AI-BASED OPTIMIZATION OF AUTO SCALING TECHNIQUES FOR LOAD BALANCING IN CLOUD COMPUTING ENVIRONMENTS

A Thesis submitted

In partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE AND ENGINEERING

By

Mr. M. ARVINDHAN

Registration Number - 18SCSE3010024

Supervisor

Dr. D. RAJESH KUMAR

Professor

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY

UTTAR PRADESH, INDIA

AUGUST 2023

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis, entitled "**AI-Based Optimization Of Auto Scaling Techniques For Load Balancing In Cloud Computing Environments**" in fulfillment of the requirements for the award of the degree of Doctor of Philosophy in the department of computer science and engineering and submitted in Galgotias University, Greater Noida is an authentic record of my own work carried out during a period from July 2018 to July 2023 under the supervision of **Dr. D. RAJESH KUMAR**

The matter embodied in this thesis has not been submitted by me for the award of any other degree of this or any other University/Institute.

Mr. M. ARVINDHAN

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

[illegible]

Dr. D. RAJESH KUMAR
Supervisor
School of Computing Science and Engineering

The Ph.D. Viva-Voice examination of ____

Research Scholar has been held on ______________________.

Sign of Supervisor   Sign of External Examiner

ABSTRACT

An efficient way for task scheduling in cloud computing environment to allocates relevant Virtual Machines in Server, based on the demand is more important .Due to the complicated structure of jobs and resources in cloud data centers, this task scheduling problem is known to be NP-complete, making it a difficult issue to solve. Schedule length (or "make span") minimization is the primary focus of task scheduling. They can make better use of cloud resources and shorten the duration of jobs' execution, response, and waiting times if our lower the span of execution. Load balancing, in which incoming work is dispersed among available resources, comprises one of the primary uses of task scheduling.

In order to solve the Unrelated Parallel Machine Scheduling Problem with sequence-dependent setup times, the Courtship Learning-Improved Firefly Algorithm has been presented. In order to enhance cooperation among the Fireflies and prevent them from settling for suboptimal solutions, this approach makes use of a Cauchy's value density value. The primary goal of Firefly is to dynamically distribute work among available machines in order to achieve maximum efficiency. The basic goal of Firefly is to dynamically balance workload across computers to enhance performance. A sequential UPMSP numerical solution is provided by objective fitness value using an upgraded Firefly algorithm with engagement learning.

Dynamic Q-Learning aims to address the complexity and overhead of handling structured and unstructured data formats, the demand for power and energy efficiency in modern processor design, and the need for resource utilization and allocation in processing complex tasks using neural networks. Additionally, the focuses on task scheduling algorithms in the cloud to improve processor productivity and utilization while considering system bandwidth. Reinforcement learning can be applied in cloud load balancing situations to optimize the distribution of network traffic across servers in a cloud environment. By using reinforcement learning, load balancing algorithms can continuously learn and adapt to changing conditions.

The reinforcement learning agent can perceive and interpret the current state of the system, take actions such as redirecting traffic to different servers, and receive feedback in the form of rewards or penalties based on the performance of those actions.

Through trial and error, the reinforcement learning agent can learn the optimal load balancing strategy for a given environment. It can learn to adjust traffic distribution based on variables such as server capacity, network latency, and the current workload. This adaptive learning capability can lead to more efficient and effective load balancing, improving performance and resource utilization. Overall, applying reinforcement learning in cloud load balancing enables the development of intelligent and adaptive load balancing algorithms that can optimize the distribution of traffic and improve the performance of cloud-based applications.

Keyword: *Dynamic Q-learning, Virtual Machine Load balancing with Auto scaling, Deep reinforcement learning, Actor-Critic based Workload Algorithm.*

ACKNOWLEDGEMENT

I Mr. M. Arvindhan Working as an Assistant Professor and doing research for the degree of Ph.D. at Galgotias University was a quite magnificent and challenging experience for me. In all these years, many people directly or indirectly contributed to shaping my career. It was hardly possible for me to complete my doctoral work without the precious and invaluable support of these personalities.

I would like to give my small tribute to all those people. Initially, I would express my sincere gratitude to my supervisor Dr. D. RAJESH KUMAR Professor, School of Computing Science and Engineering for his valuable guidance, enthusiasm, and over-friendly nature that helped me a lot to complete my research work promptly.

I must owe a special debt of gratitude to Hon’ble Chancellor Mr. Suneel Galgtotia Mr. Dhruv Galgotia, CEO, and Hon’ble Vice-Chancellor Dr. K. Mallikharjuna Babu, Galgotias University for their valuable support throughout my research work.

I express my sincere thanks to Dr. Munish Sabrawal, Dean School of Computing Science & Engineering, and Dr. R. Rajesh Kannan, Head, Ph.D. Programme for their guidance and moral support during my research work and to all faculties of the School of Computing Science & Engineering who helped me a lot in my course of research work and all those who stood behind me.

I express my heartfelt gratitude to my parents for their moral support and encouragement. I am thankful to my family members, friends, and well-wishers who have been constantly motivating me throughout the progress of this work.

M. ARVINDHAN

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

IST OF ABBREVIATIONS

ABBREVIATION		FULL FORM
RL-LB	-	Reinforcement Learning Load Balancing
RLLBP	-	Reinforcement Learning-Based Load Balancing Policy
RQLB	-	Reinforcement Q-learning Load Balancing
RDLB	-	Reinforcement Deep Learning Load Balancing
DRLB	-	Distributed Reinforcement Load Balancing
DQLB	-	Dynamic Q-Learning Load Balancing
DQBLB	-	Dynamic Q-Learning Cloud Balancing
DQRLBPaaS	-	Dynamic Q-RLoad Balancing as a Service
DLBL	-	Dynamic Load Balancing
SLB	-	Static Load Balancing
ALB	-	Adaptive Load Balancing
ELB	-	Elastic Load Balancing
WLB	-	Weighted Load Balancing
RLB	-	Round Robin Load Balancing
HLBR	-	Hierarchical Load Balancing Router
FBLB	-	Feedback-based Load Balancing
VLB	-	Virtual Load Balancing
QoS-LB	-	Quality of Service Load Balancing
ASG	-	Auto Scaling Group
ACAP	-	Autoscaling Capacity Adjustment Policy
AAM	-	Autoscaling Application Manager
ARA	-	Autoscaling Resource Allocation
APM	-	Autoscaling Performance Management
BaaS	-	Backup as a Service
DRaaS	-	Disaster Recovery as a Service
IoTaaS	-	Internet of Things as a Service
SECaaS	-	Security as a Service
AIaaS	-	Artificial Intelligence as a Service

LIST OF PUBLICATIONS

INTERNATIONAL JOURNAL

1. **Arvindhan, M., & Kumar, D. R.** (2023). Adaptive Resource Allocation in Cloud Data Centers using Actor-Critical Deep Reinforcement Learning for Optimized Load Balancing. International Journal on Recent and Innovation Trends in Computing and Communication, 11(5s), 310–318. https://doi.org/10.17762/ijritcc.v11i5s.6671 **(SCOPUS).**
2. **Arvindhan, M., & Dhanaraj, R. K**. (2023). Dynamic Q-Learning based Optimized Load Balancing Technique in Cloud -Journal Mobile Information System **(SCI Revision submitted).**
3. **Arvindhan, M., & Dhanaraj, R. K.** (2022). The firefly technique with courtship training optimized for load balancing independent parallel computer task scheduling in cloud computing. International Journal of Health Sciences, 6(S1), 8740²8751. https://doi.org/10.53730/ijhs.v6nS1.6999 **(SCOPUS)**

INTERNATIONAL CONFERENCES

4. **Arvindhan, M** et.al., scheming a Proficient Auto Scaling Technique for Minimizing Response Time in Load Balancing on Amazon AWS Cloud (March 15, 2019). International Conference on Advances in Engineering Science Management & Technology (ICAESMT) - 2019, Uttaranchal University, Dehradun, India, Available http://dx.doi.org/10.2139/ssrn.3390801 (**SCOPUS**).
5. **Arvindhan, M., Rajesh Kumar, D.** (2022). Analysis of Load Balancing Detection Methods Using Hidden Markov Model for Secured Cloud Computing Environment. In: Deepak, B.B.V.L., Parhi, D., Biswal, B., Jena, P.C. (eds) Applications of Computational Methods in Manufacturing and Product Design. Lecture Notes in Mechanical Engineering. Springer, Singapore. https://doi.org/10.1007/978-981-19-0296-3_53 (**SCOPUS**).

CHAPTER - 1

INTRODUCTION

The recent development of cloud computing has drastically improved our ability to store, process, and gain access to data. Managing and distributing the enormous amount of data produced by Internet of Things devices across cloud data centers is becoming increasingly important. To this end, load balancing algorithms in cloud data centers are vital for distributing both physical and virtual servers evenly across the network. These load-balancing algorithms take into account resources like processing power, memory, and network throughput to ensure that no single server is overwhelmed.

1.1 LOAD BALANCING BACKGROUND MODEL

There are two different kinds of load balancing: Dynamic Load Balancing (DLB) and Static Load Balancing (SLB). SLB performs load distribution using a system model that has been predefined. The equipment metrics and indeed the software metrics together define the aspects of the system. Based on this model, a central load balancer makes an effort to achieve a workload distribution that is as even as possible across all of the resources to which it distributes work. The density function is limited to relying only on the workloads that have already been distributed and also the list of workflows that still need to be distributed because of system is believed to be static. DLB, in contrast to SLB, use of a static workload but rather relies on the current state of the system. As a result, the distribution function has a propensity to be more complicated because it adds information regarding the system province and the workload to the input[1]. The level of complexity is increased even further when considering cloud systems that have a dynamic topology. In order to get the most out of the system, the load balancers need to respond appropriately to the changes. The openness of the SLB method, which ultimately results in improved performance, is one of its primary selling points. It is possible to make effective use of this if the characteristics of the workloads are well understood and, as a result, accurate predictions regarding the execution of the workloads can be made. Consequently, a static load balancer may be more economically efficient and simpler to incorporate within already existing systems[2].

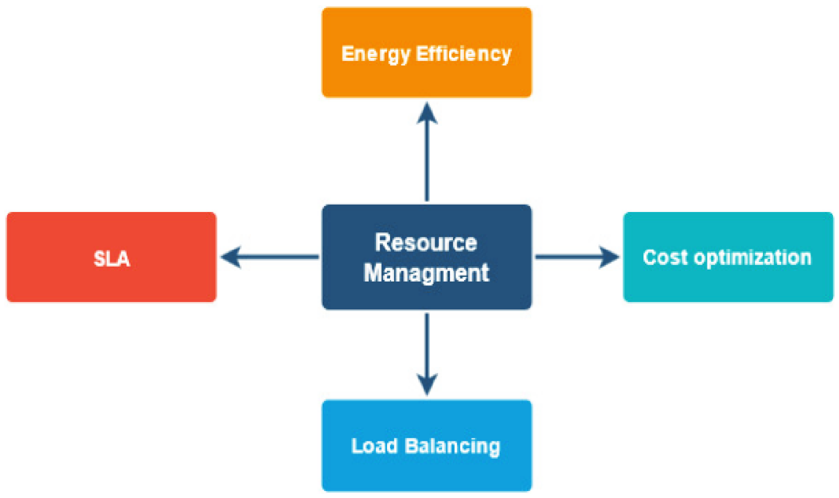


Figure 1.1 Resource management dimensions in cloud computing

In clouds, load balancing can occur between either physical hosts or virtual machines. All of the hosts or virtual machines (VMs) get an equal share of the dynamic workload to this balancing technique. Load balancing as a service (LBaaS) is another name for cloud-based load balancing. Optimizing resource utilization and satisfying the performance needs of different applications are both greatly aided by effective scheduling in cloud computing. Load balancing, job scheduling, resource allocation, workflow layout, energy conservation, and environmental preservation are all aspects of resource scheduling in cloud computing[3]. One of the most important technologies connected to cloud computing efficiency is resource scheduling. Cloud service companies can increase performance and save money for their customers by concentrating on these areas. As shown in the Figure 1.1 in cloud computing, services can only be made accessible and efficiently with the use of effective workflow scheduling.

