Spring 5-18-2020

Load Balancing in Cloud Computing

Snehal Dhumal

Load Balancing in Cloud Computing

A Thesis

Presented to

Dr. Robert Chun

Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Snehal Dhumal

May 2020

The Designated Thesis Committee Approves the Thesis Titled

Load Balancing in Cloud Computing

By

Snehal Dhumal

Approved for The Department of Computer Science

San José State University

May 2020

Dr. Robert Chun	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science
Govind Rajan Chandra	Senior Software Engineer (Instacart Inc.)

Abstract

Cloud computing is one of the top trending technologies which primarily focuses on the end user's use cases. The service provider needs to provide services to many clients. These increasing number of requests from the clients are giving rise to the new inventions in the load scheduling algorithms. There are different scheduling algorithms which are already present in the cloud computing, and some of them includes the Shortest Job First (SJF), First Come First Serve (FCFS), Round Robin (RR) etc. Though there are different parameters to consider when load balancing in cloud computing, makespan (time difference between start time of first task and finish of last task on the same machine) and response time are the most important parameters. This research surveys different load balancing algorithms and aims to improve the SJF load balancing algorithm in cloud computing. In this project, a Modified Shortest Job First (MSJF) and Generalized Priority (GP) load scheduling algorithms are combined to reduce the makespan and optimize the resource utilization. Together, MSJF and GP sends the longest task having high MIPS (million instructions per second) requirements to the machine with a high processing power and the shortest task having low MIPS requirements to the machine with a low processing power. Hence, neither the task with the lowest MIPS requirements nor the task with the highest MIPS requirements needs to wait for a very long time for resource allocation. Every task gets fair priority. Results are shown for SJF, MSJF, and GP in order to compare the different number of tasks using cloud simulator.

Index terms - Cloud computing, resource management, load management, resource allocation, computation energy, virtual machine (VM).

Acknowledgments

I would like to thank my advisor Dr. Robert Chun for his continued support and providing me the guidance necessary to work on this project. Also, for teaching me core skills needed to succeed and reviewing my project. And, I would also like to thank my parents for their patience and advice, they gave me throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. Introduction

Cloud computing is a collection of different computer resource systems and services which can be easily configured and provided over the internet with reduced efforts. Cloud computing is distributed in nature. There are different aspects of the cloud such as speed, security, and privacy [1]. Thousands of users need to access the cloud at the same time; therefore, applications face many challenges at such times to balance the load among themselves. The entire system can even break due to overloading. Load balancing is the process of distributing cloud service requests to computing resources in a cloud environment. By using load balancing techniques, service providers can manage the incoming load or application requests by allocating resources between multiple servers, networks, or computers. Many scheduling algorithms exist to schedule tasks in the cloud computing. However, a problem with basic load balancing algorithms is that they do not use resources in an effective manner. This slows down the overall processing time in cloud computing.

Cloud computing is a membership-based administration which can be used by paying for their services [2]. This includes programming, platform, and framework kind of services [3]. Cloud computing can be thought of as the metaphor for the internet. In cloud computing, we can give networking services in someone else's hand which is proven to be a good way for cost-saving purposes. These services are classified as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Cloud services lessen the expense of the equipment and programming. As in cloud, applications can be hosted by service providers, data can be stored in service provider's data storages, and customers can access their applications and data from anywhere without worrying about storage space and data security. Hence, it can give divergent and adaptable services to customers. Arrangement of cloud technologies makes the business more

grounded; furthermore, this gives an opportunity to center around development and imagination. Cloud computing will give rise to the cutting-edge technology services [4] for the larger amount [5] of customers. Cloud computing is distributed in nature. There are different aspects of the cloud such as speed, security, and privacy [1]. Many users need to access a cloud at the same time. Due to this, applications face much more difficulty in managing this load at the same time. It might lead to an entire cloud system's collapse due to overloading. Hence, distribution of this load among different resources is necessary. Service providers can manage application demands by optimally allocating resources.

Cloud platforms are highly scalable. It can be scaled up and scaled down any time as per user's requirements. Such dynamic nature of cloud platforms demands efficient and effective load balancing across all machines to reduce makespan, the response time of a single task, energy consumption, and interruption of services. If load balancing is done properly among different cloud resources, then it also provides high availability of services, if any of the other resources are not responding properly. Many task scheduling techniques are present to schedule tasks in cloud computing. The main problem is that basic load balancing techniques do not use resources in an effective manner. Hence, it increases the overall request's processing time in cloud computing [5]. The virtualization capacity of the cloud computing conceals the diversity of the resources which keeps it unique in relation to other advancements presented already.

Load balancing can be defined as a set of rules which allocates a specific task to a specific virtual machine. Each task takes some execution time based on capacity of the virtual machine to which the task is allocated. Load balancing algorithms assign the tasks to suitable and available virtual machines to increase makespan and reduce response time of the task. The main research question is what parameters can be used in any load balancing technique? Also, is there any

particular parameter which helps the most in achieving best makespan? Many authors in their research tried to answer such questions. The purpose of this research is to analyze similarities and differences between different load balancing techniques proposed by researchers and derive a new approach for load balancing.

The research is organized as per the following topics:

- Overview of existing load balancing challenges in cloud computing
- Current load balancing techniques and their application methods
- Limitations and advantages of the current load balancing techniques
- Derivation of a new load balancing technique

This research looks at the associated work and investigates the load balancing techniques. In the first section, the concentration is on the load balancing metrics along with challenges, which additionally proceeds with the discussion of a few criteria which decide the adequacy of each technique. The last section investigates these techniques based on the measurements. Fig. 1. shows the organization of the research project.

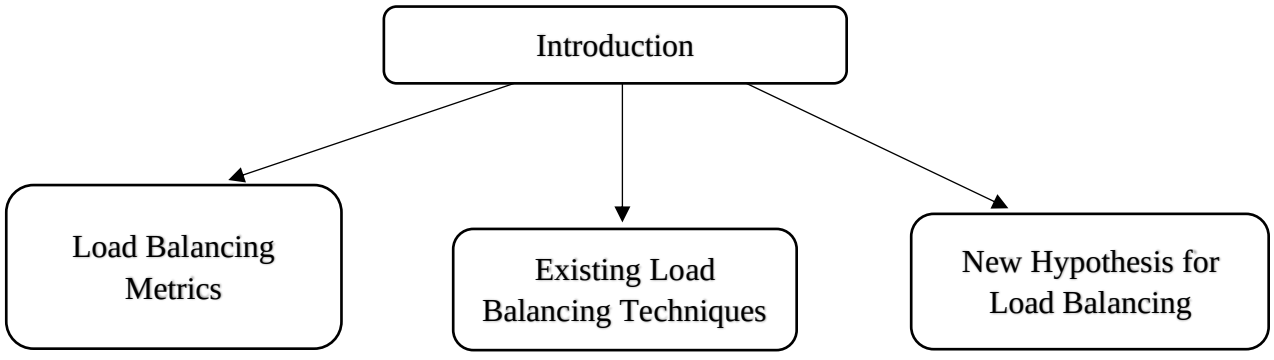

Fig. 1. Organization of Research

II. RESEARCH OBJECTIVE

The main purpose of this research is to improve the load balancing algorithms in the cloud computing which will improve the allocation of requests to virtual machines (VM) by using available resources in an optimal way. As a result, the processing time of tasks in the cloud computing will be reduced. Tasks should get allocated to each resource depending on its processing power and other availability to process the task. In RR, tasks are getting allocated to VMs based on first come first serve basis starting from some random VM. But, to improve performance task should start getting allocated in particular order of VM. This order might be depending on processing power of VM, priority of task or MIPS required by tasks to execute it.

Objective of this research project is to develop a load balancing algorithm which will take into consideration, the highest processing time required by task, the lowest processing time required by task and then adjust the allocation of these tasks to VMs as per the decided order to optimize the resource allocation and reduce the makespan.

This proposed load balancing should reduce the number of resources which are idle, thereby reducing the wait time of tasks to get allocated to particular VM. As a load balancer is considered as one of the building blocks of the cloud computing system, load balancing algorithm should aim to achieve equal distribution of the load to all available VMs. Along with considering the processing power of VMs, it should also consider MIPS required by each task to complete its execution.

One of the objectives is to show the difference between the results after implementation of different load balancing techniques like RR, FCFS and MSJF, to show the difference between resource allocation and makespan for each of the implementation.

