International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6130
***Article Received**: 20 December 2022 **Revised**: 10 January 2023 **Accepted**: 20 January 2023*

Load Balancing in Distributed Cloud Computing: A Reinforcement Learning Algorithms in Heterogeneous Environment

Mrs. Minal Shahakar[1], Dr. Surenda Mahajan[2], Dr. Lalit Patil[3]
[1]Research Scholar
Smt. Kashibai Navale College of Engineering
Savitribai Phule Pune University, India
mhjn.minal@gmail.com
[2]Assistant Professor, Department of Information Technology
PVG' College of Engg & Tech & GK Pate (Wani) IOM
Savitribai Phule Pune University, India
sa_mahajan@yahoo.com
[3]Professor, Department of Information Technology
Smt. Kashibai Navale College of Engineering
Savitribai Phule Pune University, India
lalitvpatil@gmail.com

Abstract—Balancing load in cloud based is an important aspect that plays a vital role in order to achieve sharing of load between different types of resources such as virtual machines that lay on servers, storage in the form of hard drives and servers. Reinforcement learning approaches can be adopted with cloud computing to achieve quality of service factors such as minimized cost and response time, increased throughput, fault tolerance and utilization of all available resources in the network, thus increasing system performance. Reinforcement Learning based approaches result in making effective resource utilization by selecting the best suitable processor for task execution with minimum makespan. Since in the earlier related work done on sharing of load, there are limited reinforcement learning based approaches. However this paper, focuses on the importance of RL based approaches for achieving balanced load in the area of distributed cloud computing. A Reinforcement Learning framework is proposed and implemented for execution of tasks in heterogeneous environments, particularly, Least Load Balancing (LLB) and Booster Reinforcement Controller (BRC) Load Balancing. With the help of reinforcement learning approaches an optimal result is achieved for load sharing and task allocation. In this RL based framework processor workload is taken as an input. In this paper, the results of proposed RL based approaches have been evaluated for cost and makespan and are compared with existing load balancing techniques for task execution and resource utilization..

Keywords- Cloud Computing, Reinforcement Learning, Load Balancing, Heterogeneous System, Resource Allocation, Task Scheduling.

I. INTRODUCTION

Cloud computing relies on a well-oiled resource management system, the heart of which is scheduling resources. This term refers to the process of assigning available cloud resources to specific applications. Improved response time and reduced costs are the outcomes of this procedure, which seeks for the optimum resource and maps it with cloud workload depending on consumer requirements. The allocation of resources is a four-stage process. First, workloads are categorized by their needs and requirements. In the second stage, we choose out the precise collection of assets we need from our stockpile. The third phase involves assigning the proper cloud resources to user-specified cloud workloads in terms of quality of service. Finally, schedule the available resources to carry out the workloads, further ensuring that the QoS requirements will be met as closely as possible to ideal. Reinforcement learning based existing approaches face few problems such as complexity that occurs because of the huge memory space of state and activity. However, to resolve the problem, reinforcement learning has been combined with techniques such as deep learning in deep reinforcement learning. [2].

Load balancing is a very vital factor in the cloud environment, as the users have variety in their requirements. This leads to access use of resources like RAM and CPU usage leading to less performance. This application adds extra load to cloud servers and indirectly is responsible for lots of resource consumption. There are lots of algorithms that mainly deal with time for allocating connections. Some considering performance factors are load from the task on the server, waiting time,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6130
***Article Received**: 20 December 2022 **Revised**: 10 January 2023 **Accepted**: 20 January 2023*

response time, etc. [1]. To achieve reliable resource utilization, a load balancing method is proposed for allocating dynamically arriving customer requests that is to allocate connections to the cloud server regarding load as a parameter over time.

In distributed cloud computing, it is important to consider various factors of load balancing such as existing load of processors, memory utilization, resource utilization, etc. Some heuristic algorithms have been presented to optimize system performance in the cloud based networks. Since, few existing heuristics algorithms show deficiency in global and local search optimization [3]. In a distributed cloud network, a suitable processor is selected for upcoming tasks and necessary resources are allocated for its execution in minimum time. However, congestion in cloud networks can increase when the server is heavily loaded or tasks are too large and time consuming and also if required or necessary resources are unavailable on the server, this may result in increased makespan. Such problems can cause more energy to be consumed by the server if it is heavily loaded. To overcome this problem, in this paper a reinforcement load balancing method is presented which is based on task scheduling and resource utilization named as Booster Reinforcement Controller Load Balancing (BRC). Here we make use of reinforcement learning and load balancing methods to achieve optimal results.

However, Least load balancing (LLB) and Booster Reinforcement Learning (BRC) load balancing, both these approaches, are used to make a quick process of allocating available resources in the network to incoming tasks in order to reduce makespan and achieve maximum system throughput. Since, it also maintains the backup of existing load of server or processor before it is allocated with incoming tasks for execution so that if the task is executed on another processor with minimum load then the previous processor will be restored with its earlier load through this backup variable. Since, this allocation of tasks must be done in a way that no processor should get overloaded and nor should remain idle. To achieve optimal results for task allocation LLB and BRC load balancing techniques schedule the task execution on processor or server by considering few factors such as utilization of resources, reduced makespan and cost [3].