There are numerous strategies for balancing load.

- Static way approach in Load balancing: Once the load has been assigned to a node, it cannot be moved to another node in a static topology. It disregards the current condition of the system in favor of data about its typical operation. Adapts to the current situation of each node and distributes the workload accordingly.

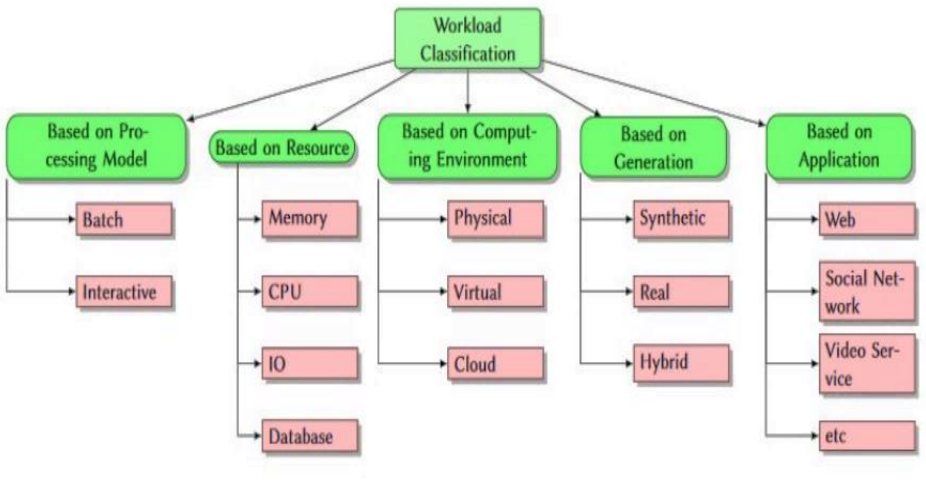


Figure 1.2 Workload classification based on different aspects

a) **Memory Workload:** Every application or directive requires memory to store temporary or permanent data and perform intermediate computations in accordance with the application's specifications. Memory utilization establishes the system's memory usage over a specified period of time or at a particular instance. The paging method and segmentation operations utilize a great deal of virtual memory, leading to an increase in the use of physical memory. Nevertheless, if the number of programs that are running increases to the point where memory turns into an execution bottleneck, this indicates a need for additional memory or programs that need to be managed more frequently[4].

b) **CPU workload:** CPU refers to the sum total of the instructions being executed by the central processing unit at any given time or point of time. If the CPU is consistently overworked, it signals the need for additional processing capacity, whereas low CPU utilization signals the opposite. By decreasing the number of cycles needed to execute a function properly, performance can be further improved for the same number of instructions running on a CPU at a given instance[5].

c) **Database Workload:** Databases are able to be evaluated for memory utilization, throughput at peak loads, and I/O throughput when interacting with applications. With the help of each of these factors, one can roughly calculate the capabilities and settings of the database. As shown in the above figure 1.2. No but by figuring out the database's overall workload can be examined by counting the total number of inquiries it processes in a specific time period[6].

d) **I/O Workload:** Most programs invest a considerable amount of time in receiving input and generating output. consequently, it is crucial to analyze the load of the input-output (I/O) combinations on a system to guarantee that the necessary load performance requirements are satisfied. One definition of input-output workload is the sum of the inputs received and the results that are generated by a system over a given time period effectively[7].

- Dynamic way of approach in load balancing: On the live status of request based on the user on demand basis these methods work.
- There are two distinct varieties of dynamic strategy:
- Centralized way Approach: Load management and redistribution in a centralized system are the responsibility of a single node.
- Dynamic way approach: Subsequently distributes load around nodes based on its present status.

Examples of qualitative measurements that can improve cloud load balancer performance include:

- Throughput refers to tasks accomplished within a specified timeframe. Tasks with high throughput outperform the other system.
- Migration time refers to the time across jobs moving between machines in a system. Bandwidth should be reduced for increased efficiency.
- **Response time:** Time taken for the system to complete the task in a given system. This method will reduce the performance by decreasing the setting of the particular system
- **Scalability:** There is a low cap on the number of computers and CPUs it can handle. If our instances want enhanced system effectiveness, it can boost this **variable's values.**
- **Resource Utilization:** Provides information about current resource usage. Resource use should be optimized for optimum load balancing.

1.2 ANALYSIS OF VARIOUS ALGORITHMS FOR LOAD BALANCING ALGORITHMS

Cloud computing distributes processors, storage, and networks. subsequently encompasses interoperability, scalability, and virtualization. The fundamental objective of cloud computing is to maximize resource use and system performance. Heuristic, metaheuristic, and composite load-balancing algorithms exist in the cloud load balancing algorithm.

1.2.1 Heuristic Approach-Based Load Balancing Algorithms

Heuristics are static or dynamic. Heuristics are constraints that aim to solve a problem optimally. The restrictions are designed to achieve a solution within a certain period, and are generally problem-dependent. This paper's approach-based algorithms contain limits and are designed to solve problems quickly. These algorithms are beneficial because they efficiently solve problems[8]. Some of the Alternative's algorithm like metaheuristic algorithms Implementing heuristic algorithms is simple.

1.2.2 Metaheuristic Approach-Based Load Balancing Algorithms

The metaheuristic algorithm is continuous. Metaheuristic algorithms include all evolutionary and swarm intelligence methods. It takes longer to execute metaheuristic algorithms than heuristic algorithms to find the final solution. In many circumstances, metaheuristic algorithms require more time to discover the final answer due to their larger scope of solutions compared to heuristic methods. In addition, metaheuristics are unpredictable[9].

1.2.3 Improved Max-Min Algorithm

The improved Max-Min algorithm introduces enhancements to the original algorithm by incorporating additional optimization techniques, resulting in improved efficiency and performance[10].

1.2.4 **Balance Reduce Algorithm**

The balance reduction algorithm is a method used to efficiently distribute and allocate resources in order to optimize performance and minimize costs. By dynamically managing task load and considering factors such as availability, flexibility, scalability, and minimal cost, the balance reduce algorithm ensures that resources are allocated effectively[11].

1.2.5 Threshold Algorithm

The threshold algorithm is a mathematical model used to determine optimal decision-making points based on predefined thresholds or conditions. These thresholds serve as indicators that trigger specific actions or outcomes in various domains, such as finance, healthcare, and manufacturing. For example, in financial decision-making, the threshold algorithm can be employed to determine the optimal time to buy or sell stocks based on specific price thresholds[12].

1.2.6 Artificial Bee Colony Algorithm

This algorithm belongs to the family of meta-heuristic algorithms and has gained significant attention due to its effectiveness in solving complex optimization problems. By imitating the collective intelligence and collaboration observed in real bee colonies, the Artificial Bee Colony Algorithm offers a unique approach to solving various optimization tasks. With its ability to efficiently explore the solution space and find optimal solutions, this algorithm has found applications in diverse fields such as engineering, economics, and data mining[13].

1.2.7 Honey bee Algorithm

The honey bee algorithm is a nature-inspired optimization method that mimics the behavior of honey bees in search of food. This algorithm was developed based on the understanding of honey bees' foraging behavior, which is a complex process influenced by factors such as genetics, physiology of the bees, ecological environment, and social conditions of the hive. The honey bee algorithm is a population-based search algorithm that combines neighborhood search with random search, mimicking the foraging behavior of honey bees[14].

1.2.8 Min-min Scheduling Algorithm

The Min-Min scheduling algorithm has been widely recognized as one of the best methods for task scheduling in cloud and grid computing. Researchers have extensively contributed to the development and improvement of the Min-Min algorithm, making it highly efficient in allocating tasks to appropriate resources Furthermore, the efficiency of Min-Min scheduling has been instrumental in minimizing the time required for scheduling execution[15].

1.2.9 Particle Swarm Optimization Algorithm

The Particle Swarm Optimization algorithm is a widely used meta-heuristic technique for optimizing various fields, including software. This algorithm is inspired by the behavior of a swarm, such as a flock of birds or a school of fish, where individuals in the swarm cooperate and communicate with each other to find an optimal solution[16].

1.2.10 Least Connection Algorithm

The least connection algorithm directs new requests to servers with fewer active connections at any given time. This approach enables effective utilization of resources as it distributes the workload based on current connection counts rather than capacity alone. It helps prevent overloading heavily loaded servers while ensuring efficient use of available resources and maintaining low response times[17].

1.2.11 GA Algorithm

It is widely used in various fields such as engineering, finance, and computer science to solve complex optimization problems that are difficult or impossible to solve using traditional methods. The Genetic Algorithm operates by creating a population of potential solutions and continually improving them through iterations of selection, and recombination.

Dynamic algorithms can be classified as distributed or non-distributed. The distributed technique involves all nodes executing the dynamic load-balancing algorithm and sharing the task[18]. The system nodes interact cooperatively and non-cooperatively. The cooperative form involves nodes working together to reduce task response times. Each node works autonomously to achieve a local goal, such as reducing task response time in the non-cooperative variant. Centralized and semi-distributed algorithms are non-distributed. In centralized load balancing, a single node, the central node, runs the algorithms and is totally responsible.

The SLB has the drawback that the balancer is unaware of the state of the system, which is a disadvantage. Even though SLB only makes assumptions about the state of a system, there really is no assurance that the assumed state and the actual state will not deviate dramatically. This is especially important to keep in mind when dealing with highly dynamic analytics and cloud architectures, both of which typically support a large number of distinct workloads on a single shared platform[19]. On the other hand, DLB needs to have a more robust integration with the system in order to respond to the current state of the system. This causes more work to be done during the initialization and setup of systems that use dynamic load balancers, and dynamic balancers expensive in terms of the amount of computation and storage that is required for metrics and system state.

The fact that Load balancing strategy can be applied to a diverse range of situations is one of its many advantages. A workload does not need to be characterized with the same level of precision as it would be for the static approach because the distribution of workloads is linked to the state of the system. The effect that each workload has on the system is measured and taken into account when making the subsequent scheduling choice. When a workload that has unusual demands is scheduled, it creates an imbalance between the utilization of the available resources. A load capacity balancer does not recognize this and keeps going scheduling to any and all allotted instance of the cloud computing, whereas a dynamic balancer recognizes the asymmetry and stops scheduling to resources. After that, various respond by shifting further workloads to a variety of resources in order to achieve more consistent utilization[20].