III. Related Work

Basic load balancing is the process of distributing requests to resources in the cloud computing environment to optimize results. There are many constraints on previously proposed load balancing algorithms. The First Come First Serve (FCFS) algorithm is the simplest load balancing algorithm which considers only time used by the task to reach the VM [2]. The Round Robin (RR) load balancing algorithm is similar to FCFS with only one difference in which it gives the same time slots to all tasks. Research on previous load balancing algorithms, like Shortest Job First (SJF) and FCFS, shows that throughput was not improved much by these implementations [2].

RR load balancing is the most common and one of the least complex techniques. It has been used traditionally in many cloud computing environments. In RR technique, tasks are allocated to different resources based on time units [1]. This technique first chooses any node in the cloud system randomly and assigns first task to this node. This load balancing technique makes sure to make use of all available cloud resources in RR fashion. If all nodes are occupied at the time of task allocation, then a task is allocated in queue to the node containing the least number of tasks. Tasks get allocated with different cuts of time. This task allocation cycle proceeds in above fashion until all tasks get allocated to some node. The main disadvantages of RR are that it does not consider tasks' lengths, processing power of VM, or the priority of the task while allocating tasks to the VM. Hence, every node has equivalent weight during execution and is equally likely to be chosen while allocating the tasks. Hence, author in [14] states that RR algorithm is based on random sampling i.e. it selects load randomly. In this execution, some nodes are heavily loaded while some are lightly loaded with tasks. Hence RR does not optimize resource utilization [5].

In SJF load balancing technique, tasks are sorted based on the MIPS required by tasks to complete its execution. In [7], author mentions that one of the advantages of using SJF load

balancing technique is that it improves overall makespan and resource utilization. Shortest task finishes very quickly. But author also states that, this implementation makes longest task to wait indefinitely. Also, resources may not be used in efficient manner. As, VM with highest processing power may get assigning to task with lowest MIPS.

A few researchers have studied different load balancing algorithms which are based on priority. They have calculated a task's waiting time based on priority and then analyzed against FCFS and SJF. Waiting time of tasks in SJF is lower compared to FCFS. This leads to the fact that SJF is more efficient than FCFS [3]. Another group of researchers studied improved techniques as compared to SJF. They have modified SJF to MSJF in which average length of tasks is calculated and then task allocation to resources is done based on the task's length as compared to the calculated average length. This MSJF is implemented in this project, and the project tries to improve it by combining it with generalized priority scheduling. Alworafi and Dhari in [1] contributed to MSJF by elaborating the smallest task finishing time improvement. Thomas Yeboah tried to combine RR algorithm and SJF to reduce makespan and improve resource optimization.

Another group of authors in [10], discussed about GP load balancing technique. In this technique, priority is assigned to the VMs and tasks. Then, the task with highest priority get assigned to the VM with highest priority. One of the advantages of this technique is that the longest task executed on the VM having highest MIPS. But, this technique creates uneven distribution of load in resources. As, the task having highest MIPS with get executed first, the task having lowest MIPS needs to wait indefinitely.

An appropriate load balancing algorithm must have a few metrics that make it unique and valuable. These metrics ought to give a high response time and throughput. It should have adaptation to non-critical failure, versatility, elite and proficient resource usage, and low overhead.

In [3], the author considered throughput as the sending and accepting rates of information of the aggregate number of finished tasks on a given contribution at a given time unit. The higher the throughput, the better the execution of the cloud framework.

IV. Load Balancing Measures

Load balancing among different resources is the key issue for the cloud computing. Load balancing makes sure there is optimal resource usage by allocating tasks to available resources evenly. There must be some metrics to decide if the load balancing algorithm is optimal in terms of resource allocation and improving the makespan of tasks. These metrics generally determines the response time of each request and overall makespan of tasks. These metrics must be chosen properly to adapt load balancing algorithms to versatility, low overhead, and optimum and proficient resource allocation. Below are the few measures which are considered in analysis of load balancing algorithms.

1. Throughput:

   In [3], the author considered throughput as number of requests served in single time unit. If there is a higher throughput, cloud computing systems will have better performance. Any cloud framework should adapt to any failure while easily maintaining the same throughput. E.g. If any node fails in cloud framework, then framework should not fail. Load balancing algorithm should automatically assign the same request on a failing node to any other available node in the best possible way. This also helps maintain the throughput throughout the entire execution of all tasks. This system of allocating a task on a failing node to any other available node is referred to as high availability [2].

2. Adaptation to Internal Failure:

   Nonstop handling of users' requests without failure or pauses leads to highly available and successful systems. If any internal component of the system fails, then the system should divert its task to other available components without halting the execution process. This leads to higher throughput and high availability [29].

3. Response Time:

   Authors in [4,7] considered response time as time taken by cloud framework to execute and respond to users' requests.

4. Scalability:

   Cloud frameworks are generally distributed in nature. Distributed systems can be scaled horizontally by adding more nodes to satisfy more data storage and data processing requests. Any productive load balancing algorithm must have optimal resource usage to achieve high scalability. If frameworks keep on adding nodes to scale horizontally, but load balancing algorithms assigns requests to only few of the nodes, then it's of no use. Hence, any load balancing algorithm must adhere to some policies which will optimize the resource allocation to the request, and this will achieve maximum horizontal scalability.

5. Resource Utilization:

   The resources in the framework need to be considered. A productive load balancing algorithm must have higher and even resource usage [3].

6. Overhead:

   In cloud computing, to achieve high availability or horizontal scalability, requests might be redirected from one node to another node. Sometimes, this reallocation of requests might demand data movement from one node to another node even over the network. This movement of data or requests uses network bandwidth. Such kind of overhead must be considered in a load balancing algorithm as one of the important measures [6].

7. Performance:

Performance alludes to the viability of the framework after total execution of load balancing calculations. In the event that every single recorded parameter performs well, then it will amplify the execution of the whole framework [7].

V. Load Balancing Challenges in Cloud Computing

Cloud computing consists of a large number of resources. Research [17] makes a point that the organization of these resources and allocation of users' requests to particular resources requires genuine planning and proper formats. Hence, before diving directly into the design of load balancing techniques, one should consider situations and problems that might arise due to the design of the techniques. Many difficulties can be faced while designing the optimal load balancing technique. This research aims to discuss the difficulties to be considered while proposing the proper load balancing technique. Some design techniques are modeled only for the intranet where nodes can be firmly found, and correspondence delay can be avoided. Real difficulties are in the design of the load balancing technique, which manages the load across the network. Such techniques should function properly with the dispersion of cloud nodes to reduce the response time of individual requests, increase makespan, adapt to horizontal scalability, and achieve high availability.

Cloud systems generally consist of a large number of virtual machines. All virtual machines may have the same parameters in terms of bandwidth, processing power, memory, and storage. This setup is called homogeneous virtual machines. These parameters might differ depending on the structure of a cloud framework, which leads to heterogeneous virtual machines. The response time of each request submitted by users will depend upon the capacity of a virtual machine to which the request is allocated. As there can be a huge number of cloud users, tasks executed in cloud computing might be in large numbers. Some tasks need to preserve the precedence constraint while executing in the cloud environment. Load balancing techniques should consider all such prerequisites before allocating requests to the optimal virtual machine. In other words, a load

balancing technique is one of the important pillars of cloud computing systems, which plays a vital role in the overall performance improvement of the system.

The main challenge in designing an optimal load balancing technique is that it is very difficult to satisfy all the expectations of the users to improve each metric in load balancing. In [4], the author states that it is just a little tradeoff between different metrics to satisfy the particular requirements of cloud system users. Some users might be interested in less response time over maintaining the execution precedence of requests. On the other hand, some users might demand strict execution order as per the precedence of requests over one another. Some users are interested in improving the overall makespan of the system; likewise, others are interested in the high availability of the system. Hence, to achieve the best suitable results, one should do some tradeoff between available metrics and develop the best, suitable load balancing technique to improve the response time and optimize resource allocation.

There are various complex load balancing algorithms in terms of the time needed for their execution and the number of tasks they need to perform. These complex algorithms lead to extra processing time and a more complex process. Furthermore, implementation of such load balancing techniques demands more monitoring, communication overhead, and delays which leads to more bottlenecks which discards overall efficiency of the system. Load balancing techniques should be constructed considering worst case scenarios. For example, if a node fails, how will a load balancing technique handle such a node failure? If the task is taking more time than normal for execution, how will a load balancing technique migrate such a task on to another node if required?