Few hybrid heuristics face challenges for balancing load in the cloud networks as it allows servers more prone to overload and so the QoS is not achieved [5]. However for smoother task execution and to improve the system performance, an optimal load balancing must be achieved. Thus, in this paper, we focus on achieving optimal results in allocating best suitable processors and necessary resources available in the network for task execution that maximizes throughput such that no processor should remain idle or overloaded and each processor is allocated a task with balanced load. However, in this paper we have implemented Least Load Balancing (LLB) and Booster Reinforcement Controller (BRC) load balancer approaches that help in evenly distributing load to processors across the networks. Both of these methods focus on achieving optimal throughput and system performance. Task Reallocation takes place by computing more resources and required time for task execution whenever server is heavily loaded and whenever server is lightly loaded with incoming tasks few computations of resources and required time for task execution are processed at a distinct time period. [6]. However when there is an increase in the number of incoming client requests may result in rapid increase in traffic congestion and also when servers are overloaded they require more energy. So, managing the network congestion is challenging in distributed cloud computing. When there is heavy load during peak hours, the network congestion not only harms intersections but also it harms upstream traffic. So, an effective Load Balancing (LB) is needed which distributes the task evenly among the available servers. This paper presents reinforcement learning with load balancing strategies that are more effective to achieve optimal results by maximizing the computational capacity that improves both the factors system performance and system feasibility. [8]. Because of variations in computation of heterogeneous cloud resources and time required for task execution in distributed cloud networks, the server load is extremely time dependent. Resource utilization status can be known in advance by using least load balancing reinforcement learning approaches to allocate the best suitable processor for task execution.

Scheduling cloud resources calls for allocating cloud assets to cloud tasks. It is possible to improve scheduling outcomes by treating Quality of Service (QoS) factors as essential constraints. However, efficient scheduling calls for improved optimization of QoS parameters, and only a few resource scheduling algorithms in the available literature do so. The primary objective of this paper is to provide an effective method for deploying workloads to cloud infrastructure. To ensure that workloads are executed efficiently on available resources, a load balancing method based on reinforcement learning was developed. The proposed method performance has been measured in the cloud. The experimental outcomes demonstrate the effectiveness of the suggested method in lowering the aforementioned QoS parameters along with the execution cost, time, and energy consumption.

The main contributions of this paper are summarized as follows:

- We have proposed RL based dynamic load balancing for data intensive networks. The BRCLB has been designed not only to improve both users satisfaction and fairness but also to

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6130
***Article Received**: 20 December 2022 **Revised**: 10 January 2023 **Accepted**: 20 January 2023*

maximize the long-term average network throughput

- The LLB Method is proposed to evenly distribute the task among servers to reduce network traffic congestion and also to maintain the backup of load that is present before task allocation of each server. So that if the server is not allocated with incoming requests its earlier load value will be replaced as a current load value of that server.
- The experimental results show that the proposed solution can achieve much better performance than existing solutions.

The remaining sections of the paper are organized as follows. Section II represents the scope of related work. Section III presents the proposed framework for task & resource allocation. Section IV describes the RL based booster reinforcement controller load balancer approach. Section V shows the experimental results. And section VI concludes the paper.

II. RELATED WORK

Existing load balancing policies can be roughly classified as static or dynamic, and as single-, bi-, or multi-objective. These regulations have centered on ensuring that machines aren't overloaded while also considering other load balancing considerations such make span time, energy usage, and resource utilisation. [4]. In order to solve this issue, scientists have created a number of load balancing algorithms for the cloud that are based on machine learning and RL. algorithms for optimizing cloud-based scheduling infrastructure. A Deep Q-network (DQN) is proposed to maximize long-term system performance by capitalizing on network latency, load balancing, and system stability; this is part of a Markov Decision Process (MDP) formulation that is designed to increase service quality while reducing costs. The results of this study contrast the optimization that takes into account only the current system performance when making mapping decisions for switch controllers with those that generate mapping decisions based solely on latency or load balancing separately, demonstrating that the DQN-based algorithm provides the best stability results while maintaining moderate switch controller latency and system equilibrium performance [10].

Based on their research in [7], Yu-Chieh Chuang and Wei-Yu Chiu propose a deep reinforcement learning based pricing strategy of an aggregator for profit maximization that takes into account the energy balance and can account for the actions of competitors as well as the variability of renewable and the varying bounds of charging and discharging events in a non stationary environment.

According to the proposal by Eunji Hwang et al. [11], while dividing up a system's limited resources among several users, it's crucial to keep in mind three factors: user fairness, system efficiency in terms of throughput, and user satisfaction in terms of reaction time. To complete the job in the allotted time, a heterogeneous computing system employs resource allocation policies tailored to multi-user and multi-application workloads. There are three distinct policies to choose from here: fairness, greedy efficiency, and fair efficiency. Based on the simulation findings, it is clear that the fair efficiency strategy is the optimal choice for allocating resources since it strikes a good compromise between justice and customer pleasure.

Weichao Ding et al. [9] introduced a workload predictor method based on the modified Weighted Moving Average (WMA) algorithm, which supports dynamic resource allocation; a cluster controller is proposed based on reinforcement learning for exploring the optimal matching relationship between resource requests and host at various PPR levels; a resource allocator is designed based on a greedy strategy for achieving the trade-off between energy consumption and application performance.

Computing load Aware and Long-View load balancing was proposed by Guoxin Liu et al. [6]. CALV selects the blocks that add the most workload during the server's busiest times and the least during the server';s least busy times. CALVuses a slow data block transfer mechanism to boost load balancing performance. It schedules data transfers to avoid taxing destination servers and make use of off-peak network bandwidth for reallocation. In comparison to other approaches, CALV is superior at increasing data locality and decreasing task delay, network load, and reallocation overhead.

To zero in on the network dynamics, the authors of [12] presented a deep reinforcement learning technique. To optimize the total payoff for automobiles, selfishness, a distributed coalition-based algorithm and an incentive system based on deep reinforcement learning are presented. To further lessen the computational load, a tailored transmit power adjustment approach is implemented. However, a feedback control mechanism driven by reinforcement learning (RL) is proposed for cooperative load balancing, which can help with a few job allocation issues (RF-CLB). To begin, by using RL and machine learning algorithms together, each edge can plan jobs and distribute them among neighboring edges based on its own local knowledge. The objective multiedge load balancing strategy for the Industrial Internet of Things can be launched with the help of feedback control and multiedge collaboration. [13].