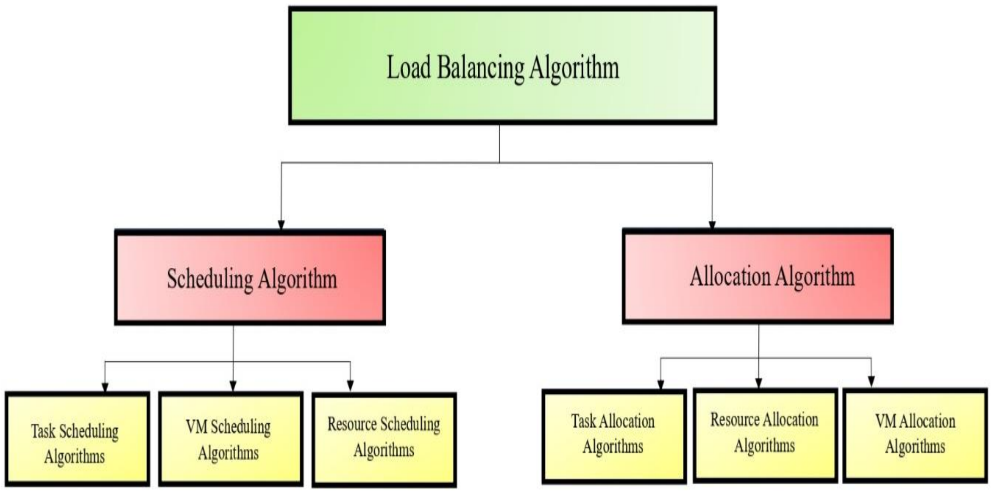
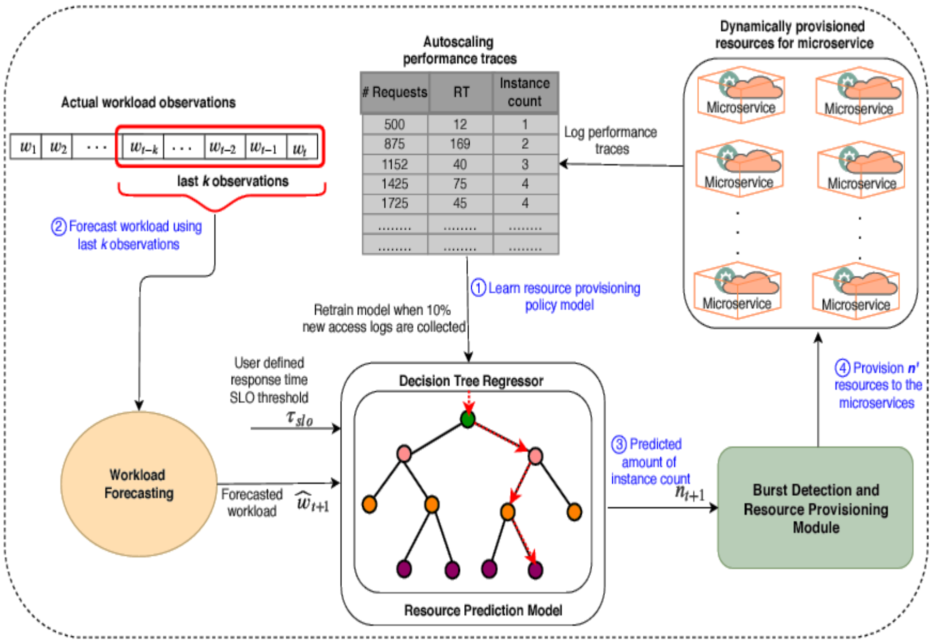

Load Balancing Algorithm
Scheduling Algorithm
Allocation Algorithm
Task Scheduling Algorithms
VM Scheduling Algorithms
Resource Scheduling Algorithms
Task Allocation Algorithms
Resource Allocation Algorithms
VM Allocation Algorithms

Figure 1.3 Load Balancing Systems

In distributed architectures including such as Web services, massive data stores, cloud computing, and other similar types of systems, load balancing is a frequently used method for assigning tasks. Within those kinds of designs, there is a dispatcher that works toward achieving a balance in the distribution of work among the various servers which thus make up the system. This helps to reduce the amount of time spent waiting in line. In this regard, the fundamental problems are including not only the performance assessment but also the introduction of innovative load-balancing algorithms. As shown in the above figure 1.3 in this proposed work consideration a traditional load balancing system, which is made up of a centralized scheduler and N data centers, all of which are connected [21].There is a single stream of exogenous tasks arriving at a rate of, and as soon as it is received, its required to be immediately distributed to one of the queues. Because it decides directly that also queue the newly arriving tasks should join, the load balancing strategy that such a system employ is an important factor in determining how well it performs. According with message flows, load balancing regulations can be divided into two distinct groups: those that are push-based and those that are pull-based. In a strategy that is based on push, the scheduler actively needs to send query items to the servers and then waits for its own responses; in such a policy that is premised on squeeze, the scheduler simply listens towards the report that is sent from the servers. The job routing and scheduling decision is made at the dispatcher, and it is based just on squeeze that are received from the servers and that are saved inside the local memory[22].



Actual workload observations
w_1 w_2 $\cdots$ w_{t-k} $\cdots$ w_{t-2} w_{t-1} w_t
last k observations
② Forecast workload using last k observations
Autoscaling performance traces
Log performance traces
Dynamically provisioned resources for microservice
Microservice
Microservice
Microservice
Microservice
Microservice
Microservice
① Learn resource provisioning policy model
Retrain model when 10% new access logs are collected
User defined response time SLO threshold
τ_{slo}
Decision Tree Regressor
Resource Prediction Model
Workload Forecasting
Forecasted workload
$\widehat{w}_{t+1}$
③ Predicted amount of instance count
n_{t+1}
④ Provision n' resources to the microservices
Burst Detection and Resource Provisioning Module

# Requests	RT	Instance count
500	12	1
875	169	2
1152	40	3
1425	75	4
1725	45	4
....	....	....
....	....	....

Figure 1.4 System model of general load balancing.

1.3 TYPES OF LOAD BALANCER

a) **Network load balancer**: also known as a Layer 4 Balancer, is an OSI Model component that operates on the fourth layer of the network and decides which server or servers are best suited for processing TCP/UDP connection requests from website visitors. As one type of load balancer between many others.

b) **Internal load balancers:** are similar to network load balancers in that they function on layer 4 of the OSI model. When it comes to managing, stabilizing, and balancing the physical servers and virtual machines, including the network area storage, internal load balancing occurs most frequently in onsite infrastructure[23].

c) **Hardware load balancer:** is an actual piece of hardware that has its own operating system pre-installed. Its job is to divide up the traffic from the world-wide web's users and send it to various server farms (a group of web servers connected to a network that host online applications). To function, a hardware load balancer needs at least two virtual machines. The system administrator sets it up with their own set of policies for optimal performance and to prevent the virtual machines from being overworked[24].

d) **Virtual load balancer:** is similar to a software load balancer in that it distributes network traffic across multiple servers, As shown in the above figure 1.4 but it is not the same thing. The web track is dispersed by the virtual load balancer, which uses the hardware load balancer's software to run on virtual computers[25].

1.4 AUTO SCALING TECHNIQUES THROUGH HYPERVISOR IN CLOUD ENVIRONMENT

On cloud computing, an autoscaling architecture is used to autonomously alter the amount of hardware and software dedicated to an application or service in response to changes in demand. The aim is maximum efficiency with minimum expenditure. In most autoscaling frameworks, the following elements make up the performance model:

Gathering Data and Metrics Monitoring and collecting data and metrics about an application or service's effectiveness is the initial stage in any autoscaling system. CPU load, memory use, network traffic, application volume, and server response time are all examples of such measures. The gathered data is examined to reveal patterns in present workload and trends in resource consumption[26]. This analysis is useful for figuring out what steps to follow in terms of scaling. Policies for determining when and how much to scale up or down are defined within the autoscaling architecture. Workload assessment is used to set thresholds and regulations for these regulations. For instance, the system may be instructed to launch additional processes if CPU use is at or above 60% for an extended period of time.

There are two types of auto-scaling. 1. System level (Physical Machine) 2. Application level (Virtual Machine)

When the auto scaling framework detects that a scaling policy has been triggered, it will begin the appropriate scaling steps. Leasing new cloud computing resources like virtual machines, containers, or whatever else is required to satisfy the increased demand is one example of what might be done. When the amount of labor drops, it's possible to free up some of the available resources. Predicting the impact that scaling will have on the efficiency of an application is essential for achieving seamless autoscaling. This helps ensure that neither too many nor inadequate resources are on hand.

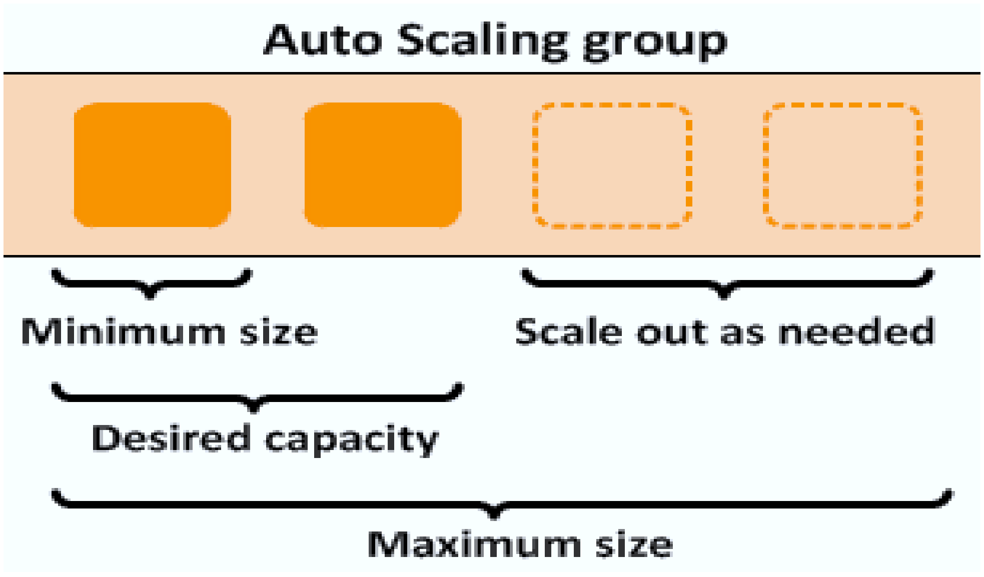


Figure 1.5 Process of Auto scaling groups in Virtual Machine

A warmup interval is typically enforced by the auto scaling framework following a scaling action has been taken. In order to let the system balance while avoiding rapid oscillations in no additional scaling operations are made during this time. Some auto scaling frameworks use machine learning and artificial intelligence methods to foresee similarities in workload evolution and refine scaling priorities. As shown in the above figure 1.5.This has the potential to result in more timely and precise scaling measures[27]. The performance model optimizes outcomes while simultaneously considering the cost of inputs. It aims for cost-effective auto scaling by balancing performance needs with budget constraints. Upper and lower constraints for distributing resources are defined by the framework to prevent excessive scalability beyond the cloud's capabilities. Effective scaling automatically frameworks collect input from users on the scaled application's effectiveness and use that information to fine-tune its scaling policies and forecasts in real time.

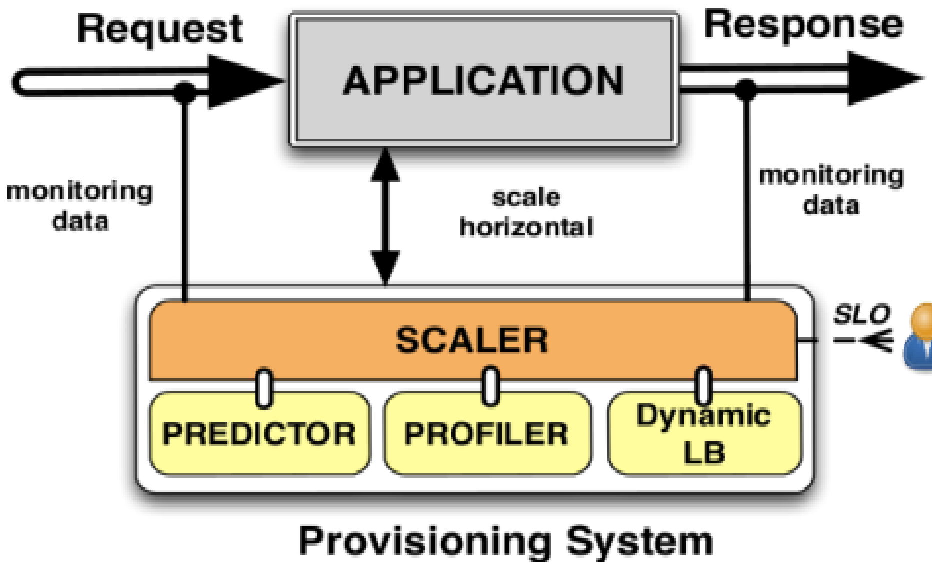


Figure 1.6 Autoscaling in Virtual machine environment

1.5 AUTO SCALING FEATURES ARE PROVIDED BY CLOUD SUPPLIERS

Autoscaling providers are increasing day to day process. A number of manufacturers offer autoscaling services and technology. Each of the major public cloud providers offers its own services that enable autoscaling. Third-party services, which are frequently grouped together as "cloud management platforms," enable enterprises to improve cloud installations by including autoscaling policies that connect to a cloud provider platform.

