISDFS 2019 (October), 1–6, https://doi.org/10.1109/ISDFS.2019.8757549.
Abdullah, M., Ostman, M., 2012. An improved genetic algorithm for job scheduling in cloud. Procedia Inf. Technol. Comput. Sci. 2 (May), 291–296.
Adaniya, A., Paliwal, K., 2019. A Proposed Load Balancing Algorithm for Maximizing Response Time for Cloud Computing, Int. J. Res. Appl. Sci. Eng. Technol., 7(IV).
Adhikari, M., Angorning, T., 2018. Heuristic-based load-balancing algorithm for IaaS cloud. Futur. Gener. Comput. Syst. 81, 156–165. https://doi.org/10.1016/j.future.2017.10.035.
Afzal, S., Ganesh, K., 2019. A Taxonomic Classification of Load Balancing Metrics : A Systematic Review,” 33rd Indian Eng. Congr., no. January.
Afzal, S., Kavitha, G., 2019. Load balancing in cloud computing – A hierarchical taxonomical classification, J. Cloud Comput. Adv. Syst. Appl., 8(1). doi: 10.1186/s13677-019-0146-7.
Agarwal, M., Srivastava, D.G.M.S., 2017. Cloud Computing: A Paradigm Shift in the Way of Computing, Int. J. Mod. Educ. Comput. Sci. 9 (12), 38–48. https://doi.org/10.5815/ijmecs.2017.12.05.
Alam, M., Ahmad Khan, Z., 2017. Issues and challenges of load balancing algorithm in cloud computing environment. Indian J. Sci. Technol. 10 (25), 1–12. https://doi.org/10.17485/ijst/2017/v10i25/105688.
Alam, M.A., Elbashir, M.K., Osman, A.A., 2017. A load balancing algorithm to enhance the response time in cloud computing, J. Basic Appl. Sci. 2 (2), 473–490.
Ali, S. et al., 2020. Towards pattern-based change verification framework for cloud-enabled healthcare component-based. IEEE Access 8, 148007–148020. https://doi.org/10.1109/ACCESS.2020.3014671.
Aliyu, A.N., Souley, P.B., 2019. Performance analysis of a hybrid approach to enhance load balancing in a heterogeneous cloud environment. Int. J. Adv. Sci. Res. Eng. 5 (7), 246–257, https://doi.org/10.31695/ijarse.2019.33430.
Al-Marhabi, R., Haggag, M., Aboubtalib, A.E., 2014. Roulette wheel selection model based on virtual machine weight for load balancing in cloud computing. IOSR J. Comput. Eng. 16 (5), 65–70. https://doi.org/10.9790/0661-16576570.
Elngomi, K., Khanfar, Zahra Mohammed, 2016. A Comparative Study of Load Balancing Algorithms: A Review Paper. Int. J. Comput. Sci. Mob. Comput. 5 (6), 71–76. https://doi.org/10.5220/0006373006790691.

Fatima, S.G., Fatima, S.K., Sattar, S.A., Khan, N.A., Adil, S., 2019. Cloud computing and resource optimization for cloud computing environment. Int. J. Inf. Comput. Sci. 8 (1), 1863–1870 doi: 10.35940/ijrte.C5197.098319.

Hota, A., Mohapatra, S., Mohanty, S., 2019. Survey of Different Load Balancing Approach-Based Algorithms in Cloud Computing. Int. J. Comput. Technol. 10 (2), 189–209.https://doi.org/10.1109/INVENTIVE.2016.7830175.

Kaur, R., Prashar, T., 2015. Performance Analysis of Load Balancing Algorithms in Cloud Computing Environment–A Review. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 6 (4).

Kaur, S., Verma, D.B., 2018. A Review on Various Load Balancing Algorithms with Merits-Demerits in Cloud Computing. Int. J. Adv. Eng. Res. Dev. 5 (5).

Kaur, S., Ghumman, M.N.S., 2017. Allocation of Heterogeneous Cloudlets on Priority Basis in Cloud Environment. Int. J. Comput. Technol. 16 (3), 6240–6246. https://doi.org/10.32429/ijaret.2019.015.

Kaur, R., Luthra, P., 2014. Load balancing in cloud system using Max Min and Min Algorithm. Int. J. Comput. Technol. 5 (5), 468–475. https://doi.org/10.24297/ijct.v5i5.4942.

Karthika, K., Kanakambal, R.B.K., 2015. Load balancing algorithm review's in cloud computing environment. Int. J. Adv. Res. Comput. Sci. Softw. Eng., 6(4). doi.org/10.17148/IJARCCE.2015.4587.

Kapur, S., 2015. A survey on dynamic load balancing algorithms in cloud computing. Adv. Comput. Sci. Inf. Technol. 2 (7), 87–91. https://doi.org/10.13140/RG.2.2.15001.06247.

Kothari, K., Kharcha, A., 2015. Dynamic load balancing in cloud computing: A survey. Int. J. Adv. Comput. Sci. Appl. 6 (4), 299–308. https://doi.org/10.14569/IJACSA.2018.080462.

Kukreja, D., Prabhu, S., Shrivastava, V., 2019. Load Balancing in Cloud Computing. IOSR J. Comput. Eng. 16 (4), 23–28.https://doi.org/10.9704/iosr-jce.v16i4.27.

Kumar, A., Chawla, D.P., 2020. A systematic literature review on load balancing algorithms in cloud computing environment. Int. J. Comput. Technol. 16 (3), 6240–6246. https://doi.org/10.32429/ijaret.2019.015.

Kumar, R., Prashar, T., 2015. Performance Analysis of Load Balancing Algorithms in Cloud Computing Environment–A Review. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 6 (4).

Kundra, S., Verma, M.K., 2016. Load Balancing Techniques in Cloud Computing Environment–A Review. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 7 (6).

Kunchala, P., Mohandas, G., 2016. A Comparative Study on Load Balancing Algorithm with Partitioning of Public Clouds. Int. J. Adv. Comput. Sci. Appl. 9 (4), 376–381.

Kumar, R., Prashar, T., 2015. Performance Analysis of Load Balancing Algorithms in Cloud Computing Environment–A Review. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 6 (4).

Kundra, S., Verma, M.K., 2016. Load Balancing Techniques in Cloud Computing Environment–A Review. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 7 (6).

Kundra, S., Dixit, A., Dev, H., 2017. A Study on Various Load Balancing Algorithms for Response Time Reduction in Cloud. Int. J. curr. Eng. Res. Sci. 4 (10).

Kumar, K., Mahapatra, S., Mohanty, S., 2019. Survey of Different Load Balancing Approach-Based Algorithms in Cloud Computing: A Comprehensive Review. In: Behera H, Nayak J, Naik B, Abraham A (eds) Computational Intelligence in Data Mining. Advances in Intelligent Systems and Computing, 2019, vol. 711, pp. 99–110, doi: 10.1007/978-981-10-8055-5.

Kumar, R., Prashar, T., 2015. Performance Analysis of Load Balancing Algorithms in Cloud Computing Environment–A Review. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 6 (4).

Kundra, S., Dixit, A., Dev, H., 2017. A Study on Various Load Balancing Algorithms for Response Time Reduction in Cloud. Int. J. curr. Eng. Res. Sci. 4 (10).

Kundu, S., Pakrashi, P., 2020. A systematic literature review on load balancing algorithms of virtual machines in a cloud computing environment. Int. J. Comput. Sci. Eng. Educ. 18 (2), 174885/888 (37–47).