To queue computational data packets that are being processed in chunks at one of the IoT nodes, and therefore distribute the data while making advantage of the unused bandwidth of local network links. While the ordered packets are being carried out one by one on the intended IoT device, the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6130
***Article Received**: 20 December 2022 **Revised**: 10 January 2023 **Accepted**: 20 January 2023*

remaining packets that are not being processed at the moment are dispersed and kept looping across the network links [17]

Transferring and re-using data packets, is handled via a time- synchronized packet deflection system on each node. This method ensures scalability of the temporary storage capacity of the connected IoT devices, necessitating data rates of 6 Mbps, while using only 45 Kb of primary storage systems even for big data.

Load balancing algorithms [18–21], reinforcement learning algorithms [22–25], and other methods have all been proposed to fix the scheduling and load issues plaguing cloud computing. Hypergraph Partition-based Scheduling was extended using a novel resource allocation technique provided by Laiping Zhao et al. HPS+ is an enhanced hypergraph partition technique to reduce WAN traffic by modeling the interdependencies between tasks and data as well as between data centers. To further reduce the makespan, it employs a coordinating system to distribute network resources in accordance with the principles of job needs. When compared to other algorithms, HPS+ speeds up production by up to 39% and reduces data transfer times by up to 53%, according to an evaluation conducted across the genuine China-Astronomy-Cloud model and the Google data center model.

Both the layered batch allocation by Jiuchuan Jiang et al. [27] and the core-based batch allocation by selecting core tasks to form batches can achieve suboptimal performance with lower complexity and significantly reduced computational cost. The former approach primarily uses the hierarchy pattern to form all possible batches, which can achieve better performance but may require higher computational cost since all possible batches are formed and observed. Better results can be achieved with these methods, both in terms of the overall amount paid by requesters and the average amount earned by employees, and in terms of the success rate at which tasks are completed and the amount of time spent allocating them.

Researchers Yinghao Yu et al. [28] have proposed creating chunks with storage codes and making several clones of hot files, also called hotspots, are examples of selective partitioning strategies that can be used to manage load imbalance. This method is inefficient because of the extra space it requires in memory to store the redundant data or the complexity of the encoding and decoding processes. SP-Cache is a cluster caching system for data-parallel clusters that uses load balancing and no redundancy. By carefully dividing popular files into numerous divisions, their read requests can be distributed among multiple servers. SP-Cache is able to efficiently reduce hotspots while minimizing the effects of laggards thanks to its periodic load balancing, but it is unable to respond quickly enough to short-term changes in popularity, such as sudden spikes in the number of requests for specific files.

It was determined that in dispersed cloud computing settings, it is crucial to build efficient load balancing algorithms for picking the correct resources and task scheduling. Scheduling approaches including single objective, bi-objective, and multi-objective scheduling have been the subject of extensive study because of their potential to improve resource allocation and job organization.

Since prior research has mostly concentrated on the objectives of processing time and cost or makespan, this work proposes the Least Load Balancing and (RL) Reinforcement learning approach for the task scheduling problem. However, in order to function in decentralized cloud computing, it is necessary to take into account a number of goals or requirements. Load balancing and reinforcement learning in a cloud-based distributed environment also needs to be taken into account. Improved cloud system performance can be achieved with the use of the Booster Reinforcement Controller method. While many algorithms excel in global search optimization, several of them struggle when it comes to local search.

Since the LLB method exhibits exploratory behavior, it has been shown in related research to be particularly useful in figuring out the task allocation problem. Booster RL can aid in this situation since learning can lead to better solutions. Therefore, we propose a method that can solve the task scheduling problem by employing the BRCLB algorithm and the LLB least load balancing algorithm to determine the tasks order for right resources that are available in environments and to find the most appropriate task or resource allocation solution. Since neither of these approaches leaves any CPU in the cloud system idle or overworked, throughput is maximized.

III. PROPOSED FRAMEWORK FOR TASK AND RESOURCE ALLOCATION

First, A crucial part of extending the life of a network is load balancing [18]. When nodes are overworked due to an ineffective task allocation strategy, the networks suffer. Furthermore, in a dispersed cloud network, each processor will work independently without an appropriate job allocation technique, which prevents all processors from cooperating in an energy-efficient manner. More vulnerabilities and uncertainties exist for real-time applications in cloud networks due to demanding features and constraints, such as environmental limits, the dynamic topology, and the instability of wireless link. The basic framework of cloud reinforcement learning is shown in Fig. 1, which contains three layers. These layers store the data related to load on each processor and give server statistics accordingly. The ability to send packets per minute to the same server known as per connection consistency. If not done, can lead packets to go to the wrong server that may lead to reset and timeout. However, our objective is to reduce fault tolerance and maximize resource utilization throughput. We

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6130
***Article Received**: 20 December 2022 **Revised**: 10 January 2023 **Accepted**: 20 January 2023*

need to ensure that servers get more, less or the same load, to implement uniform load balancing, as load balancing may receive millions of requests. However, the load balancer must be able to do end corrections on the server, and add additional servers if required. (Efficiency & Dynamicity). Based on each objective we found the proposed BRC load balancing algorithm helps us to achieve best optimization and throughput of the system.