VI. Load Balancing Techniques

Different load balancing techniques in cloud computing have been discovered to accomplish various tasks. These load balancing techniques are present with different complexity levels to achieve their purpose and to optimize the system. There are different goals of load balancing techniques which are discussed by authors in [8,9] which includes substantially improving performance of an algorithm in terms of time and complexity, keeping the system stable while executing, increasing flexibility of the system to make it adaptable for different failures etc. Depending on such different purposes, different load balancing techniques have been created. Basically, load balancing techniques are classified as shown in Fig. 2.

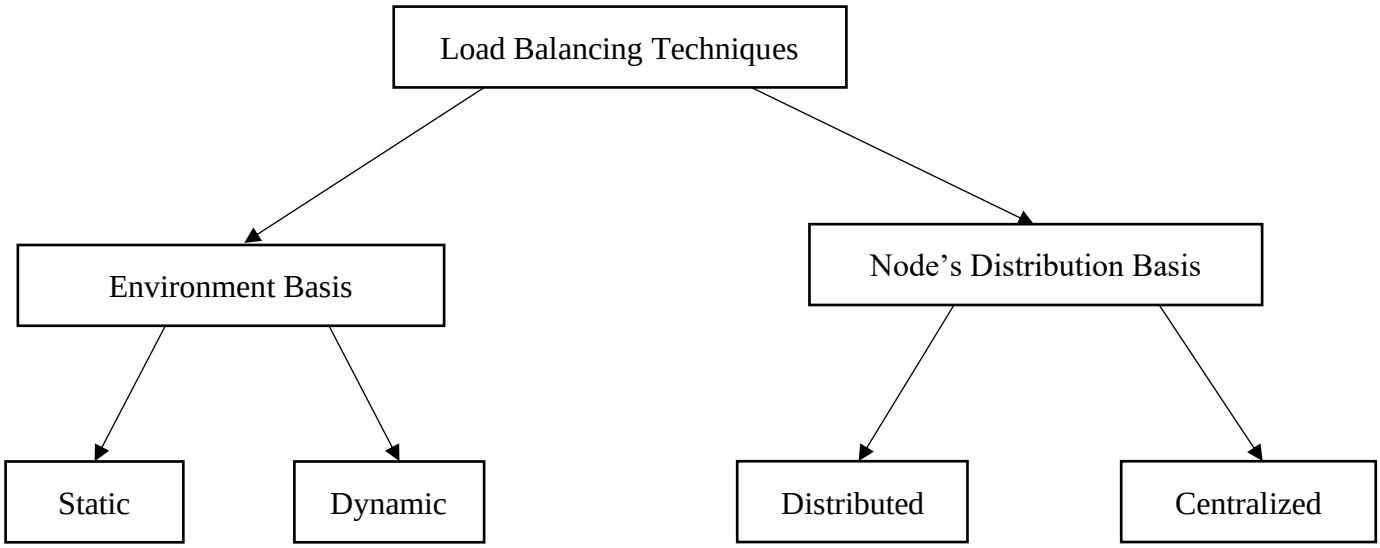

Fig 2. Load Balancing Techniques

In environment basis load balancing, load balancing algorithm depends on cloud environment. Environment includes VM's attributes like processing power, bandwidth etc. If these parameters remain constant, then the cloud system is static, and if we can change the attributes of the cloud environment then it is dynamic. In node's distribution type, load balancing technique depends

upon how many nodes are involved in handling the load. It is further classified into distributed and centralized load balancing technique.

In distributed load balancing technique, load balancing algorithm is executed by all the nodes in the system, and the task of load balancing is shared by all these nodes. Hence, load balancing task is equally distributed in the system. On the contrary, in centralized load balancing, single node is responsible for the execution of the load balancing algorithm, which is the central node. All other nodes interact with this central node. This central node is completely in charge of the load balancing of a whole system.

Static load balancing algorithms require prior knowledge about the system resources and tasks which are going to run on the system. In this technique, load balancing does not depend on the state of the system at the time of executing the task. Static load balancing technique manages the load across different nodes with some predefined set of rules related to attributes of input tasks. On the contrary, dynamic load balancing algorithms depend on the state of the system at the time of executing the tasks. It does not require prior knowledge about the attributes of input tasks. In dynamic load balancing, tasks can move from one utilized machine to another underutilized machine dynamically for faster processing [15].

In [16], the author added to this examination region by comparing these different load balancing techniques to one another to state advantages and disadvantages of each individual technique. There are different load balancing techniques which are involved in this comparison including RR, SJF, Min-Min load balancing, Min-Max load balancing etc. Author has additionally stated the correlation of these techniques. This research states that RR can distribute load based on the stated priority of the task, but in this technique, longer tasks need to wait for a really long time.

On the other hand, in Min-Min load balancing technique, completion time of tasks can be less, but it cannot predict task variations. Let's see some of the load balancing techniques in detail.

1. RR Technique

This technique of load balancing is the most common and one of the least complex techniques. It has been used traditionally in many cloud computing environments. In RR technique, tasks are allocated to different resources based on time units [1]. This technique first chooses any node in the cloud system randomly and assigns first task to this node. After this allocation, the next task will be allocated to the next available node and so on. This load balancing technique makes sure to make use of all available cloud resources in RR fashion. If all nodes are occupied at the time of task allocation, then a task is allocated in queue to the node containing the least number of tasks. Tasks get allocated with different cuts of time. This task allocation cycle proceeds in above fashion until all tasks get allocated to some node. The main disadvantages of RR are that it does not consider tasks' lengths, processing power of VM, or the priority of the task while allocating tasks to the VM. Hence, every node has equivalent weight during execution and is equally likely to be chosen while allocating the tasks. Hence, author in [14] states that RR algorithm is based on random sampling, i.e. it selects load randomly. In this execution technique, some nodes are heavily loaded while some are lightly loaded with tasks. Hence, RR does not optimize resource utilization. In case all tasks have the same length in terms of execution time and same MIPS and if all nodes have the same MIPS, then RR can provide the best result which is really a rare case in practical implementations [5].

2. SJF Technique

This technique aims to improve the two most important aspects of load balancing techniques, one is makespan and the other is response time. In SJF load balancing technique, all tasks are sorted

according to task length, i.e. MIPS in non-decreasing order. After sorting, each task is allocated to a node in RR fashion. The first node is selected randomly, and the shortest task is assigned to this node. After that, each task is allocated to the next available node in sorted order of task lengths. This process is repeated until all tasks are finished. The first advantage in this load balancing technique is that the shortest task is finished first. Hence, overall response time is improved. But, likewise with RR, resource utilization in SJF is also very poor. In this technique, the task with shortest length may get assigned to the VM with highest MIPS. Also, the task with highest length in terms of execution time and MIPS needs to wait for an indefinite time until all other tasks get allocated. SJF is also non-preemptive in nature, i.e. a task is not moved from one node to another node until its execution is finished. SJF will improve makespan of short tasks and the average response time of a system which will indirectly improve resource utilization. Though this resource utilization improvement is better than RR, it is still not the optimal resource utilization method.

3. Min-Min Load Balancing Technique

This technique computes minimum execution time for each node. Minimum completion time of each task is also computed. Once this is done, this technique selects the task with minimum execution time and assigns it to the node with minimum computing time. This process is repeated until all tasks get allocated to corresponding nodes for execution. This technique demands prior knowledge about task lengths in terms of execution time and processing power of each resource in the cloud system. This technique also reduces the makespan, but resource utilization is not improved. This technique selects small tasks to be finished initially which indefinitely delays the long tasks [6].

4. GP Algorithm

This load balancing techniques assigns priorities to the VMs and tasks based on some attributes. It chooses the length of the task and processing power of the VM, i.e. MIPS, as attributes to assign priority respectively [12]. The VMs are prioritized according to their MIPS value such that the virtual server with the highest MIPS value gets highest priority. The task with highest priority gets assigned to the VM with highest priority. In this technique, the longest task finishes first, and the shortest task has to wait indefinitely. Sometimes the longest task may get assigned to the VM with low processing power. Also, the author made a brief comparison of this technique with First Come First Serve (FCFS) and Round Robin (RR) scheduling algorithms. The author used a cloud simulator for the generalized priority algorithm and other algorithms to work in order to make an analysis between the proposed algorithm and the other algorithms.