Autoscaling features are provided by the following cloud vendors:

AWS stands for Amazon Web Services. AWS offers a variety of autoscaling services, such as AWS Auto Scaling and Amazon EC2 Auto Scaling. AWS Auto Scaling is a solution designed for users who require the ability to scale resources across various AWS services. As shown in the above Figure 1.6 the Amazon EC2 Auto Scaling service, on the other hand, is dedicated to providing autoscaling capabilities for Amazon EC2 instances that provide virtual compute resources[28].

GCE stands for Google Compute Engine. GCE offers autoscaling capabilities to cloud users who run managed instance groups of virtual machines (VM) instances.

Managed instance groups are an optional Google Compute Engine deployment strategy in which groups of similar virtual machines are distributed throughout GCE in a managed manner to offer improved availability.

The IBM Cloud. IBM has a cluster-autos caler module that can be implemented on IBM Cloud workloads. Based on the scaling requirements set by scheduled workload regulations, this auto scaler can increase or reduce the number of nodes in a cluster.

Azure from Microsoft. The Azure Auto Scale feature allows customers of the Microsoft Azure cloud platform to automatically scale resources. Azure Auto Scale can be used with virtual machines, mobile apps, and websites[29].

As cloud computing becomes the center of attention, businesses of all stripes are scrambling to adopt it so they can keep their digital operations running smoothly and produce the best possible outcomes. The adoption of microservices was a tremendous improvement for the company as a whole, since it allowed the many parties involved to concentrate on creating high-quality apps. Microservices design replaces the old monolithic architecture, which is made up of a large application, with a single application made up of smaller services that can talk to each other. It improves teams' coding quality, allows for more versatile use of other components, and reduces application costs[30].

a) **Performance Optimization:** Load balancing ensures that incoming requests are distributed evenly across multiple instances of the application or service. Autoscaling complements this by dynamically adjusting the number of instances based on the workload. This combination ensures that the application can maintain optimal performance even during peak traffic periods.
b) **High Availability and Fault Tolerance:** With load balancing and autoscaling, if one instance of the application or service fails, the traffic is automatically redirected to healthy instances. This enhances the application's availability and fault tolerance, as it can quickly recover from failures and continue serving users without interruptions.
c) **Scalability:** Autoscaling allows the application to scale both horizontally (adding more instances) and vertically (increasing resources of existing instances) based on demand. This flexibility enables the application to handle

sudden spikes in traffic or accommodate increased user demand without manual intervention[31].

d) **Resource Optimization:** Autoscaling ensures that resources are allocated efficiently to match the current workload. During periods of low demand, instances can be automatically scaled down, reducing resource wastage and associated costs.

e) **Time and Resource Savings:** Autoscaling eliminates the need for manual intervention in provisioning or deprovisioning resources based on fluctuating workloads. This saves time for the IT team and reduces the risk of human errors in resource management.

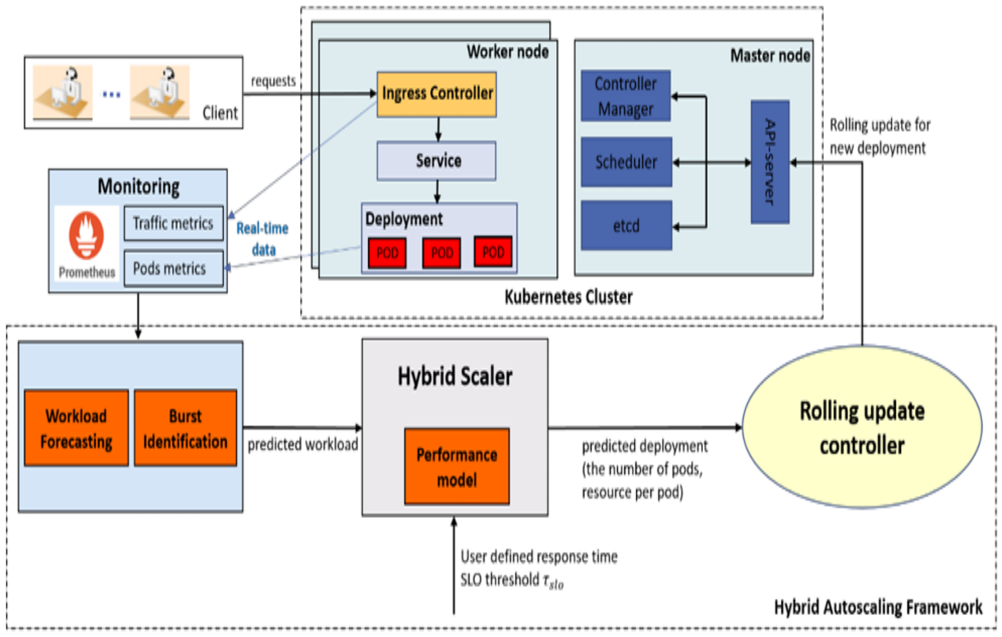


Figure 1.7 Performance model of Hybrid Autoscaling framework

Applications load is taken into account when making autoscaling decisions. As shown in the above Figure 1.7 for instance, the autoscaling approach swiftly executes multiple resource allocation configurations for sustaining response time requirements in response to unexpected and dramatic changes in a workload, often known as burstiness[32]. Conventional autoscaling approaches do not detect and control burstiness, leading to decreases in the performance of applications.

1.6 USING AUTO-SCALING TO IMPROVE THE EFFICIENCY OF EVENT-DRIVEN TOPOLOGY

Each service in a micro services architecture experiences some amount of demand. An instantaneous fashion load distribution strategy must be implemented to allocate and utilize the resources best, as the disparity in load necessitates a distinct number of instances to be created for each service. In essence, this is what auto-scaling is all about. It offers a flexible method of dividing system resources according to the needs of individual microservices processes. There are, however, a variety of automatic scaling methods. As shown in the below figure1.8 Several elements, like traffic factor, backpressure, etc., must be taken into account before settling on the optimal scaling approach. Global Logic is an industry leader in cloud computing and has developed various cloud-based microservices architectures for prominent companies around the world. We investigate cutting-edge developments in cloud computing and microservices with the help of a specialized team of experts in the industry.

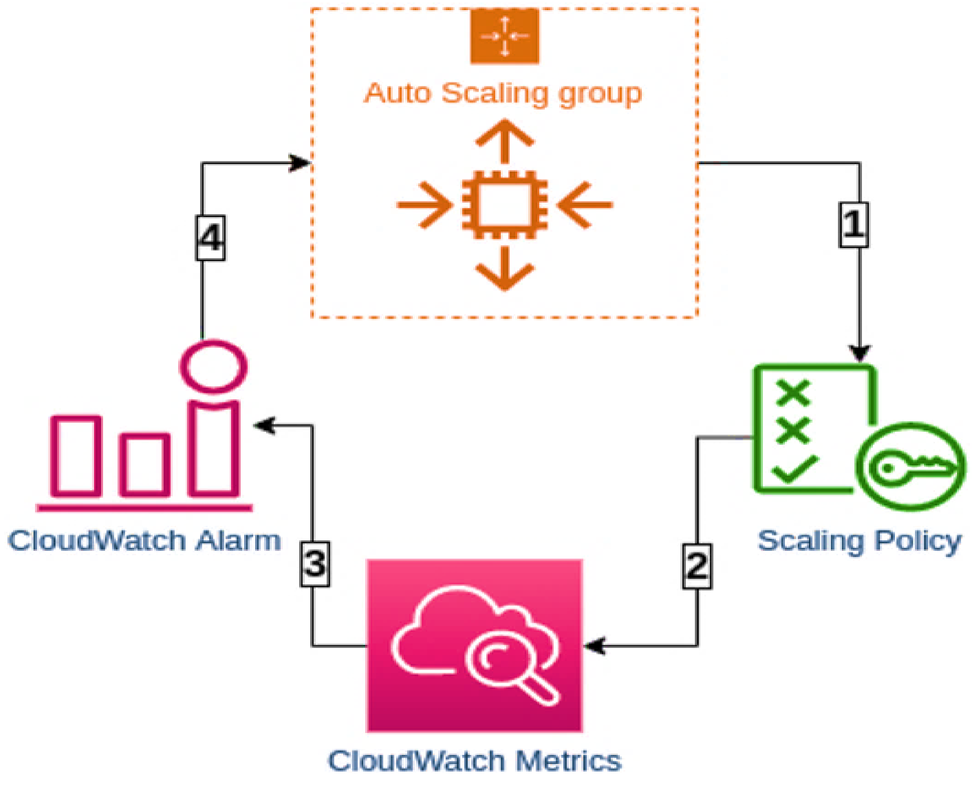


Figure 1.8 Scaling Policy for Autoscaling in Cloud Environment

1.6.1 Throttling

One successful method for lowering the system's bandwidth is to slow the rate at which producers send requests. In most cases, this is done so that consumers with outstanding requests have some breathing room before the system is reset to its original state. Once this is completed, regular message output rates can be resumed by the producers. However, clusters on conventional distributed queuing systems like Kafka cannot restrict the system's total number of users. Producers written in Scala and Java can be tweaked to improve the system's speed, and the system is built such that consumers can consume at high rates[33]. Large amounts of data are pushed rapidly, such as the amount of data a server should return for a message of a certain size, the amount of data to be delivered each partition, etc. Allocating new partitions, reducing the overall size of the message, etc., are all examples of management techniques that can be used effectively on such systems. Kafka provides two distinct types of consumers, each with their own set of configuration settings. One of the most efficient ways to limit a system's bandwidth is to slow the rate at which requests come in from producers. In most cases, this is done so that consumers with outstanding requests have some breathing room before the system is reset to its original state. Once this is completed, regular message output rates can be resumed by the producers. However, clusters on conventional distributed queuing systems like Kafka cannot restrict the system's total number of users. It's made so that users can take in information quickly, and creators can push out plenty of data rapidly. Allocating new partitions, compressing the size of the message, etc., are all examples of management actions that can improve the performance of such systems. Kafka's two types of consumers and their respective configuration settings, One of the most efficient ways to limit a system's bandwidth is to slow the rate at which requests come in from producers. In most cases, this is done so that consumers with outstanding requests have some breathing room before the system is reset to its original state. Once this is completed, regular message output rates can be resumed by the producers. However, clusters on conventional distributed queuing systems like Kafka cannot restrict the system's total number of users[34]. It's made so that users can take in information quickly, and creators can push out plenty of data rapidly. Allocating new partitions, compressing the size of the message, etc., are all examples of management actions that can improve the performance of such systems. Kafka's two types of consumers and their respective configuration setting.

1.6.2 Horizontal Pod Auto scaler (HPA)

HPA adjusts the number of Pods in operation based on the needs for available resources. It guarantees reliable performance under all conditions, yielding high-quality output at a reasonable price. HPA only adds new Pods when the memory limit is reached, when an abnormally high number of requests per second from clients is observed, or while processing requests from outside the system. There is a unique HPA object for each workflow that keeps tabs on the metrics' threshold for early modification. It's simple to include new customers within existing consumer groups, but impossible to do so across partitions. The system's throughput is enhanced and the workload is lightened as a result of the increasing number of users[35]. Slowly but surely, equilibrium is restored across all EC2 (Elastic Compute Cloud) nodes. The following is the preferred approach to scaling an EKS (Elastic Kubernetes Service) cluster, which means adding more instances.

1.6.3 Vertical Pod Auto scaler (VPA)

When expanding a system vertically, rather than horizontally, the goal is to boost the system's power output. Increasing the processing power, memory, and other forms of storage are all good examples. VPA will provide suggestions for CPU limits in accordance with the changing nature of requests that are received. As a result, this is typically implemented during batch processing of the messages. Whenever demand is high, the amount of data of the batch grows, optimizing stress on components like the central processing unit. When this happens, VPA gradually restores stability by increasing or decreasing system equipment such as CPU and memory. Microservices architectures may not benefit most from vertical scalability due to the fact that they often handle stateless operations rather than their own internal states[36].

1.6.4 Cluster Auto scaler (CA)

The total number of users can be increased using horizontal scaling, but there is a safe upper limit for each cluster that shouldn't be exceeded. When it is exceeded, no amount of work done to reduce backpressure will prevent the cluster's resources from being depleted and the system from being unstable.