The bottom layer represents the clients and the server clusters. Client generates a tcp packet to establish a connection, so the request goes to BRC load balancer, wherein proposed load balancer algorithm is used to allocate the task and resources. At the top layer, Once the server is selected, the packet is forwarded to the selected server, the response packet is sent and the data is encoded into a temp file that is done at the middle layer. Further, the temp file is sent to the client so when the client generates a new packet, a temp file is received through which, server id is extracted. This temp file contains server id as represented in (1).

$$temp = id \text{ xor } hash(4 \text{ tuple}) \quad (1)$$

The server id details are essential to identify server status so once the temp file is set with server id, this id is extracted from the file as represented in (2).

$$id = temp \text{ xor } hash(4 \text{ tuple}) \quad (2)$$

Once the id details are stored in a file it can be directly connected to the server, which allows packets not proceed through load balancer; so that task will be allocated to the same server for further processing, that is known as direct server return process. This entire concept is stateless, that can be made stateful (keep per connection on load balancer) and is applicable for developing NAT on load balancer, statistics, rate limiter.

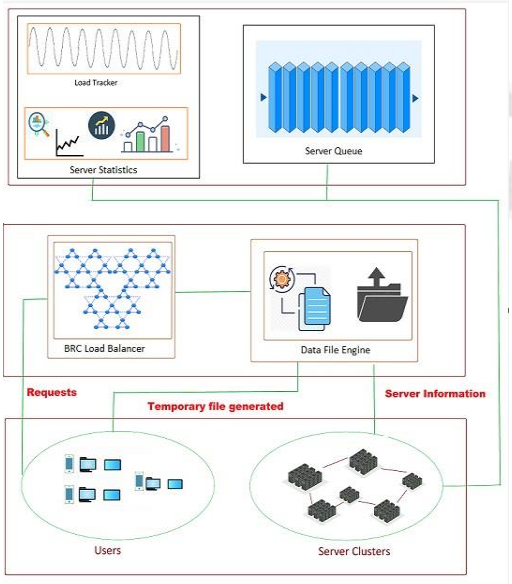

Figure 1. Cloud Reinforcement Learning Framework.

The main challenge in distributed cloud computing is balancing load equally among the available processors. Less response time, high throughput, improved fault tolerance, scalability, high user satisfaction, less heat generation, optimum power consumption and less operational cost can be achieved by optimal utilization of resources. [1]. Terms and meanings used in the proposed algorithm are shown in Table 1.

TABLE I. TEARMS AND ITS MEANING

Terms	Meaning
ϕ_{cap}	Capacity of Processor
ϕ_b	Bandwidth of Processor
ϕ_{pr}	Rate of Processing
ϕ_l	Load of Processor
ϕ_{tl}	Total load of Processor
ϕ_{avgl}	Average load of Processor
T_n	Number of task incoming
T_l	Length of task
T_s	Size of task
LB_{cpu}	Load Balancer Processor
T_{min}	Minimum Threshold
T_{max}	Maximum Threshold
ϕ_{curr}	Current Processor
bk_n	Backup load variable
ϕ_{am}	Allocated Processor
L_t	Total Load
S_b	Bandwidth of server
L_u	Memory in use
T[]	Array of task
t_l	Tasl load
T_{est}	Estimated Time

A. Changeover Mecahanism of Load Balancer

Static load balancing algorithms and dynamic load balancing algorithms are the two primary classifications of load balancing algorithms. Static Load Balancing allocates work to

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6130
***Article Received**: 20 December 2022 **Revised**: 10 January 2023 **Accepted**: 20 January 2023*

processing elements ahead of time, while dynamic Load Balancing allocates work as needed when an algorithm is running [8]. It is possible that a static mapping is the best option if the task's computing requirements are known in advance and do not vary during the computation. However, a static mapping might lead to a significant imbalance, making dynamic load balancing more effective [8] if the computing requirements are unknown before execution and can alter at run-time. The parallel with dynamic load balancing is simple: just as avalanches disperse sand across a lattice, so too may they balance the workload of queued activities in a distributed system. [29]. Static scheduling techniques are non-preemptive.

The changeover mechanism for Load Balancer (LB) is shown in Fig. 2. It shows that when a LB performs any action, it may switch to a different state. Now the states may vary depending on the possible outcomes. If LB receives any user request in the form of task for allocating it to server, then in the form of response LB calculates the current load of server in order to check the server status whether it is overutilized, underutilized so that task execution can be done faster within minimum time and also available resource will get fully utilized. Once the server load is calculated, LB sends load transfer instructions such as by which processor which task should be executed. However, if any request is not received by LB, it will go into a sleeping state triggering timeout. This mechanism explains the working of LB in order to check load status of different servers in distributed cloud networks. The changeover mechanism for load balancing where the applications can be dynamically tuned according to several objectives such as time, energy, communications, or their combinations. It adopts the method to the objective that provides a better use of the resources at any moment through a dynamic objective function, which can change over time [30]. It discusses where the workload is regular in between iterations; energy metrics are proven to be very useful and where the workloadis irregular in between iterations.

B. Capacity and Load of Processor

In distributed cloud computing, all the incoming requests in the form of tasks are assigned to a suitable Processor for execution within a short time period. Our proposed approach BRC Load balancer selects the best processor to execute the incoming request, that is dependent on the task length, processor capacity, and previous load of the processor. The bandwidth, CPU, memory and processing speed of a processor represents its capacity in (3). The load of the processor (ϕ) is determined by the task length and total number of tasks that processor already holds with respect to its capacity. The processor load is calculated in (4). The average load of all processors on a server is represented in (5) and (6), respectively.