VII. New Hypothesis for Load Balancing

A new model is proposed for improving makespan of VMs and optimizing resource utilization. As we mentioned previously, SJF improves makespan and resource utilization. However, it does not claim optimum resource utilization. The task with the longest length might have to wait for an indefinite time depending on availability of the VM. In the new proposed model, this SJF is modified to optimize resource utilization. This new hypothesis is referred to as MSJF because it is the modified shortest job first load balancing technique. In MSJF, tasks are sorted according to their length, i.e. computation time required by task in non-decreasing order. In addition to this sorting, in MSJF, average task length is calculated. If VMs have different processing powers, i.e. MIPS, then VMs are also sorted according to their processing powers. Roughly, average processing power of VMs is calculated in terms of MIPS. VMs with MIPS lower than the average processing power get assigned in one group, and VMs with MIPS higher than the average processing power get assigned in another group. Hence VMs are divided into two groups by their processing power. The key idea of this load balancing technique is that tasks with lengths below the average task length are assigned to VMs with low processing power, i.e. first group, and tasks with lengths higher than average task length are assigned to VMs with higher processing power [1]. By this technique, long tasks do not need to wait indefinitely. It assigns long jobs to fastest VMs. One disadvantage of this technique is that shortest task might get assigned to VM with high processing power. This load balancing technique improves makespan significantly with more even distribution of tasks among available VMs as compared to SJF. Experiments shows that this technique distributes task more evenly among all available VMs which indirectly optimizes the resource utilization as shown in Fig 5.

This MSJF load balancing technique is further improved by combining it with GP load balancing technique. In this combination, once VMs and tasks are divided into two different groups depending upon processing power and task length respectively, priority is assigned to each task and VM. Then, in these two different groups, the task with the highest priority gets assigned to the VM with the highest priority. In general, the VM with the highest MIPS gets the highest priority, and the task with the highest length gets the highest priority. Hence, the longest task does not need to wait for an indefinite time. It gets assigned to the VM with the highest MIPS, which improves the response time, makespan, and the resource utilization [12].

To implement this new load balancing technique, a cloud environment is simulated using CloudSim. VMs with different processing powers and tasks with different task lengths have been created. VMs are sorted by their MIPS, and tasks are sorted by their task lengths. Priorities are assigned to tasks and VMs depending on their length and processing power respectively. The algorithms for MSJF and MSJF+GP are mentioned below.

Proposed Algorithm for MSJF:

1. Simulate cloud computing system with heterogeneous resources' attributes.
2. Create cloud tasks called cloudlets with different random lengths.
3. Sort all cloud tasks in non-decreasing order.
4. Sort all VMs in non-decreasing order of processing power, i.e. MIPS.
5. Calculate the average length of task lengths.
6. Divide VMs in two groups depending on MIPS, i.e. VMs with processing power less than average processing power in one group and VMs having processing power higher than average MIPS in another group.

7. Send the task with a length below average task length to a VM in the first group, i.e. a VM with a low processing power in RR fashion.
8. Send the task with a length above average task length to a VM in another group, i.e. a VM with a higher processing power than the average processing power in RR fashion.
9. Repeat above process until all tasks have been processed.
10. Calculate the makespan and the total response time.

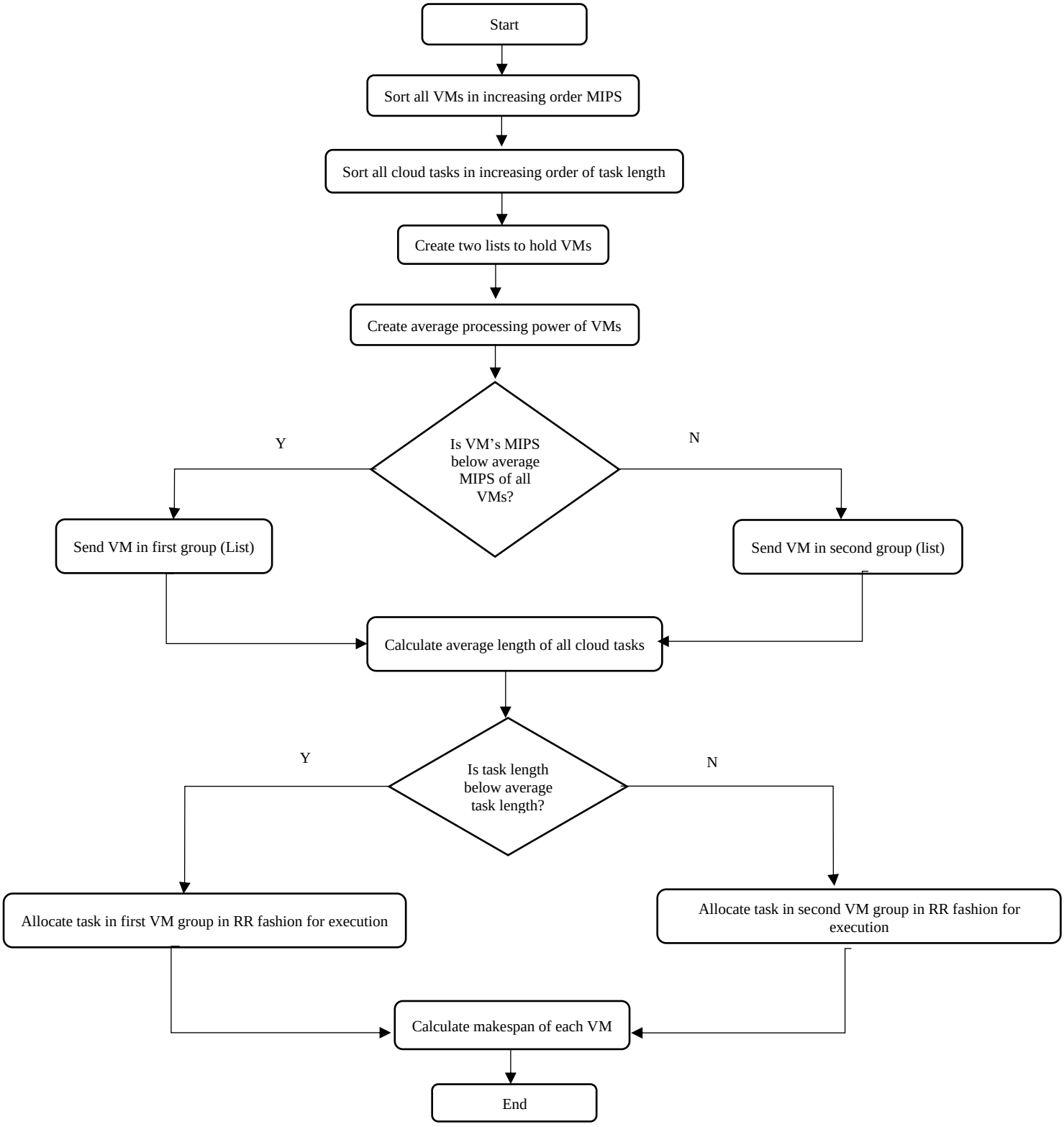

Fig. 3. Algorithm for MSJF Load Balancing

Experiments shows that this implementation improves overall makespan and VMs' utilization, but this implementation keeps the longest task waiting for a long time until all smaller tasks ahead of it are finished processing. In this implementation, the shortest job may get assigned to the longest machine unnecessarily, and the longest job may get assigned to a low processing power machine. Experiments showed that MSJF is further improved by combining it with the generalized priority algorithm for task scheduling. Both implementations are compared as shown in Fig. 4 and Fig. 5. Experiments are conducted by using same cloud environments as discussed below in results section. Detailed analysis of results showed that adding GP to MSJF has improved performance in terms of reduced makespan and even distribution of task among available VMs.

The main idea behind further improvement on MSJF is to combine it with the generalized priority load balancing algorithm. In MSJF, we have already divided cloudlets, i.e. tasks, according to their lengths, which are either below the average task length or above the average task length. Virtual machines are also divided according to their processing power. Now we will assign tasks in reverse order instead of normal allocation of MSJF. That means we will start assigning the last task to the last virtual machine, then the second last task to the second last virtual machine and so on. In that way, whatever tasks and virtual machines we have, the longest task among them will get allocated to the longest virtual machine. In this way, the longest task does not need to wait for a long time. Also, the shortest task will get assigned to a small machine, and whenever that machine finishes execution, next small task will get assigned to this low processing power virtual machine.

Algorithm for Generalized Priority:

1. Simulate cloud computing system with heterogeneous resources' attributes.
2. Prepare input with the different lengths of cloud tasks.