1.7 CLUSTER AUTO SCALAR WITH HORIZONTAL AUTO-SCALING TECHNIQUES IN CLOUD ENVIRONMENT

When it comes to implementing microservices, Elastic Kubernetes Services (EKS) is one of the most popular Kubernetes services. Because the number of requests is never fixed, allocating the appropriate size of EKS cluster is critical. The vehicular traffic issue has an influence on the overall application, but the EKS offers a solution for it. The use of the Cluster Auto scaler is crucial in this regard. Although this is the optimal strategy, if the process continues to expand the number of Pods, it may fail due to the fact that will be no instances of EC2 ready for the latest Pod, leading the EKS ensemble to saturate.

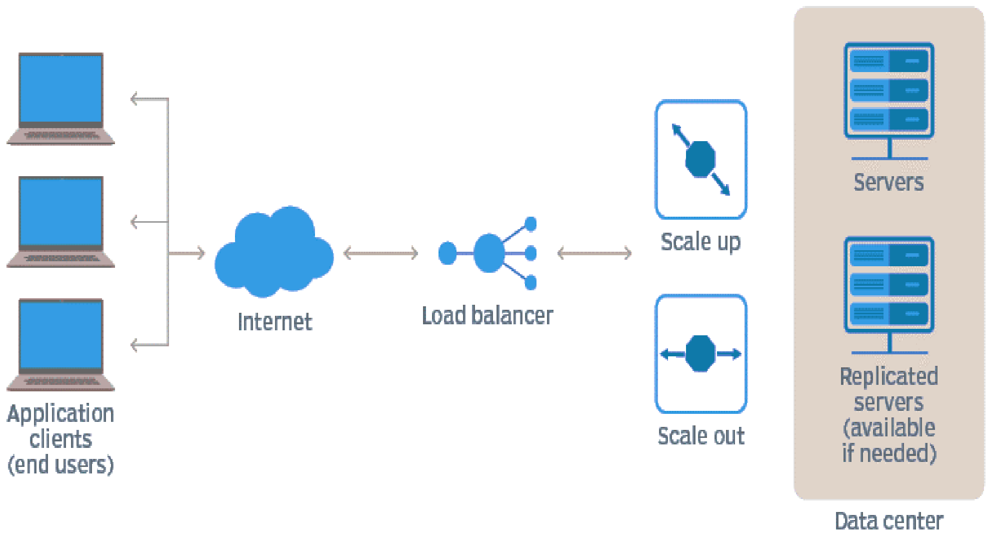

Application clients (end users)
Internet
Load balancer
Scale up
Scale out
Servers
Replicated servers (available if needed)
Data center

Figure 1.9 Autoscaling working principle

As one of the most widespread clients available, broad network access is typically achieved by using the device's constructed internet browser. High liquidity is the principle that different organizations can communicate the infrastructure underpinning this same cloud. The dynamic assignment and reassignment of physical and digital resource base in response to consumer demand[37]. Customer typically has no authority or expertise over the precise location of the underlying infrastructure, but it may be able to specify a greater depth of understanding for location (e.g., country, state, or datacenter). As shown in the above figure 1.9 Storage, computation, memory, and network and width are examples of resource base.

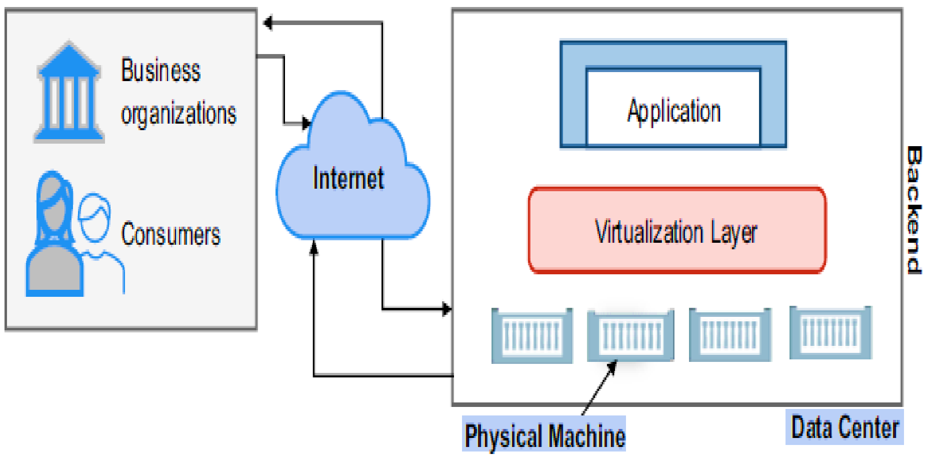


Figure 1.10 Cloud Computing Physical Application Model

Accelerated Elasticity Rapid flexibility is the capacity to just provide services that are scalable. It enables users to instantaneously seek additional cloud storage space or other facilities. Rapid elasticity is the extent to which an organization is able to respond to new in volume of work by investment scheme in an involuntary muscle manner, so that at each period in history, the available resources closely suit the current requirement As shown in the above figure 1.10. To the consumer, the achieving standards for supplying frequently appear limitless and can be appropriated at any moment in any volume[38]. Elasticity enables the cloud provider's customers to realize cost savings, which is frequently a driving factor behind the adoption of cloud computing.

Virtualization seems to be the foundation of Infrastructure as a Service. It is used in the Cloud Computing platform to distribute material assets by trying to integrate them ad hoc to meet this same varying resource requirements of Cloud service users. Consequently, virtualization enables IaaS providers are offering nearly unlimited reserves to customers and ensure the efficient and expense use of power grid. Additionally, it permits the consumer to borrow infrastructure components on an as-needed basic principle and pay just for the resources available. Amazon EC2 and Eucalyptus are examples of well-known PaaS providers.

Highly scalable is the principle that multiple organizations can communicate the basic infrastructure undergirding the cloud. The dynamic assignment and reassignment of virtual and physical resource base in response to customer needs.

Customer typically has no authority or understanding over the precise location of the underlying infrastructure, but may be able to define a higher level of abstraction for location. Storage, computation, memory, and network bandwidth are examples of resource base[39].

Rapid Elasticity Rapid elasticity is the capacity to provide assistance that are scalable. It enables users to instantaneously send additional cloud storage space or other services. Rapid elasticity is the extent to which an organization is adaptable to change throughout workload by investment scheme in an autonomic sort of way, so that at so every point in time, the resources available closely fit the current requirement. To the end user, the capabilities available for providing frequently appear limitless and can be apportioned at any time in any quantity. Elasticity enables the cloud supplier's customers to realize cost savings, which is frequently a driving factor behind the adoption of cloud services.

Measured Service: A application server that automatically controls a user's or tenant's the use memory space by leveraging a flow measurement functionality. Measured offering refers to services in which the cloud service measures, monitors, or controls the delivery of services for a variety of reasons, including billing, efficient use of resources, and better predictive making plans[40].

A cloud may be either public or private. The public cloud essentially sells Internet-based services. Amazon Web Services is currently recognized as a leading public cloud provider. The cloud infrastructure is a data center or proprietary network that provides platform as a service to a limited number of users. When network operators utilize the public cloud's resources to create a private cloud, a 'virtual private cloud' is generated. The purpose of cloud computing, whether public or private, is to provide scalable, simple access to computing resources and technology services.

To improve auto-scaling in a cloud setting, this study proposes to build a framework through the generation of dynamic rules. The suggested study endeavor takes into account the subsequent sub-objectives to accomplish this primary goal[41].

The concept of cloud scalability is based on the need to accommodate a rapidly expanding user base. With either horizontal or vertical scalability, it improves service and makes resources available in under a minute. When there is a high volume of requests (workload), the time it takes to scale can cause users to have to wait in line for the service, which in turn raises both the price of the service and the time it takes to receive it. The most effective scaling maximizes resource utilization without incurring additional costs, violates service level agreements (SLAs) only seldom, shortens wait times, and boosts quality of service (QoS).

The ability to scale up or down resource utilization based on system demand is a key differentiator between cloud computing and other computational systems that need advance reserving of resources. The workload will fluctuate, thus the resource distribution strategy needs to be flexible[42]. Smart auto-scaling systems can successfully handle these issues by estimating future resource requirements and carrying out the required scalability procedures. However, the effectiveness of such approaches depends extensively on the nature of the prediction model employed and the nature and quantity of the training data used. Responsive solutions that are too simple either over-allocate resources, driving up costs and under-utilization, or under-allocate resources, leading to a resource shortage and a drop in performance. An adaptive approach of scaling and supplying resources that can learn from its surroundings is desirable for anticipating future needs. Allocating resources efficiently ensures that the system's performance and efficiency are not compromised. The quality of service provided by large software systems is measured in several ways, including response time, throughput, and availability of services. Companies providing services run the risk of losing consumers and money if they don't provide this assurance[43].

1.8 PROBLEM DEFINITION

However, Recent research has shown that optimization cloud computing can be conceptualized as an elastomeric network of resource base that are conversing together in order to reduce the amount of time spent waiting and maximize the amount of data processed. Because of this, load balancing and resource management can be highlighted as one of the primary concerns in cloud computing because of the direct impact they have on the performance of the network.

Therefore, some applications experience sudden and unpredictable spikes in traffic due to events such as marketing campaigns or viral content. The problem is to detect such spikes and rapidly scale up resources to accommodate the increased load without causing performance degradation.

Decoupled applications and microservices are gaining traction. These container-based designs have paved the way for the rapid creation of sophisticated SaaS programs. Microservices can have many varied capabilities, from handling information and storage to calculating predictive and prescriptive analytics. Modern cloud architectures include provisioning technologies that increase or decrease the computing power of applications in real time. Unfortunately, these scaling systems have a number of drawbacks, including (i) being limited to only one type of resource, (ii) being unable to meet the requirements for QoS when confronted with of sharp workload, and (iii) providing an equivalent QoS different levels to all customers regardless of factors such as service that is performance and accessibility individual test.

The workload in cloud computing systems exhibits a high degree of dynamism, which is influenced by various factors including temporal variations and the occurrence of certain events. The primary objective is to promptly identify fluctuations in workload and adjust resource allocation accordingly, in order to prevent performance limitations and unnecessary resource utilization.

An autoscaling is to achieve optimal resource allocation in order to effectively address the constantly changing needs of the application in question. The task at hand involves the identification of the optimal quantity of instances to be included or excluded, as well as the allocation of resources such as CPU and memory to each individual instance.

1.9 PROPOSED WORK

Load balancing in cloud computing has emerged as a critical area of focus in tackling the issue of efficient distribution of workloads across computing resources. Many current scaling systems have limitations in that they focus on a singular resource type, encounter difficulties in managing fluctuating workloads, and offer uniform quality of service to all clients, regardless of their individual preferences. This research presents a Dynamic autoscaling system that effectively addresses the aforementioned restrictions through the utilization of diverse resource types and the establishment of different tiers of quality of service (QoS) needs. The system under consideration employs an intelligent approach to determine an optimal resource scaling plan by taking into account the characteristics of the workload as well as the specific requirements of the customer.

The objective of the research work is to provide an innovative strategy that uses real-time load observing natural-based a meta heuristic learning algorithms to strategically distribute on demand requests across the VM instances to design and develop a method of dynamic load-balancing techniques for cloud virtual machines (VMs) in clusters of varying sizes. The proposed firefly algorithms will make intelligent decisions regarding VM selection during load distribution to enhance overall performance by taking into account the unique characteristics of each VM type. The Firefly algorithm mimics the flashing patterns of fireflies to find optimal solutions through the attraction and movement of virtual fireflies in a search space.

Next, autoscaling system tackles these challenges by employing heterogeneous resources, enabling the system to scale efficiently across various resource types. autoscaling with load balancing is a powerful combination that provides improved performance, cost efficiency, and resilience for applications and services in cloud computing environments, making it a crucial aspect of modern cloud-based infrastructure. Dynamic Q -learning learn an optimal load-balancing policy, which is used to assign tasks to the available virtual machines (VMs) in the cloud. The Q-learning algorithm is trained using a reward function that takes into account both the completion time and the resource usage of the tasks. To achieve optimal resource allocation in a dynamic cloud setting, we offer a strategy that combines the strengths of transfer learning and reinforcement learning. In particular, we employ transfer learning to improve the resource allocation process in today's rapidly changing cloud environment by making use of information gained through addressing similar distribution challenges in previous generations. Optimization of the distribution of resources in a dynamic cloud environment can be achieved by drawing on transfer learning within the framework of reinforcement learning.