$$\phi_{cap} = \phi_b + \phi_{pr} \quad (3)$$

$$\phi_l = \frac{T_n \times T_l \times T_s}{\phi_{cap}} \quad (4)$$

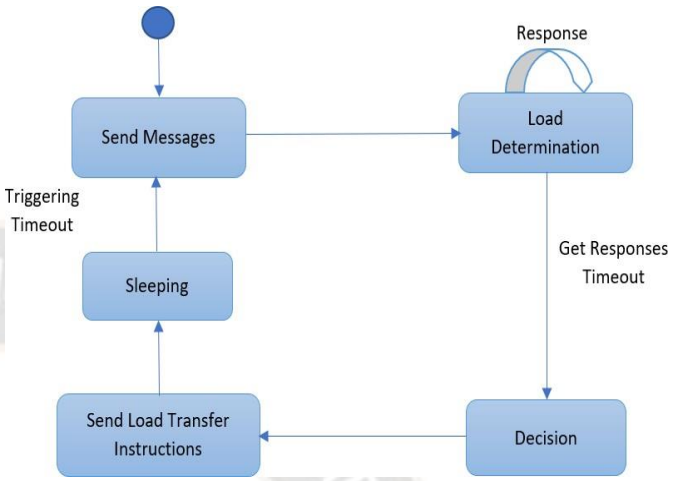

Figure 2. Changeover Mechanism of Load Balancer

$$\phi_{tl} = \sum_{p=1}^{0} \phi_l \quad (5)$$

$$\phi_{avgl} = \frac{\phi_{tl}}{\phi_p} \quad (6)$$

C. Finding Over-Utilized, Normal - UtilizedAnd Under-Utilized LB

The cloud computing environment is a network of distributed datacenters, wherein it consists of hundreds of servers. So, when a user submits a task, the datacenter controller handles it and makes use of load balancer. Further load balancer determines which machine should be allocated to the next upcoming request for processing [1]. The pseudo code of Over-Utilized, Normal - Utilized and Under-Utilized LB is described in Algorithm 1. It is used for balancing and distributing load equally among the available servers in distributed cloud networks. The LB Controller uses this algorithm to gather the data related to current load status of CPU before and after allocation of tasks for achieving better system throughput and response time. Total CPU Utilization can be calculated based on comparing it with minimum Threshold and maximum threshold values. Nonetheless, in order to decentralize the data and make use of the unused bandwidth of local network lines, packets of computational data that are being processed in chunks at one of the IoT nodes must be queued sequentially. While the ordered packets are being carried out one by one on the intended IoT device, the remaining packets that are not being processed now are dispersed and kept looping across the network links [17].

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6130
***Article Received**: 20 December 2022 **Revised**: 10 January 2023 **Accepted**: 20 January 2023*

Algorithm 1. Over-Utilized, Normal-Utilized And Under-Utilized LB

Begin
For each Load Balancer LB **do**
get LB_{cpu} utilization
initialize T_{min}
initialize T_{max}
if $LB_{cpu} < T_{min}$ **then**
    LB_{cpu} under-utilized
else
    LB_{cpu} rejected
End if
if $T_{min} < LB_{cpu} < T_{max}$ **then**
    LB_{cpu} normal-utilized
End if
if $LB_{cpu} > T_{max}$ **then**
    LB_{cpu} is over-utilized
End if
End For
Return new LB
End

IV. LOAD BALANCING ALGORITHMS

In the Least Load Balancer algorithm, every processor reports its current status of load to the load balancer. Each processor gives current load details that are utilized at the time of allocating requests to the best selected processor. If any processor fails then that task is allocated to the next suitable processor for processing so again changes in the task distribution process occurs.

A. LLB Algorithm

If all servers are running a given client IP address will always go to the same processor [1]. Through communication among active resource agents the load balancing is accomplished. Given that the set of processors ϕ, if no tasks are waiting in queue to be executed then processor Load ϕ_{ld} is by default set to null. However, when no tasks are running, it needs to calculate the load on the servers for further task allocation processes. [1]. The pseudo code for LLB Algorithm is described in Algorithm 2. However, if a task is allocated to n th processor then Processor load will be summation of current load and average load of that task, and if load condition is satisfied the best suitable processor will be selected and assigned for processing of task. But if the same task is allocated to another processor with minimum load as compared to the previous processor, then, this task T_m is assigned to that new processor. And, the previous processor will be restored with its earlier load that was there before calculation which is stored in backup variable bk_n , thus allowing task allocation to be done efficiently.

B. BRC Load Balancer Algorithm

To process the tasks, it is necessary to select the relevant processor to maximize the throughput of the system that is the primary goal of finding the optimal policy in distributed cloud computing.

Algorithm 2. LLB Algorithm

Begin
For each task T_m **do**
min = ∞
For each Processor **ϕ do**
Initialize Backup load bk_n
if $\phi_{ldn} + T_m <$ min **then**
    $\phi_{ldn} = \phi_{ldn} + T_m$
    min = ϕ_{ldn}
    $\phi_{am} = n$
Else
    current processor ϕ_{curr} do
    $\phi_{curr} = bk_n$
End if
End For
End For
Return Current load status
End

The load value depends on the selection of action in the state. Given the load value of each processor and available processors in the network for task execution, the algorithm is expected to select the best suitable processor and achieve maximum throughput. For every new request state of load will be initialized with a new value, as task processing is done continuously on each server. However simply doing this is not sufficient, to achieve the best throughput we need to do task allocation efficiently with reduced makespan and energy consumption, this is achieved by our proposed Boost Reinforcement Controller Load Balancer algorithm. The pseudo code for BRC Load Balancer Algorithm is described in Algorithm 3. To ensure that workloads are executed efficiently on available resources, a load balancing method based on reinforcement learning was developed. The proposed method performance has been measured in the cloud. The experimental outcomes demonstrate the effectiveness of the suggested method in lowering the aforementioned QoS parameters along with theexecution cost, time, and energy consumption.