3. Sort all tasks in decreasing order of task lengths.
4. Sort all VMs in decreasing order of their MIPS.
5. Assign the highest priority to the task with the longest length and so on.
6. Assign the highest priority to the VM with the highest processing power and so on.
7. Send the task with the highest priority to the VM with the highest priority. In other words, send the longest task to the VM with the highest processing power.
8. Calculate the makespan and the total response time.

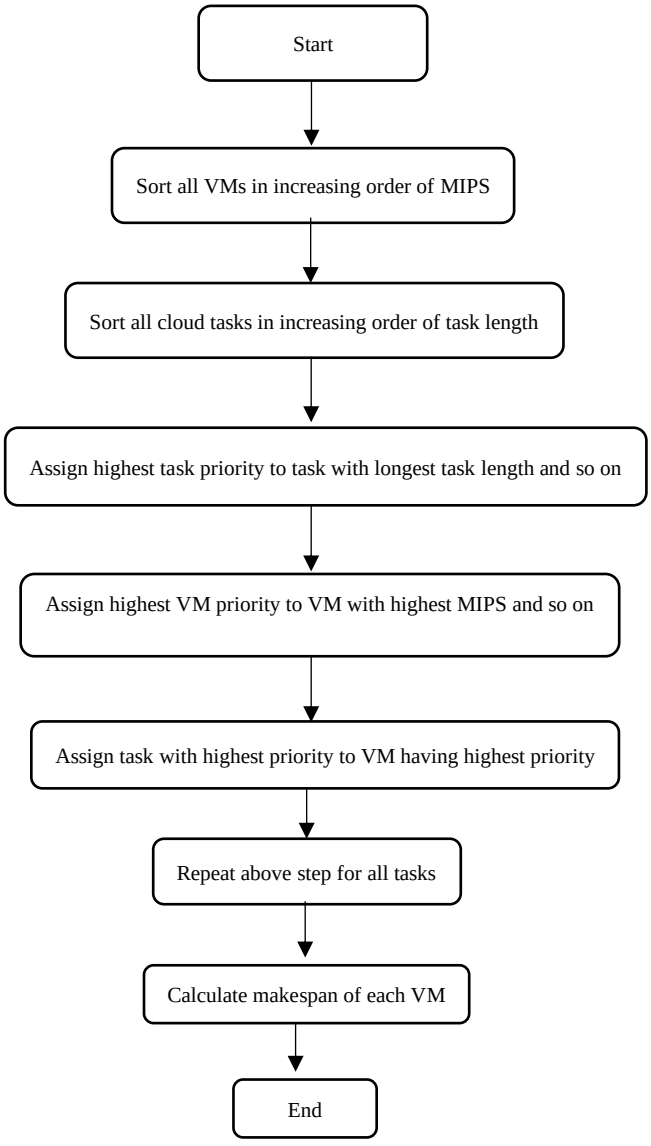

Fig. 4. Algorithm for GP Load Balancing

Proposed Algorithm for MSJF + Generalized Priority:

1. Simulate cloud computing system with heterogeneous resources' attributes.
2. Create cloud tasks called cloudlets with different random lengths.
3. Sort all cloud tasks in non-decreasing order.
4. Sort all VMs in non-decreasing order of processing power i.e. MIPS.
5. Divide VMs in two groups depending on MIPS. i.e. VMs with processing power less than average processing power in one group and VMs with processing power higher than average MIPS in another group.
6. Calculate the average length of task lengths.
7. Divide tasks with length below average task length in one group and tasks with length above average task length in another group of tasks.
8. Assign priority to VMs in decreasing order of their processing power i.e. MIPS.
9. Assign priority to tasks in decreasing order of their task length.
10. Assign the task with the highest priority to the VM with the highest priority in each group simultaneously.
11. Repeat above process until all tasks have been processed.
12. Calculate the makespan and the total response time.

Fig. 5. Algorithm for MSJF+GP Load Balancing

Overall, the time and space complexity of the above algorithm can be explained as follows. Let's consider that there are N number of tasks and M number of virtual machines. Hence, the sorting complexity of N tasks is O (N*log N). The space required to store N tasks is O(N). For sorting M virtual machines, we require O (M*log M) time. To store virtual machines, the space required is O(M). Hence, total time complexity is O (N * log N) + O (M * log M). As this time complexity is not quadratic, this is good time complexity as compared to O (N**2).

VIII. Implementation Details

CloudSim can be used to implement new proposed hypothesis. It is a stretchy framework for modeling and simulation of any application's performance in cloud environment on large scale. CloudSim is basically a library of cloud simulation scenarios. By considering policies of CloudSim, we can evaluate a method [11]. It is a Java-based cloud simulator. CloudSim allows users to configure and modify its architectural components like processing powers of virtual machines, task lengths of cloudlets etc. CloudSim has a data center which consist of different hosts distributed across a network. Cloudlets are assigned to different virtual machines for processing using a load balancing algorithm.

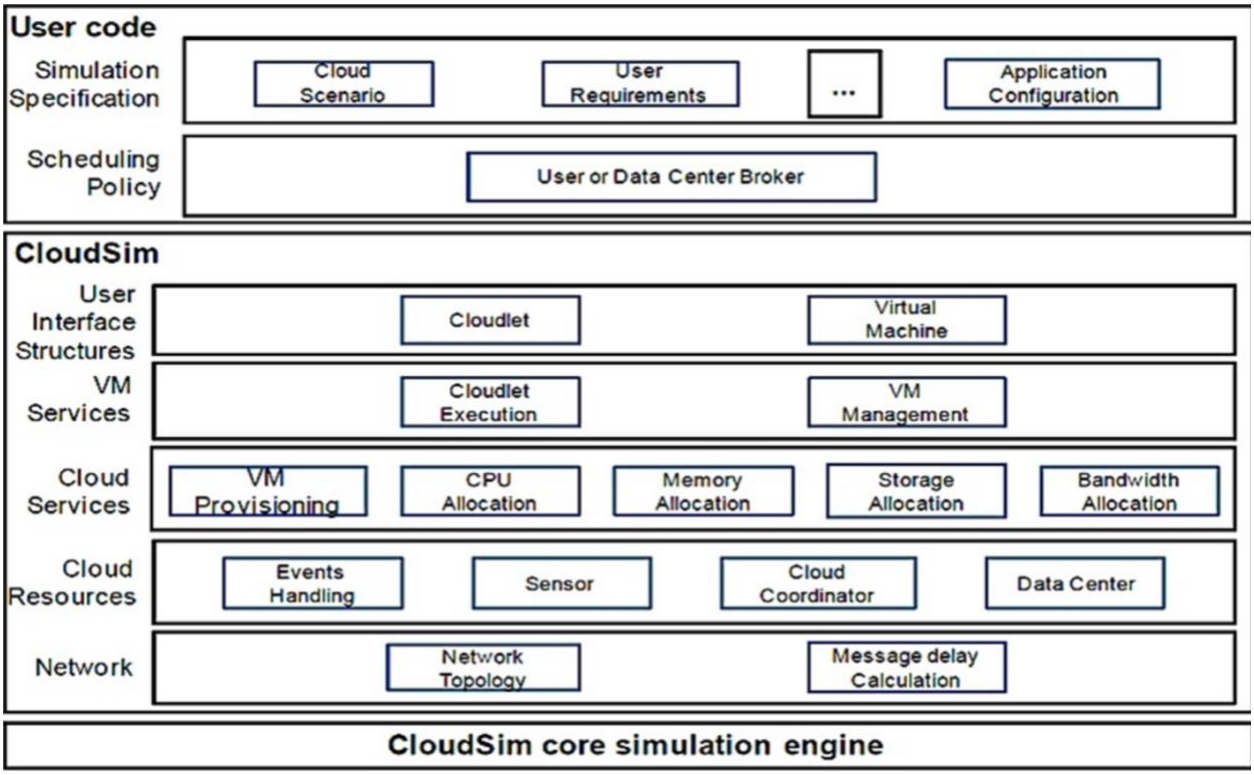

Fig. 6. CloudSim Architecture

CloudSim framework includes below mentioned components.:

- Regions

  Regions are geographically division to provide resources on different locations.

- Datacenter

  It is set of hosts or servers which provide infrastructure service. Datacenter configuration can be heterogeneous or homogeneous resources.

- Hosts

  It is the physical entity that is resource to tasks.