By taking this method, we may enhance scalability and boost performance in the cloud's dynamic resource utilization process. Finally, The Actor-Critic algorithm combines the benefits of value-based methods (critic) and policy-based methods (actor) to achieve more stable and efficient learning in reinforcement learning tasks. Reinforcement learning is leveraged by the AC-CIWAS technology to assess and reward desirable actions. AC-CIWAS uses a deep reinforcement learning technique to train an actor-network to develop the allocation strategy and a critic-network to evaluate its quality. By incorporating the critic network's criticisms into its decision-making process, the system can improve its allocation method. Reinforcement learning is used to enhance the efficiency of the allotment methodology in the AC-CIWAS system.

1.10 ORGANIZATION OF THE THESIS

Chapter 1 Narrates about the Introduction on Load balancing across geographically distributed data centers challenge is to develop load balancing mechanisms that consider network latency, data replication, and inter-data center communication to optimize resource allocation and improve overall system performance.

Chapter 2 Reviews the previous work done in the field load balancing and autoscaling techniques.

Chapter 3 The Unrelated Parallel Machine Scheduling Problem involving a group of virtual computers can be solved numerically using the suggested augmented firefly algorithm involving engagement learning. Improve performance and allocation of resources using this computational strategy for the best scheduling in distributed computer environments.

Chapter 4 Explains the proposed approach uses a Dynamic Q-learning algorithm to learn an optimal load-balancing policy, which is used to assign tasks to the available virtual machines (VMs) in the cloud. The Dynamic Q-learning algorithm is trained using a reward function that takes into account both the completion time and the resource usage of the tasks.

Chapter 5 The proposed approach, the Actor-Critic based Compute-Intensive Workload Allocation Scheme, aims to capture the dynamic features of server states and consider the influence of different workloads on the demand requirement for workload allocation in order to achieve efficient and logical task allocation.

The final Chapter 6 summarizes the key results, conclusions of this thesis and outlines potential future work.

1.11 SUMMARY

This chapter deals with the brief explanation of dynamic model for resource handling in the cloud environment. These load balancing aspects include make span, energy consumption, and execution cost. Firstly, autoscaling with load balancing enables efficient utilization of computing resources, which leads to cost savings and improved performance. By automatically adjusting the number of instances or virtual machines based on demand, organizations can avoid underutilization during low traffic periods and prevent overload during peak times. This allows them to optimize their infrastructure costs while maintaining a high level of service for users. Consequently, these cost savings can be passed on to customers, resulting in more affordable services for businesses and individuals. Moreover, the use of autoscaling with load balancing enhances system reliability and resilience. Through load balancing algorithms, computing resources are evenly distributed across instances, mitigating the risk of bottlenecks and single points of failure. As a result, applications and services become more robust and less prone to disruptions, ensuring uninterrupted access for users. This is particularly relevant in critical applications such as e-commerce platforms, healthcare systems, and financial services, where any downtime or performance degradation can have severe consequences. By employing a task-scheduling strategy based on an adaptive reinforcement learning model, the recommended MCOC mechanism seeks to make best use of cloud-based virtual machines. To improve the effectiveness and functionality of cloud-based virtual machine job time management, the MCOC mechanism employs a multi-resource collaborative optimization control technique. The suggested MCOC mechanism accomplishes this through a task scheduling strategy based on an adaptive reinforcement learning model.

CHAPTER - 2

LITERATURE REVIEW

2.1 INTRODUCTION

However, the quality of something like the various feasible solutions that are engendered by all these optimization methods becomes disgracefully poor as the volume of the challenge and the multitude of effect was evaluated that need to be streamlined increases. But at the other hand, different users of the cloud want a particular quality of service satisfaction, in special for applications that are used in the academic and technical fields. Recent efforts have been made to problem solve with scheduling tasks by utilizing meta-heuristic algorithms such as Genetic Algorithm (GA), Moth Search Algorithms (MSA), Whale Optimization Algorithms (WOA), Particle Swarm Optimization (PSO), and artificial bee colony optimization. A few examples of these algorithms are: (ACO). The use of meta-heuristic approaches to answer task scheduling issues in cloud services has demonstrated promising advancements in obtaining effective performance by reducing the size of the solution search region[82].

Especially considering that the distribution of resources in the cloud is plagued by a great deal of issues. The use of reinforcement learning improvement methods is also required in order to resolve prediction issues. In most cases, they are utilized in the process of creating system replicas since data. Such models ought to advance by additional data in addition to individuals that have been observed during the course of the training in this sense; Q-leaning is a reinforcement learning exemplary, which means that it acquires the value for action based on the state it is currently in. The adaptability of the Q-learnings method and its capacity to adjust to the changing characteristics of the cloud infrastructure is the primary benefit of utilizing this teaching approach. It is a subfield of artificial intelligence that focuses on enabling intelligent behavior in artificial agents within the context of their environments in order to maximize the concept of cumulative reward. Therefore, one class of problem-solving methods known as reinforcement learning has the potential to perform admirably when applied to the challenge of allocating dynamic resources. Many problems are being brought to light as a result of the distribution of resources in cloud computing environments. Because of the varying amounts of work.

A highly adaptable model that takes into account the collaboration between broker agents. When processing requests made by customers, the model takes into consideration a number of different criteria. However, in order to accommodate the distribution of resources, this solution still requires further development and modification. Autonomous Agent-Based Load Balancing Algorithm (AALB), also known as Autonomous Agent-Based Load Balancing, is a procedure that vigorously achieves a proactive load scheming of VM affording to threshold value. Create a new method known as QMPSO by combining two different optimization strategies—namely, particle swarm optimization and an enhanced version of the Q- learning algorithm. The solution that has been proposed makes it possible to find an appropriate action since the regular of conceivable VM actions in accordance with the state for VMs although simultaneously attaining load balancing for a system[83].

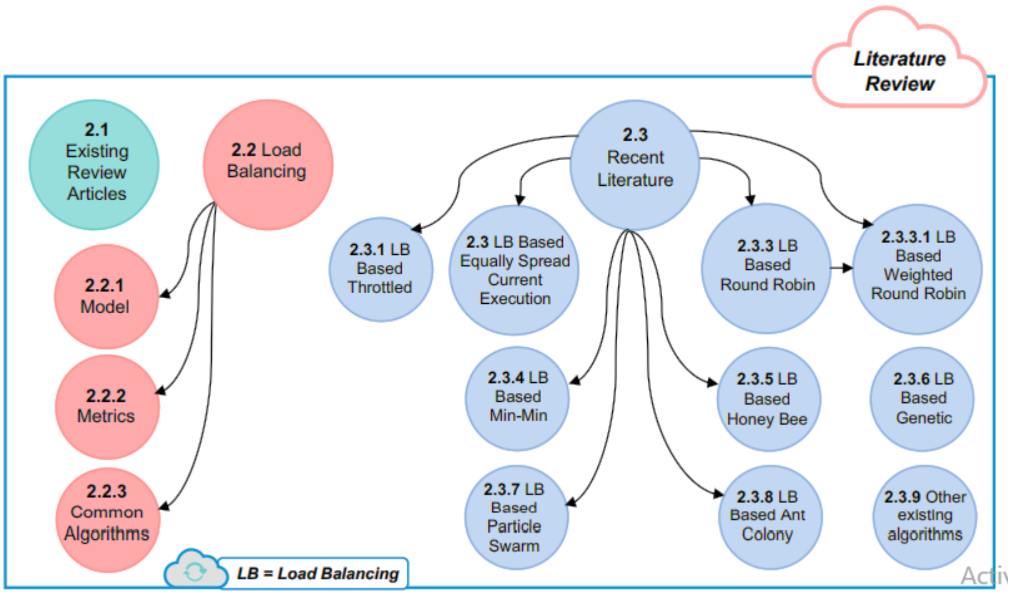

Literature Review
2.1 Existing Review Articles
2.2 Load Balancing
2.2.1 Model
2.2.2 Metrics
2.2.3 Common Algorithms
2.3 Recent Literature
2.3.1 LB Based Throttled
2.3 LB Based Equally Spread Current Execution
2.3.3 LB Based Round Robin
2.3.3.1 LB Based Weighted Round Robin
2.3.4 LB Based Min-Min
2.3.5 LB Based Honey Bee
2.3.6 LB Based Genetic
2.3.7 LB Based Particle Swarm
2.3.8 LB Based Ant Colony
2.3.9 Other existing algorithms
LB = Load Balancing

Figure 2.1 Load balancing Literature Review

This method is both speedy and effective; however, it is not always possible to locate the connected servers, which results in uneven distribution of available resources. The actual operation of the system is not taken into sufficient consideration during the decision-making process, which is the most significant issue with this kind of approach. As shown in the above Figure 2.1 these methods optimize load balancing by mimicking the strategies of real-world organisms like ants, bees, and other social insects. Natural phenomenon-based load balancing strategies strive to ensure efficient and effective allocation of jobs and assets within a system or network by modeling these efforts after the foraging habits of insects like ants and bees.

This method is affected by natural occurrences or the actions of biological organisms. There are several different kinds of Load Balancing LB techniques that are included in the network-responsive task scheduling LB. These techniques include active clustering, biased random sampling, the shortest job scheduling LB algorithm, and task scheduling approach based on LB.

The proposed algorithm considers the capabilities of each virtual machine, the length of requested jobs, and the interdependency of tasks. By migrating tasks to underutilized VMs, the algorithm aims to optimize resource utilization and improve overall system performance. The evaluation of the new algorithm includes comparing its performance with existing methods. This research contributes to the advancement of load balancing techniques in cloud computing and emphasizes the importance of considering various factors in task scheduling. Further research in this area can explore the impact of fault-tolerant systems on load balancing algorithms to improve the overall performance and reliability of cloud computing enviro The proposed algorithm considers the capabilities of each virtual machine, the length of requested jobs, and the interdependency of tasks. By migrating tasks to underutilized VMs, the algorithm aims to optimize resource utilization and improve overall system performance. The evaluation of the new algorithm includes comparing its performance with existing methods. This research contributes to the advancement of load balancing techniques in cloud computing and emphasizes the importance of considering various factors in task scheduling[84].

Designing the administration of Cloud-based resource scheduling is a significant challenge. The scheduling methodology, price, quality of service, timeliness, and request access to service conditions are all important considerations. A good task scheduler should be able to adjust its scheduling methods in response to environmental variables and Cloud service requirements for load balancing. Cloud Computing software is highly desirable because of its potential to revolutionize the IT software market. It has also altered the practices of IT gear manufacturers and retailers.

One of the more frustrating aspects of the cloud infrastructure is load balancing. It guarantees the system's dependability and availability. By dividing the work evenly between parallel processes, it improves overall system performance. Load balancing's key objective is to cut down on response time and costs while increasing throughput. Researchers have developed a variety of strategies over the past few decades to address this problem. Nonetheless, there are still load balancing parameters that could be improved upon.

An increasing number of users necessitates more load traffic, which must be properly managed in order to keep the cloud operating at peak performance. Load balancing is used to equitably distribute work across every one of the accessible nodes, preventing overloads. There are, however, a number of Load balancing difficulties that need to be addressed in order to provide the greatest possible outcome for cloud technology.

Offering an in-depth analysis of current auto-scaling methods that work well in a hybrid setting and can accommodate the varied nature of applications. This paper presents an overview of the outstanding questions and research opportunities around auto-scaling in the cloud. Specifically, it draws attention to the difficulty of determining how many additional resources are needed and the importance of effectively managing dynamic requests with the available resources.