The first objective proposed algorithm is to identify the time required to execute the task that is makespan. The task execution time is the time that the CPU needs to complete the decided set of tasks. Another objective includes to reduce the task completion time by selecting the most configured and optimized fastest processor for the task. In Algorithm 3, we represent the processor controller as LB, and the environment under which servers are running is termed as environment. LB

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6130
***Article Received**: 20 December 2022 **Revised**: 10 January 2023 **Accepted**: 20 January 2023*

performs task allocation and depending on the condition the best selected processor is assigned with further task processing. Important parameters of the algorithm are described as:

1) Server parameters L_t, S_b, L_u

The necessary parameters of the server such as total load bearing memory, bandwidth, and memory in use. C={L_t, S_b, L_u}

2) Load on CPU's Environment T[], t_l, T_{est}

This section contains all the information about how many tasks the processor is handling, whether it is underload or overload or balance. So here task count, task lengths and estimated time for task is

Algorithm 3. LLB Algorithm

1. Initialize Server parameters L_t, S_b, L_u
2. Initialize Processor load $\boldsymbol{\Phi}_{ld}$
3. **Begin**
4. **For** each new task T_n **do**
5. Initialize load balancer state lb_s
6. **For** each task T_n in queue **do**
7. execute LLB algorithm
8. **if** $\boldsymbol{\Phi}$ == underload
   allocate task T to $\boldsymbol{\Phi}$
10. **End if**
11. Update Load parameters T[], t_l, T_{est}
12. Update Server parameters L_t, S_b, L_u
13. Change Load Balancer State
14. Generate temporary file F
15. **End For**
16. **End For**
17. Return F
18. **End**

maintained. The task estimation may vary with processors due to different processing power.

L={T[], t_l, T_{est}}

3) Changeover mechanism

It shows that when a LB performs any action, it may switch to a different state. Now the states may vary depending on the possible outcomes. Fig. 2 explains the states.

V. EXPERIMENTAL EVALUATION

Our new LLB and BRC Load Balancer algorithm is compared to four other algorithms in this section: MOCS, FCFS, minmin, and maxmin. Our simulation findings suggest that our proposed approach outperforms state-of- the-art alternatives. Minimizing Load Balancer settings and effectively managing processor load are two of its main features. The LLB algorithm's outputs are fast and accurate, too, which improves the efficiency of our system.

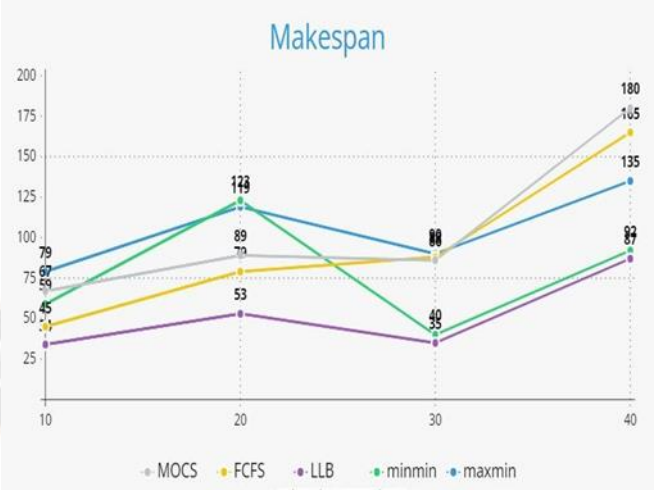

Figure 3. Comparison of makespan for different task

The result displayed in Fig. 3 is obtained by different types of algorithms with different numbers of tasks assigned to available processors. From this figure, compared to the other four algorithms our proposed algorithm takes less computation time.

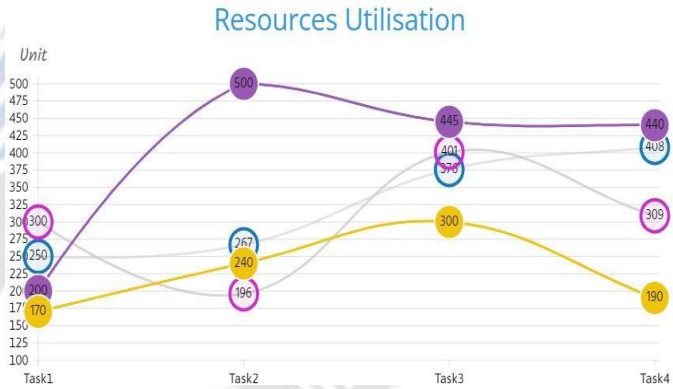

Figure 4. Memory Utilization for different set of tasks

Memory utilization is shown in Fig. 4 for different sets of tasks assigned to best suitable processors for execution with the BRC Load Balancer algorithm. Our proposed BRC Load Balancer algorithm makes use of minimum resources for processing the task execution. From the figure, when the number of processors increases, the processing speed also increases. BRC Load Balancer shows a great improvement in speedup as compared to other existing methods, as BRC Load balancer makes use of minimum resources for task execution so more number of task processing can be done effectively. This proves that BRC Load Balancer is very suitable to be implemented in distributed cloud computing.

VI. CONCLUSION

The Booster Reinforcement Controller Load Balancer task scheduling algorithm is a distributed cloud computing load

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6130
***Article Received**: 20 December 2022 **Revised**: 10 January 2023 **Accepted**: 20 January 2023*

balancing algorithm that is presented in this paper. This algorithm is based on both the multiple load balancer PSO algorithm and the LLB algorithm. When the next task is assigned, the processor can be brought back to its prior state thanks to the LLB Backup of load, which is kept. Finding the Over-Utilized, Normal-Utilized, and Under-Utilized loads of the various processors that are available reduces the overall processing time needed to complete a task. In a distributed cloud setting, this load balancing algorithm is suggested to optimize various load balancing parameters with the least amount of time required as compared to other existing load balancing algorithms. The local environment aids in the completion of our simulation experiment. Two distinct experiments are run in the simulation to determine percentage, accuracy, and process speedup. When the multiple task set is distributed to different processors, the LLB & BRC Load Balancer method enhances server settings when compared to other load balancing techniques now in use. This information leads to the conclusion that our suggested approach performs better even when a large number of tasks are brought into the datacenter. We will contrast our suggested approach with other meta-reinforcement learning techniques in the future. A hybrid approach that combines swarm optimization and machine learning techniques has also been presented to enhance the ideas of resource management and resource allocation. Real-time analytics on the intricate and dynamic cloud network will be provided by this algorithm.