- Cloudlet

  It is set of requests from user for computation. This class in CloudSim designs the application services like content delivery etc.

- Service Broker

  It decides which VM will provide service to request.

- VM Allocation

  These policies in CloudSim model the allocation of resources to tasks.

IX. Results and Analysis

1. Configuration of Hosts:

Table 1. Configurations of Hosts

MIPS	Memory	Bandwidth	Storage
1000	16384 Mb.	10000 Mb.	1000000 Mb.

2. Configuration of VM

Table 2. Configuration of VM

VM No	MIPS	Memory	Bandwidth	Storage
VM_0	1000	512 Mb.	1000 Mb.	10000 Mb.
VM_1	4000	512 Mb.	1000 Mb.	10000 Mb.
VM_2	6000	512 Mb.	1000 Mb.	10000 Mb.
VM_3	1200	512 Mb.	1000 Mb.	10000 Mb.
VM_4	1000	512 Mb.	1000 Mb.	10000 Mb.

3. Assignment of Task to VMs using SJF

In SJF, all configurations on VMs have been kept as mentioned above. In this implementation, tasks are sorted according to task lengths, i.e. MIPS required by tasks to execute it. Once all tasks are sorted, then each task is assigned to the VM on a RR basis from sorted task list. Assignment of cloudlets to VMs is mentioned in the table and order below. After this task assignment table, this implementation of SJF is compared with default RR load balancing implementation.

Table 3. Assignment of Task to VMs using SJF

Cloudlet ID	VM ID	Start Time	Finish Time	Time
3	2	0.1	3.47	3.37
1	1	0.1	4.35	4.25
0	0	0.1	10.6	10.5
5	3	0.1	19.64	19.54
2	4	0.1	23.6	23.5
9	2	3.47	8.45	4.98
4	1	4.35	11.03	6.68
10	2	8.45	14.52	6.07
7	0	10.6	37.27	26.67
13	1	11.03	20.12	9.08
19	2	14.52	22.19	7.67
6	3	19.64	44.59	24.95
14	1	20.12	30.82	10.71
20	2	22.19	30.94	8.75
11	4	23.6	56.71	33.11
23	1	30.82	43.94	13.11
29	2	30.94	41.29	10.35
8	0	37.27	70.44	33.17
30	2	41.29	52.73	11.44
24	1	43.94	58.68	14.74
15	3	44.59	77.56	32.97
39	2	52.73	65.77	13.04
12	4	56.71	96.33	39.61
33	1	58.68	75.82	17.14
40	2	65.77	79.89	14.12
17	0	70.44	113.22	42.78
34	1	75.82	94.58	18.77
16	3	77.56	115.94	38.38
49	2	79.89	95.61	15.73
43	1	94.58	115.76	21.17
21	4	96.33	145.56	49.23
18	0	113.22	162.5	49.28

44	1	115.76	138.55	22.8
25	3	115.94	162.34	46.4
22	4	145.56	201.29	55.73
26	3	162.34	214.15	51.81
27	0	162.5	221.4	58.9
31	4	201.29	266.63	65.34
35	3	214.15	273.98	59.83
28	0	221.4	286.8	65.4
32	4	266.63	338.48	71.84
36	3	273.98	339.22	65.24
37	0	286.8	361.81	75.01
41	4	338.48	419.94	81.46
45	3	339.22	412.48	73.25
38	0	361.81	443.33	81.51
46	3	412.48	491.15	78.67
42	4	419.94	507.9	87.96
47	0	443.33	534.46	91.13
48	4	507.9	605.53	97.63

4. Makespan Using SJF

Makespan is the time measured from start of the first task till finish of the last task on the same VM.

Table 4. Makespan using SJF

	Makespan				
	VM-0	**VM-1**	**VM-2**	**VM-3**	**VM-4**
50 Tasks	534.46	138.55	95.62	491.15	605.53

In the figure below, default RR load balancing is compared to the SJF implementation. It can be seen that maximum and minimum makespan is drastically reduced.

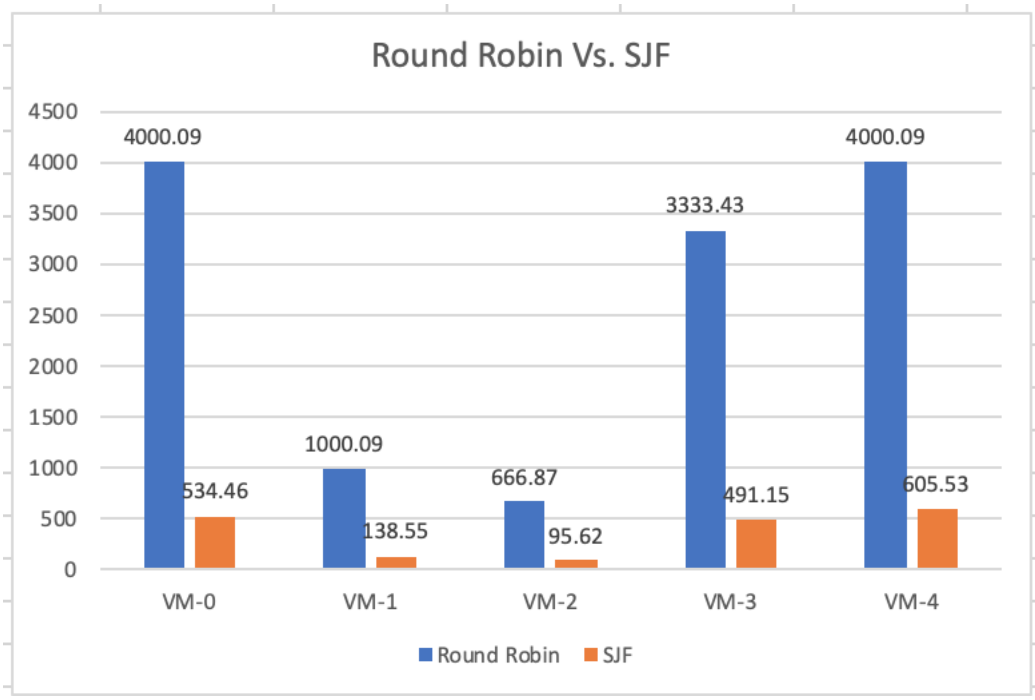

Fig. 7. Makespan of Each VM for Round Robin Vs. SJF

5. Assignment of Task to VMs using MSJF

In MSJF, all configurations on VMs have been kept as mentioned above. In this implementation, tasks are sorted according to task lengths, i.e. MIPS required by tasks to execute it. In this approach, the average of task lengths is calculated, and then VMs having processing power higher than the average processing power form one group, and those having processing power lower than the average processing power form another group. Tasks having length below the average length are assigned to a VM with low processing power, and tasks having length greater than the average length of cloudlets are assigned to a VM with high processing power in RR fashion from a sorted list of cloudlets. Assignments of cloudlets to VMs are shown in the able below.

Table 5. Assignment of Task to VMs using MSJF

Cloudlet ID	VM ID	Time	Start Time	Finish Time
27	2	9.82	0.1	9.92
0	0	10.5	0.1	10.6
1	3	14.17	0.1	14.27
24	1	14.74	0.1	14.84
3	4	20.28	0.1	20.38
29	2	10.35	9.92	20.27
2	0	23.5	10.6	34.1
5	3	19.54	14.27	33.8
26	1	15.54	14.84	30.38
31	2	10.89	20.27	31.16
4	4	26.72	20.38	47.1
28	1	16.35	30.38	46.73
33	2	11.43	31.16	42.59
6	3	24.95	33.8	58.76
7	0	26.67	34.1	60.77
35	2	11.97	42.59	54.55
30	1	17.16	46.73	63.89
9	4	29.89	47.1	76.99
37	2	12.5	54.55	67.06
11	3	27.6	58.76	86.35
8	0	33.17	60.77	93.94
32	1	17.96	63.89	81.85
39	2	13.04	67.06	80.1
10	4	36.39	76.99	113.39
41	2	13.58	80.1	93.67
34	1	18.77	81.85	100.61
12	3	33.01	86.35	119.37
43	2	14.11	93.67	107.79
13	0	36.34	93.94	130.27
36	1	19.57	100.61	120.19
45	2	14.65	107.79	122.44
15	4	39.56	113.39	152.95

17	3	35.65	119.37	155.02
38	1	20.38	120.19	140.57
47	2	15.19	122.44	137.63
14	0	42.84	130.27	173.11
49	2	15.73	137.63	153.35
40	1	21.18	140.57	161.75
16	4	46.06	152.95	199.01
18	3	41.07	155.02	196.09
42	1	21.99	161.75	183.74
19	0	46.01	173.11	219.12
44	1	22.8	183.74	206.54
23	3	43.71	196.09	239.8
21	4	49.23	199.01	248.24
46	1	23.6	206.54	230.14
20	0	52.51	219.12	271.63
48	1	24.41	230.14	254.54
22	4	55.73	248.24	303.97
25	0	55.68	271.63	327.3

6. Makespan Using MSJF

Makespan is the time measured from start of the first task till finish of the last task on the same VM.