As the number of people who use internet connectivity continues to grow, as do their expectations for the quality of those services, and as more and more people move their data from on-premises servers to the cloud, there is an immediate need to improve our ability to predict application demand and more effectively handle cloud resources. A mathematical explanation of the cloud's autoscaling service's non-stationary influx of user requests is proposed[85].

Having the optimal number of servers available at any one time—a virtue of autoscaling—is highlighted as an interesting strategy for dealing with fluctuating demand in emerging markets. demonstrates how, with moderate market success probabilities and low-capacity costs, autoscaling can reduce price competitiveness.

The offered SMS was developed with a standardized, trustworthy, and rigorous methodology and was deployed in a way that can be replicated and verified. In this method, documents are chosen from reputable digital libraries available online. The goal of this study was to identify and assess the empirical evidence pertaining to cloud computing's application in education in order to determine the most widely discussed areas of study and the areas that have not yet been sufficiently investigated so that they can be incorporated into a research agenda. The proliferation of Information and Communication Technology (ICT) and its applications motivated librarians to adopt automation tools in order to better serve library users.

Automating libraries is a significant advancement in library science. It streamlines internal operations and improves the efficiency of library services. Computing in the cloud is a relatively new innovation in the sphere of communications. The latest news is that libraries are joining the worldwide trend toward cloud computing.

In order to more effectively help library patrons, librarians have adopted automation techniques as a result of the widespread availability of Information and Communication Technologies (ICTs). There has been a major development in library science with the introduction of automation. It helps the library run more smoothly internally and makes library services more effective for patrons. Cloud computing is a recent development in the information technology industry. The most up-to-date information indicates that libraries are following the worldwide trend and transitioning to cloud services[86].

Two corpus screening procedures, one using the double-sentence pair length ratio method and the other using word-based orientation information, are initially proposed centered on the dual-sentence quality corpus screening. These two techniques are novel since they do not rely on supplementary linguistic resources like a multilingual dictionary or a syntactic analyzer. Automatic selection of low-quality sentence pairs across all language pairs with no human interaction. Second, we present a massive-corpus-based domain-adaptive technique.

The proposed method not only eliminates the subjectivity of decision-makers in determining the weights of indicators, but also takes into account the uncertainty and ambiguity of the online education evaluation process. This ensures a more objective and accurate ranking of the satisfaction level of different colleges in the context of online education. By incorporating the cloud model and possibility degree matrix into the entropy method, the proposed fuzzy TOPSIS approach provides a robust and reliable framework for evaluating the satisfaction of online education in different colleges. This approach can be applied to various practical problems where the weight information may be unknown or uncertain[87].

The utilization of a computer was necessary in order to achieve this goal. In addition to this, it is responsible for scheduling as well as making certain that the available resources are utilized in an efficient manner. The user will send requests through the internet, and the Virtual Machines will store these requests once they are received (VMs). CSPs are needed to maintain quality of the service in all delivery models by ensuring that the requests generated by participants can be performed and accomplished within such a predetermined amount of time. This obligation applies to all requests made by users. As shown in the above figure 2.2. A scheduling policy be able to result in a workload that is equitably spread among computers and server farms, is required for the process of attributing user requests to the right VMs. This process is dependent on the scheduling policy. Two steps that can be taken to improve the efficiency of planning and resource utilization are the invention of a various loading balancer and the carrying out of planning activities.

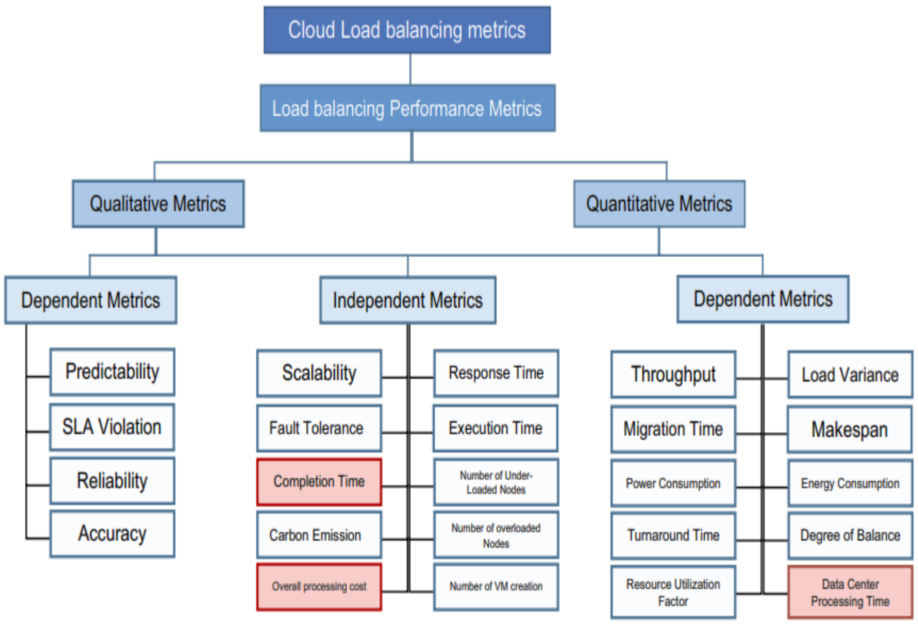


Figure 2.2 Metrics for load balancing.

2.2 LOAD BALANCING METRICS

When examining the most recent state of the art, the majority of authors take into consideration the metrics that are provided in the following subsection. When it comes to manipulative and building a load-balancing automated system, these metrics are absolutely necessary. In definitions of how well it performs in cloud applications, the performance of the automated system can be determined using these metrics. The investigators use some of these parameters in order to test the proposed algorithm, and it is important that these parameters be adjusted in order to avoid imbalance situations from occurring within the surroundings of cloud computing. The writers of emphasized the metrics that were used in the previous research, and they divided these metrics into two categories: The taxonomy diagram presents both qualitative and quantitative aspects of research [88].

The model presented below has been updated with the addition of new parameters derived from more recent research in order to present the most recent information possible and enhance the reliability of a figure. This analysis paper centered its analysis of recent literature on the following nine primary qualitative and quantitative metrics, which are enumerated and described below:

2.2.1 Resource Utilization (RU)

RESOURCE UTILIZATION (RU) refers to a degree to which has system's resources (such as CPU, memory, and so on) is being put to use. The purpose of this was determining the level of RU present in a cloud data center. RU will continue to be necessary so long has there is a growth in the demand of services. It has necessary to have a thoroughgoing RU in order for the load-balancing algorithm to function correctly.

Scalability (S): An algorithm, just like a system, should work effectively in any unexpected circumstances. That is, the algorithm ought to be able to maintain its level of scalability even if the number of jobs and also load are both increased. Highly scalable in order to achieve satisfactory results with the load-balancing algorithm.

The term "throughput" (TP) refers to the measurement of the amount of job requests that could successfully carried out and managed in a given amount of time in the VIRTUAL MACHINE (VM). It refers to the quantity of information that is moved since one location to other. A in elevation TP is required for optimal performance of the algorithm for load balancing.

Response Time, abbreviated RT, refers to the quantity of time it takes an algorithm to reply to a given task. It considers into version both the time spent waiting as well as the time spent transmitting and providing the service. It is the amount of time required to respond to an inquiry made by a user. A good algorithm for load balancing will require that there be a minimum RT.

2.2.2 Make Span (MS)

MAKE SPAN (MS) refers to the total amount of time needed to appearance for all of the tasks and dispense resources to the various users in a system. In the context of the cloud environment, the scheduling process, this metric plays a crucial role. The purpose of this is to determine how extensive it proceeds to perform a sequence of tasks[89]. A virtuous algorithm for load balancing should have Minimum MS requirement.

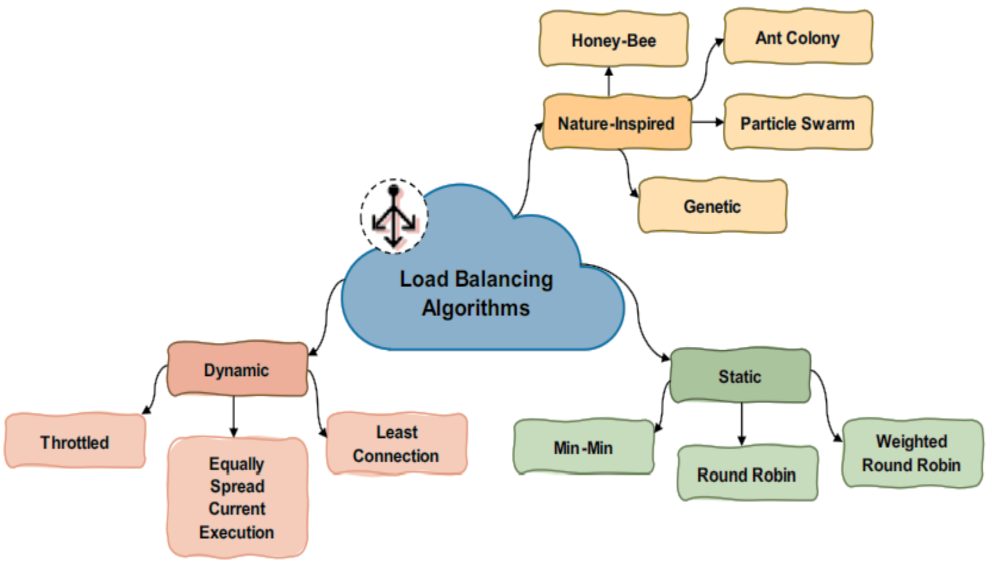


Figure 2.3 Load balancing Algorithm

2.2.3 Associated Overhead (AO)

ASSOCIATED OVERHEAD (AO) refers to the number of administrative costs that is created while the load-balancing algorithm is being executed. It's possible that an excessive amount of task migration and co - ordination communication are to blame. The load balancing algorithm will have the least amount of overhead to deal with if the platform's load is balanced.

2.2.4 Fault Tolerance (FT)

FAULT TOLERANCE (FT) is the ability of a load balancing system to continue operating effectively in spite of the malfunction of some components of the system. This means that if one VM is too busy, another VM that is available should still be able to carry out tasks. A elevated FT is mandatory for optimal pre-forming an algorithm for load balancing.

The total period of hours required to transfer a task since one virtual machine (VM) to another is referred to as the MIGRATION TIME (MT). That has expected for migration process will proceed without disrupting the availability of the system. It is extremely dependent on the idea of virtual machines that is used in the cloud. A reduced MT is required for a successful achievement of the load-balancing algorithm.

SLA Violation: Designates the number of declines in SLA violation aspects with regard to the deadline restriction, significance, and so on. SLA violations happen while resources, such as virtual machines (VMs), are inaccessible since they are operating at their maximum capacity. Minimum SLA to ensure a greater degree of overall user satisfaction.

Collective static algorithms are use in a cloud environment, including like Round-Robin, were no longer effective and suitable owed to the numerous restrictions. One of these constraints is the uneven delivery of load on nodes, in which approximately machines might convert overwhelmed although others might be permitted of load. Both these limitations include the lack of scalability in the cloud environment. In adding to the utilization of static quantum, that also leads to the occurrence of multitasking and, as a consequence, delays both the completion of tasks and their rejection, this results in an improper task allocation as well as an uneven distribution of the course load[90].

As a result, a large number of authors have made contributions to the development of algorithms that object to progress the performance for cloud applications through utilizing the concept of load balancing. As shown in the above Figure 2.3. Developers require the possibility of placing these approaches into action either on the user end, where they would be implementing what is recognized as Service Broker Policy, or on the data center end. The following graphic depicts nomenclature for collective load-balancing algorithms that are currently available.

This statue's objective is to categories algorithms according to the primary key that is taken into account in the review paper that is being presented. In order to improve CC's overall performance, various load-balancing algorithms that are commonly used are applied. The underpinning surroundings of these algorithms is typically used to classify them into one of three primary types: static, vibrant, or nature-inspired algorithms, as will be discussed further down in this article.