REFERENCES

[1] Mrs. Minal Shahakar, Dr. S. A. Mahajan, Dr. Lalit Patil, "Assignment Of Independent Tasks Based on Load Balancing in Distributed Cloud Systems", *J. Harbin Inst of Tech.*, vol. 54, pp. 301-311, 2022.

[2] Sunghwan Kim, Seunghyun Yoon, and Hyuk Lim, "Deep Reinforcement Learning-Based Traffic Sampling for Multiple Traffic Analyzers on Software-Defined Networks", *IEEE Access*, vol. 9, pp. 47815-47827, 2021.

[3] Warangkhana Kimpan, "Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning", *IEEE Access*, vol. 10, pp. 17803-17818, 2022.

[4] Arabinda Pradhan, Sukant Kishoro Bisoy, Sandeep Kautish, Muhammed Basheer Jasser, And Ali Wagdy Mohamed, "Intelligent Decision-Making of Load Balancing Using Deep Reinforcement Learning and Parallel PSO in Cloud Environment", *IEEE Access*, vol. 10, pp. 76939-76952, 2022.

[5] Rizwana Ahmad (Member, Ieee), And Anand Srivastava (Member, Ieee), "Sequential Load Balancing for Link Aggregation Enabled Heterogeneous LiFi WiFi Network", *J. Vehicular Technology*, vol. 3, pp. 138-148, 2022.

[6] Guoxin Liu, Student Member, IEEE, Haiying Shen, Senior Member, IEEE, and Haoyu Wang, Student Member, IEEE, "Towards Long-View Computing Load Balancing in Cluster Storage Systems", *Ieee Transactions On Parallel And Distributed Systems*, VOL. 28, pp. 1770-1784, 2017.

[7] Yu-Chieh Chuang and Wei-Yu Chiu , Member, IEEE, "Deep Reinforcement Learning Based Pricing Strategy of Aggregators Considering Renewable Energy", *Ieee Transactions On Emerging Topics In Computational Intelligence*, VOL. 6, pp. 499-508, 2022.

[8] Andrea Giordano, Alessio De Rango, Rocco Rongo, Donato D'Ambrosio, and William Spataro, "Dynamic Load Balancing in Parallel Execution of Cellular Automata", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 32, No. 2, 2021.

[9] Weichao Ding , Fei Luo , Chunhua Gu, Haifeng Lu, And Qin Zhou, "Performance-to-Power Ratio Aware Resource Consolidation Framework Based on Reinforcement Learning in Cloud Data Centers", *IEEE Access*, vol. 8, pp. 15472-15483, 2020.

[10] Jia Chen, (Member, Ieee), Shihua Chen, Xin Chengand Jing Chen, (Graduate Student Member, Ieee), "A Deep Reinforcement Learning Based Switch Controller Mapping Strategy in Software Defined Network", *IEEE Access*, vol. 8, pp. 221553-221567, 2020.

[11] Eunji Hwang, Suntae Kim, Tae-kyungYoo, Jik-Soo Kim, Soonwook Hwang, and Young-ri Choi, "Resource Allocation Policies for Loosely Coupled Applications in Heterogeneous Computing Systems", *Ieee Transactions On Parallel And Distributed Systems*, Vol. 27, pp. 2349-2362, 2016.

[12] Yalan Wu , Jigang Wu , Member, IEEE, Long Chen , Jiaquan Yan, and Yinhe Han, "Load Balance Guaranteed Vehicle-to-Vehicle Computation Offloading for Min-Max Fairness in VANETs", *Ieee Transactions On Intelligent Transportation Systems*, VOL. 23, pp. 11994-12013, 2022.

[13] Xing Chen , Member, IEEE, Junqin Hu, Zheyi Chen , Bing Lin, Naixue Xiong , Senior Member, IEEE, and Geyong Min , Member, IEEE, "A Reinforcement Learning-Empowered Feedback Control System for Industrial Internet of Things", *Ieee Transactions On Industrial Informatics*, VOL. 18, pp. 2724-2733, 2022.

[14] Li Shi, Zhemin Zhang, and Thomas Robertazzi, "Energy-Aware Scheduling of Embarrassingly Parallel Jobs and Resource Allocation in Cloud", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 28, 2017.

[15] Dazhao Cheng, Jia Rao, Yanfei Guo, Changjun Jiang, and Xiaobo Zhou, "Improving Performance of Heterogeneous MapReduce Clusters with Adaptive Task Tuning", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 28, 2017.

[16] Zhiyao Hu , Dongsheng Li, Dongxiang Zhang , Yiming Zhang , and Baoyun Peng "Optimizing Resource Allocation for Data-Parallel Jobs Via GCN-Based Prediction", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 32, 2021.

[17] Anandarup Mukherjee, Pallav Kumar Deb, and Sudip Misra, "Timed Loops for Distributed Storage in Wireless

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6130
***Article Received**: 20 December 2022 **Revised**: 10 January 2023 **Accepted**: 20 January 2023*

Networks", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 33, 2022.

[18] Wenzhong Guo, Jie Li, Guolong Chen, Yuzhen Niu, and Chengyu Chen, "A PSO-Optimized Real-Time Fault-Tolerant Task Allocation Algorithm in Wireless Sensor Networks", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 26, 2015.