Table 8. Makespan using SJF

	Makespan				
	VM-0	**VM-1**	**VM-2**	**VM-3**	**VM-4**
50 Tasks	327.30	254.54	153.35	239.79	303.96

In the figure below, the makespans of VMs for both the SJF and the MSJF implementations are compared to each other. It can be observed that the maximum and minimum makespan is reduced more than SJF implementation. Also, VMs utilization is done more evenly as shown in Fig. 8 below.

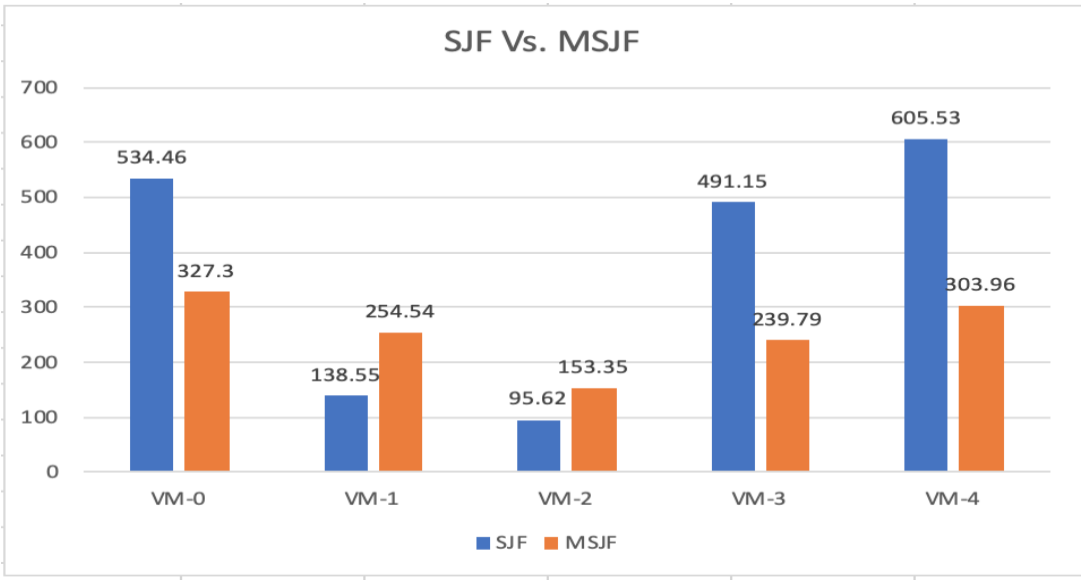

Fig. 8. Makespan of Each VM for SJF Vs. MSJF

7. Assignment of Task to VMs using MSJF + GP

In MSJF+GP implementation, all configurations on VMs have been kept as mentioned above. In this implementation, tasks are sorted according to task lengths, i.e. MIPS required by tasks to execute it. In this approach, the average of task length is calculated, and then VMs are divided into two groups of low processing power and high processing power according to MIPS. Tasks having length below the average length are assigned to a VM with low processing power and tasks having length greater than the average length of cloudlets are assigned to VMs with a high processing power. One more improvement that has been done in MSJF implementation is that cloudlets are not assigned as they come from a sorted list. Rather, priority is assigned to cloudlets, and a cloudlet with the maximum task length is assigned to the VM available with the maximum processing power from two groups. Hence, tasks with the longest length do not need to wait for an indefinite time. Assignments of cloudlets to VMs are shown in the table below.

Table 7. Assignment of Task to VMs using MSJF + GP

Cloudlet ID	VM ID	Time	Start Time	Finish Time
27	2	9.82	0.1	9.92
0	0	10.5	0.1	10.6
24	1	14.74	0.1	14.84
3	3	16.85	0.1	16.95
1	4	17	0.1	17.1
29	2	10.35	9.92	20.27
4	0	26.72	10.6	37.32
26	1	15.54	14.84	30.38
5	3	19.54	16.95	36.49
2	4	23.5	17.1	40.6
31	2	10.89	20.27	31.16
28	1	16.35	30.38	46.73
33	2	11.43	31.16	42.59
7	3	22.22	36.49	58.71
6	0	29.95	37.32	67.27
9	4	29.89	40.6	70.49
35	2	11.97	42.59	54.55
30	1	17.16	46.73	63.89
37	2	12.5	54.55	67.06
10	3	30.33	58.71	89.04
32	1	17.96	63.89	81.85
39	2	13.04	67.06	80.1
8	0	33.17	67.27	100.44
11	4	33.11	70.49	103.6
41	2	13.58	80.1	93.67
34	1	18.77	81.85	100.61
12	3	33.01	89.04	122.05
43	2	14.11	93.67	107.79
15	0	39.62	100.44	140.05
36	1	19.57	100.61	120.19
13	4	36.34	103.6	139.94
45	2	14.65	107.79	122.44

38	1	20.38	120.19	140.57
14	3	35.7	122.05	157.75
47	2	15.19	122.44	137.63
49	2	15.73	137.63	153.35
16	4	46.06	139.94	186
17	0	42.78	140.05	182.84
40	1	21.18	140.57	161.75
21	3	41.02	157.75	198.78
42	1	21.99	161.75	183.74
19	0	46.01	182.84	228.84
44	1	22.8	183.74	206.54
18	4	49.28	186	235.29
23	3	43.71	198.78	242.49
46	1	23.6	206.54	230.14
22	0	55.73	228.84	284.57
48	1	24.41	230.14	254.54
20	4	52.51	235.29	287.79
25	3	46.4	242.49	288.88

8. Makespan Using MSJF + GP

Table 8. Makespan using MSJF + GP

	Makespan				
	VM-0	**VM-1**	**VM-2**	**VM-3**	**VM-4**
50 Tasks	284.57	254.54	153.35	288.09	287.79

In the figure below, the makespans of VMs for both the SJF and the MSJF and the MSJF+GP implementation are compared to each other. It can be observed that the maximum and minimum makespan is reduced more than the MSJF implementation. Also, VMs utilization is improved.

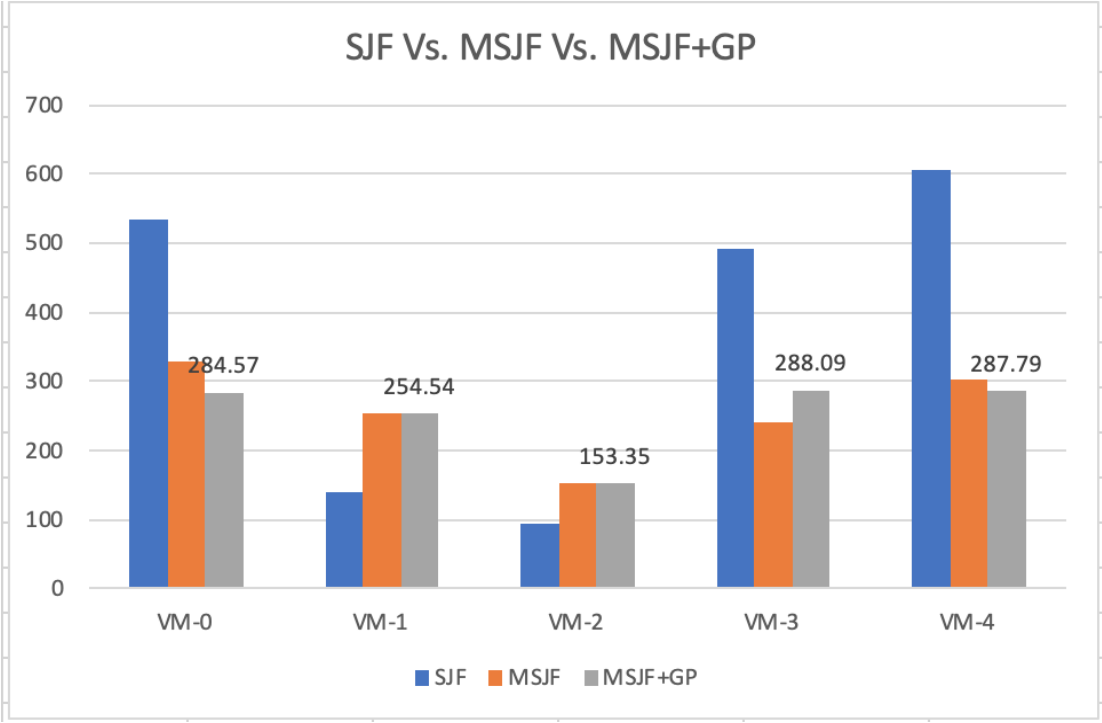

Fig. 9. Makespan of Each VM for SJF Vs. MSJF Vs. MSJF+GP

Comparison between Three Algorithms:

Table 9. Comparison between Three Algorithms

	SJF	**MSJF**	**GP**
Advantages	Small jobs finish first	Longest job to fastest machine	Task having highest size has the highest rank
Disadvantages	Longest job needs to wait for long time	Smallest job can be assigned to the slowest machine	Shortest job will get assigned late

Experiments and related work of different authors in load balancing had shown that the SJF is best suited when tasks are not varying too much in terms of length, and all available VMs are having same processing power. If tasks are varying too much in terms of task lengths, the SJF load balancing algorithm results in a higher makespan, as the longest task needs to wait indefinitely.

Experiments had also show that, MSJF is best suited when tasks lengths and VMs are varying in terms of MIPS. Combining GP and MSJF schedules the longest task on the fasted VM resulting in long tasks finishing in expected timespan. MSJF+GP implementation results in the same performance as RR, if VMs are having same processing power. As, assigning priority to the VM will not change any order of execution as all the VMs will receive the same priority.

X. Conclusion and Future Scope

Cloud computing is a very broad notion, and load balancing in cloud computing is one of the most important areas which needs to be addressed to improve the cloud users' experiences. Many different approaches are suggested by researchers and scientists to implement load balancing in cloud computing. However, none of them have addressed all the problems of load balancing. As stated in prior sections of this report, load balancing in cloud computing takes into account different parameters and tries to improve the performance of cloud systems based on those parameters. Some of these parameters are response time of machine to users' requests, makespan of VM, resource optimization, etc. As different users require optimization of different parameters, there is a vast scope of improvement in these load balancing techniques.

As seen, FCFS and RR just provide basic frameworks for load balancing in cloud computing. These techniques do not use resources optimally though they improve response time of individual requests by some amount of time. SJF improves resource utilization which is better than RR. MSJF further adds to this improvement by providing optimal resource utilization but still lacks in providing an accurate makespan. Combining MSJF with GP gives a better makespan and resource allocation. These are the factors considered in this research as parameters for improvement of load balancing techniques, but there are many other parameters.

Likewise, some other cloud users might be interested in bounding these request executions by deadlines of task execution time. Some users might want to prioritize tasks on basis of importance of request by users' perspectives. These are some areas which can be considered as future scope for load balancing research and can be explored more to improve the overall system performance for these parameters.

REFERENCES

[1] M. A. Alworafi, A. Dhari , A. A. Al-Hashmi , A. B. Darem, Suresha, "An improved SJF scheduling algorithm in cloud computing environment", Presented at Int. Conf. on *Elect., Electron., Commun., Comput. and Optimization Techn*., 2016.

[2] S. Abed, D. S. Shubair, "Enhancement of task scheduling technique of big data cloud computing", presented at Int. Conf. on *Advances in Big Data, Comput. and Data Commun. Syst*., Durban, South Africa, 2018, pp. 1-6.

[3] O. Kaneria, R. Banyal, "Analysis and improvement of load balancing in cloud computing", presented at Int. Conf. on *ICT in Business Industry & Government*, 2017.

[4] S. Abed, D. S. Shubair, "Enhancement of task scheduling technique of big data cloud computing", presented at Advances in Big Data Comput. and Data Commun. Syst., 2018, pp. 1-6.

[5] S. Wang, K.Q. Yan, W. P. Liao, SS Wang, "Towards a load balancing in a three-level cloud computing network", in Proc. 3rd IEEE Int. Conf. on *Comput. Science and Information Technology*, 2010, pp. 108–113

[6] E. K, M. Naghibzadeh, "A min-min max-min selective algorithm for grid task scheduling", presented at Int. Conf. in *Central Asia on Internet*,2007, pp. 1–7

[7] M. Randles, D. Lamb, T. Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing", presented at Int. Conf. in *Adv Inf Netw Appl Work*, pp. 551–556

[8] A. Nuaimi , N. Mohamed, A. Jaroodi, "A survey of load balancing in cloud computing: challenges and algorithm", in Proc - *IEEE 2nd Symp. Netw. Cloud Comput. Appl .NCCA*, 2012, pp. 137–142

[9] X. Shao, M. Jibiki, Y. Teranishi, N. Nishinaga , "Effective load balancing mechanism for heterogeneous range queriable cloud storage", presented at IEEE 7th Int. Conf. *Cloud Computing Technology*, 2015, pp. 405–412

[10] H. Sun, T. Zhao, Y. Tang, X. Liu, "A QoS-aware load balancing policy in multi-tenancy environment", Proc. - IEEE 8th International Symposium on Service Oriented Syst. Engineering, 2014, pp. 140–147

[11] E. Rani and H. Kaur, "Study on fundamental usage of CloudSim simulator and algorithms of resource allocation in cloud computing," 2017 8th International Conference on Comput, Comm and Networking Technologies (ICCCNT), Delhi, 2017, pp. 1-7.

[12] A. Hans and S. Kalra, "Comparative study of different cloud computing load balancing t

techniques," 2014 Intl Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom), Greater Noida, 2014, pp. 395-397.

[13] Nusrat Pasha, Dr. Amit Agarwal1 and Dr. Ravi Rastogi2, "Round Robin Approach for VM Load Balancing Algorithm in Cloud Computing Environment", ijarcsse, volume 4, issue 5, May 2014.

[14] Rajwinder Kaur and Pawan Luthra, "Load Balancing in Cloud Computing", Association of Computer Electronics and Electrical Engineers, ITC. , 2014

[15] Ms.Nitika, Ms.Shaveta, Mr. Gaurav Raj; "Comparative Analysis of Load Balancing Algorithms in Cloud Computing", IJARCET Volume 1, Issue 3, May 2012.

[16] Koushika, A. M., and S. Thamarai Selvi. "Load Balancing Using Software Defined Networking in cloud environment." *2014 International Conference on Recent Trends in Information Technology (ICRTIT),* IEEE, 2014.

[17] Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and . "CloudSim: a toolkit for modeling and simulation of cloud *Software: Practice and experience* 41.1 (2011): 23-50.

[18] S. Aslam and M. Shah, "Load Balancing Algorithms in Cloud Computing: A Survey of Modern Techniques," presented at the 2015 NSEC, Rawalpindi, Pakistan, Dec. 17, 2015, pp. 30-35, doi: 10.1109/NSEC.2015.7396341.

[19] S. Khara and U. Thakkar, "A Novel Approach for Enhancing Selection of Load Balancing Algorithms Dynamically in Cloud Computing," presented at the 2017 Comptelix, Jaipur, India, Jul. 1-2, 2017, pp. 44-48, doi: 10.1109/COMPTELIX.2017.8003935.

[20] S. Santra, K. Mali, "A New Approach to Survey on Load Balancing in VM in Cloud Computing: using CloudSim," presented at the 2015 Int. Conf. IC4, Indore, India, Sept. 10-12, 2015, pp. 1-5, doi: 10.1109/IC4.2015.7375671.

[21] V. Velde, B. Rama, "Simulation of Optimized Load Balancing and User Job Scheduling Using CloudSim," presented at the 2017 2nd IEEE Int. Conf. on RTEICT, Bangalore, India, May 19-20, 2017, pp. 1379-1384, doi: 10.1109/RTEICT.2017.8256824.

[22] S. Sonkar and M. Kharat, "A Review on Resource Allocation and VM Scheduling Techniques and a Model for Efficient Resource Management in Cloud Computing Environment," in *ICTBIG*, Indore, India, Nov. 18-19, 2016, pp. 1-7, doi: 10.1109/ICTBIG.2016.7892646.