[19] Dazhao Cheng, Xiaobo Zhou, Yu Wang and Changjun Jiang, "Adaptive Scheduling Parallel Jobs with Dynamic Batching in Spark Streaming", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 29, 2018.

[20] Myeonggyun Han , Jinsu Park , and Woongki Baek, "Design and Implementation of a Criticality and Heterogeneity-Aware Runtime System for Task-Parallel Applications", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 32, 2021.

[21] Renyu Yang, Chunming Hu, Xiaoyang Sun, Peter Garraghan , Tianyu Wo, Zhenyu Wen, Hao Peng , Jie Xu, "Performance-Aware Speculative Resource Oversubscription for Large-Scale Clusters", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 31, 2020.

[22] Minghao Ye , Yang Hu , Junjie Zhang , Member, IEEE, Zehua Guo , Senior Member, IEEE, and H. Jonathan Chao , Life Fellow, IEEE, "Mitigating Routing Update Overhead for Traffic Engineering by Combining Destination-Based Routing With Reinforcement Learning", *Ieee Journal On Selected Areas In Communications*, vol. 40, 2022.

[23] Rizwana Ahmad, (Student Member, Ieee), Mohammad Dehghani Soltani, Majid Safari, (Member, Ieee), Anand Srivastava, (Member, Ieee), And Abir Das, "Reinforcement Learning Based Load Balancing For Hybrid LiFi WiFi Networks", *IEEE Access*, vol. 8, pp. 132273-132284, 2020.

[24] Ankit Shah , Rajesh Ganesan , Sushil Jajodia , Fellow, IEEE, Pierangela Samarati , Fellow, IEEE, and Hasan Cam, Senior Member, IEEE, "Adaptive Alert Management for Balancing Optimal Performance among Distributed CSOCs using Reinforcement Learning", *Ieee Transactions On Parallel And Distributed Systems*, Vol. 31, pp. 16-33, 2020.

[25] Qingzhi Liu , Tiancong Xia, Long Cheng , Senior Member, IEEE, Merijn van Eijk, Tanir Ozcelebi , and Ying Mao , Member, IEEE, "Deep Reinforcement Learning for Load-Balancing Aware Network Control in IoT Edge Systems", *Ieee Transactions On Parallel And Distributed Systems*, Vol. 33, pp. 1491-1502, 2022.

[26] Laiping Zhao, Yanan Yang, Ali Munir, Alex X. Liu, Yue Li, and Wenyu Qu, "Optimizing Geo-Distributed Data Analytics with Coordinated Task Scheduling and Routing", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 31, pp. 279-293, 2020.

[27] Jiuchuan Jiang, Bo An, Yichuan Jiang, Senior Member, IEEE, Peng Shi, Zhan Bu, and Jie Cao, "Batch Allocation for Tasks with Overlapping Skill Requirements in Crowdsourcing", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 30, pp.1722-1737, 2019.

[28] Yinghao Yu , Wei Wang , Renfei Huang , Jun Zhang , and Khaled Ben Letaief, "Achieving Load-Balanced, Redundancy-Free Cluster Caching with Selective Partition", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 31, pp. 439-454, 2020.

[29] Juan Luis Jimenez Laredo, Frederic Guinand, Damien Olivier, and Pascal Bouvry, "Load Balancing at the Edge of Chaos: How Self-Organized Criticality Can Lead to Energy-Efficient Computing", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 28, pp. 517-529, 2017.

[30] Alberto Cabrera, Alejandro Acosta, Francisco Almeida, and Vicente Blanco, "A Dynamic Multi–Objective Approach for Dynamic Load Balancing in Heterogeneous Systems", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 10, October 2020.

[31] YuAng Chen and Yeh-Ching Chung, "Workload Balancing via Graph Reordering on Multicore Systems", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 33, pp. 1231-1245, 2022.

[32] Mahdi Jafari Siavoshani , Farzad Parvaresh , Ali Pourmiri , and Seyed Pooya Shariatpanahi, "Coded Load Balancing in Cache Networks", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 31, pp. 347-358, 2020.

[33] Jonatha Anselmi and Josu Doncel, "Asymptotically Optimal Size-Interval Task Assignments", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 30, pp. 2422-2433, 2019.

[34] Qiong Chen, Zimu Zheng, Chuang Hu, Dan Wang, and Fangming Liu, "On-Edge Multi-Task Transfer Learning: Model and Practice With Data-Driven Task Allocation", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 31, pp. 1357-1371, 2020.

[35] Ashraf Suyyagh and Zeljko Zilic, "Energy and Task-Aware Partitioning on Single-ISA Clustered Heterogeneous Processors", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 31, pp. 306-317, 2020.

[36] Pingpeng Yuan, Changfeng Xie, Ling Liu, and Hai Jin, "PathGraph: A Path Centric Graph Processing System", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 27, pp. 2998-3012, 2016.

[37] Lazaros Papadopoulos, Dimitrios Soudris, Christoph Kessler, August Ernstsson, Johan Ahlqvist, Nikos Vasilas, Athanasios I. Papadopoulos, Panos Seferlis, Charles Prouveur, Matthieu Haefele, Samuel Thibault, Athanasios Salamanis, Theodoros Ioakimidis, and Dionysios Kehagias, "EXA2PRO: A Framework for High Development Productivity on Heterogeneous Computing Systems", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 33, pp. 792-804, 2022.

[38] Weng Chon Ao and Konstantinos Psounis, "Resource-Constrained Replication Strategies for Hierarchical and Heterogeneous Tasks", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, pp. 793-804, 2020.

[39] Umar Ibrahim Minhas, Roger Woods, Dimitrios S. Nikolopoulos, and Georgios Karakonstantis, "Efficient Dynamic Multi-Task Execution on FPGA-Based Computing Systems", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 33, pp. 710-722, 2022.