2.2.5 Static Load Balancing (SLB)

STATIC LOAD BALANCING (SLB) Algorithms has a closed environment, the process of load balancing algorithms be contingent on the proper understanding of the system state along with its characteristics and abilities. Static load balancing algorithms are also known as SLB algorithms. Storage capacity, Memory, and processing power are a few examples of the types of information that could be considered prior. This information reflects how much work is being done by the system. The dynamic properties in the load that occur during the runtime are ignored by algorithms that are based on static data Therefore, the most significant disadvantage of these algorithms is their low fault tolerance, which is caused by abrupt load changes.

2.2.6 Dynamic Load Balancing (DLB)

DYNAMIC LOAD BALANCING (DLB) Algorithms: it is common knowledge that algorithms of this type are superior and more malleable when it comes to load balancing. In dissimilarity to SLB algorithms, load balancing algorithms for dynamic environments take account the program's history in order to determine how to distribute the load.

2.2.7 Nature-Inspired Load Balancing (NLB)

NATURE-INSPIRED LOAD BALANCING (NLB) Algorithms such algorithms represent biological procedures or undertakings centered on human nature, like as the genetic procedure of the pattern that bees use to find honey. NLB stands for "nature-inspired load balancing," and it refers to algorithms that are inspired by natural phenomena[91].

In needed to execute load balancing in CC, these progressions are precisely modeled to adopt the natural manners that are used. The progress of such intelligent algorithms has resulted in enhanced performance on behalf of systems that were together dynamic and complex.

It is preferable to use dynamic algorithms rather than static ones because dynamic algorithms require no prior knowledge and take into account the current state for system, making them more competent for use in distributed cloud computing environments. Additionally, dynamic algorithms do away with the overhead of storage the preceding condition of the network, but such algorithms obligate a relatively high runtime complexity in comparison to static algorithms. Since nature-inspired algorithms remain part of the metaheuristic class of load balancing algorithms and therefore can bring to mind natural phenomena that are tried to explain by natural sciences, it is believed that they are more intellectual than another load balancing algorithms.

This type of algorithm is known as a dynamic LB algorithm and is abbreviated as TA. When it receives a request from a client, the load balancer's job is to look for an appropriate virtual machine (VM) that can carry out the tasks, as can be seen in the following example. TA will possess a list of altogether VMs alongside through their index value. This list is kept in what is notorious as that of the index table or a allocation table, and it also stores the respective state of VMs, such as available, busy, or idle. The task is considered complete and given to a VM if it can be accepted on that VM and it has sufficient storage space. TA will return 1 if it is unable to locate any VMs that are available; otherwise, the request will be placed in a queue for expedited handling[92].

TA achieves improved results than compared to Round Robin algorithm in terms of performance; however, it does not take into account more progressive necessities for load balancing, like as Process works in a way that is analogous to the customary TA algorithm in that it keeps an index table of everything VMs beside with their respective states. Though, the authors improved both the Response Time and the utilization of the VMs by choosing the VM at the first index if it is accessible and then assigning a request before returning 1 for Data Center. After a first index has been selected, the virtual machines (VMs) will be moved, and so on. This is dissimilar from traditional TA, in which the virtual machines (VMs) at the first index are chosen whenever there is a request. Researchers in the study presented a priority approach called "PMTA," which is based on a modified version of the throttled algorithm and has a faster execution time than the existing algorithm. The primary focus is on dividing up incoming work among multiple virtual machines in an equitable manner. This is accomplished through the utilization of a switching queue, which prevents lower priority tasks from running in order to make room for higher priority ones.

In spite of the fact that the method reduced both the Response Time and the waiting time in comparison to a TA and RR algorithm that was previously in use, it could still result in low priority jobs suffering from starvation and a prolonged

Response Time. Keeping two tables of VMs through their states (busy and available) is how the TMA addresses the problem of equally distributing the workload [93].

Equally Spread Current Execution, also known as ESCE, is a form of dynamic algorithm that is used in load balancing. It gives priority to the scope of the job at hand and randomly assigns the workload to one of the VMs that have the least amount of work to do. Because it distributes the work to multiple nodes simultaneously, this method is also acknowledged as the Spread Spectrum technique. ESCE trusts on the utilization of a backlog to both store requests and distribute load to VMs in the event that a VM is found to be operating at an excessively high capacity. Because of the communication and interaction among the Data Center controller and the load balancer, one of the most common issues with ESCE is that it may result in a growth in the amount of time obligatory to update the index table.

The ESCE algorithm's process is further broken down and explained in the flowchart that can be found below in the authors propose a method for reducing response time in CC that is a hybrid approach that associations of mutually EQUALLY SPREAD CONCURRENT EXECUTION (ESCE) and TRANSACTIONAL APPROPRIATENESS (TA). The Hybrid LB (TA & ESCE) maintains a HashMap list in which it stores the user requests. It formerly scans over this list in order to locate the accessible VMs. In contrast to TA, which returns 1 if all of the VMs are currently in use, ESCE searches for a machine that has the least amount of work to do and then assigns the task to that machine. Authors in propose yet another method that is analogous to the one used in Sachdeva.

The purpose of the Enhanced LB (TA & ESCE) is to raise resources even further. They plan to use a generate host function in order to cut down on the quantity of time that users have to delay in the queue somewhat than continuing a queue in case VMs are currently in use. Together algorithms are operative at lowering the Response Time; though, neither one receipts into interpretation the relocation of virtual machines (VMs) in the event that a problem occurs.

The method allots requests by iteratively searching over a hash map which effectively compiles altogether of the VMs in order to solve the problems associated with RR. If there isn't a virtual machine available to perform the task, a good one will be selected through the use of GA analysis to determine which tasks are the best fit.

According to the findings, the algorithm has a significant impact on lowering the Response Time of servers. On the other hand, when the search space expands, GA has a tendency to become more complicated[94].

Taking into account the burstiness workload problem, researchers have also enhanced QoS in cloud applications.   Load balancing issues might arise in cloud services if the number of users suddenly spikes, hence these two cases need to be considered while designing cloud infrastructure. For this reason, a technique known as Adaptive Load Balancing (Adaptive LB (RR + Random)) has been proposed to guarantee a fair delivery of incoming work among all VMs, regardless of the load. To achieve this, we move between a random task scheduling policy and an RR task scheduling policy depending on whether or not the workload is typical or busty. Response Time is shown to be reduced using adaptive LB, although a static quantum is imposed when using RR, which also increases waiting time. There is less waiting time with adaptive LB. This hybrid strategy presented by combines the priority policy with the RR algorithm.

The enriched RR technique employs a pair of processors, one of which calculates the time slice for apiece process and the other of which sorts the processes in ascending order according to their burst time (the priority value). The plan's goal is to ease Response Time; yet it lacks any sort of performance assessment to see if it's actually working. Conversely, if there isn't a free VM that can be owed to the incoming user request, a new VM will be created. Using a technique named as unfold spectrum, the ESCE algorithm distributes the network's load over a set of nodes[95].

Response Time service broker policy as it relates to Cloud Analyst. Before choosing which process should be scheduled, the algorithm computes the Response Time and waiting time for respectively process. The Response Time could be lowered, but the classifier wouldn't be as useful in dynamic public clouds because of the lack of attention paid to the time quantum problem. In their article, In this article, the author demonstrates how the Genetic Algorithm (GA) technique can be used to increase RR. Its goal is to progress the capabilities of data centers by given that effectual and effective load balancing. To address the concerns with RR, the approach distributes requests through repeatedly searching a hash map that effectively compiles all of the VMs. In the absence of a suitable virtual machine, GA analysis will be used to establish which jobs are the best fit, and then a suitable virtual machine will be chosen.

The results show that the algorithm significantly reduces servers' Response Times. However, GA tends to get more difficult as the search space grows. Taking into account the burstiness workload problem, researchers have enhanced the Quality of Service (QoS) in cloud applications. Load balancing issues might arise in cloud services if their user base suddenly expands, hence these conditions must be taken into mind while designing cloud infrastructure. For this reason, it has been proposed to use a method called Adaptive Load Balancing (Adaptive LB (RR + Random)) to distribute jobs to VMs fairly even when the workload is high. This is achieved by switching between random task scheduling and RR task scheduling strategies, the former being used under normal conditions and the latter under heavy loads. Although the results of adaptive LB indicate a decrease in Response Time, using RR leads to a application for static quantum, which in turn bases increased waiting time [96].

The Waiting Time is also decreased by Adaptive LB.

Priority policy and the RR algorithms are combined in the combination technique presented by. Two processors are used in the Enhanced RR method: a small processor calculates the time slice for respectively process, and a central processing unit ranks the processes according to its burst time, which serves as the priority value. The plan's goal is to ease Response Time, conversely it lacks any sort of performance assessment to see if it's actually working.

The Weighted Round Robin (WRR) algorithm takes into account the weight of each node and uses a middle approach similar to the traditional Round Robin (RR) process. This value is determined by the developer and applied to the highest-capacity VM in the system. When deciding which virtual machine to utilize, this algorithm is quite accurate at predicting how long a user will have to wait, but it ignores the varying task durations. To reduce the amount of Response Time in general, a new and improved RR technique is developed, which makes use of a hash map to keep the most recent entry that was provided by a user base. Its purpose has decrease both waiting and response times and it is based on the permutation of Weighted RR and Max-Min (W Max-Min). Max-Min first determines which jobs need the most CPU time, and then weighted RR decides which virtual machines with the most processing power (in terms of RAM, MIPS, etc.) should handle those jobs. These two methods are useless in dynamic situations. Priority queries are taken into account by this approach. This causes the unusual load balancer to be divided into two components, the regular requests and the insistent requests. To determine how many requests may be processed at once, it employs a Weighted Round Robin (WRR) mechanism in which separately VM shows its load for a server (also acknowledged as the load balancer). The running time of this technique is higher than that of WRR, and it adopts that every requests are of similar value[97].

The suggested approach provides a matrix for storing jobs. It does this by allocating its resource (a virtual machine) to the tasks according to how long it will take to finish and run each one. The usual minimum query execution time and the completion time (CT) of the responsibilities running on the VM are used to determine the scheduling of jobs. However, the proposed solution does not account for the present and revised load that is being borne by the VMs in the process of task allocation. This is a flaw in the methodology. There was a suggestion in the research literature for an extra algorithm called Efficient Load Balancing Min-Max (ELBMM).

It identifies the quickest-running jobs and assigns them to the virtual machines (VMs) and resources with the fastest throughput. While this improves efficiency and throughput, it does not account for the importance of individual tasks, therefore Response Time remains high at 80 ms. Resource based Load balanced Min-Min (RBLMM) is a technique suggested by the Shanthan project's authors and others to reduce the Make span and distribute work fairly among VMs. Once the energy source has been assigned, a calculation is made to determine the Make span time's value, and this number is then used to set a threshold.

The research indicates that whereas the Make span valuation for the standard MM method is ten seconds, the proposed RBLMM algorithm's maximum completion (Makespan) is only seven seconds, a significant time savings. However, the approach does not account for the importance of the activities, the deadline, or any other major QoS-related characteristics[98].

The Honey Bee Algorithm (HB) works on the premise a subset of bees, called the seeking bees, will move about in search of food bases and relay that evidence to the rest of the bees. That was widely acknowledged as a potent and versatile tool for addressing strategic planning and categorization issues As the subsequent flowchart illustrates, the process of finding food and assessing whether or not the bees are healthy enough to reproduce is repeated at each stage.

The authors of (Proposed LB based Bee Colony) suggest a load-balancing algorithm centered on Bee Colony. It sorts virtual machines (VMs) into groups based on workload, transfers jobs from one VM to another, and so on. Both the size and the rate are considered when calculating the VM load. When the data center's capacity exceeds the limit, balancing must be carried out. The algorithm takes into account the relative importance of the tasks, which enhances the level of service provided[99]. The Honey Bee Load Balancing Algorithm (LBA HB) is a technique proposed in that is similar to. Like the previous approach, this one considers the current job count when assigning work, but in addition, a priority value is assigned to each virtual machine. Overloading of virtual machines is also avoided. To ensure that virtual machines (VMs) are kept up-to-date whenever a task is assigned, the authors employ the Honey Bee behavior idea.

The order in which jobs are added to the queue is determined by the First Come, First Serve (FCFS) algorithm. To avoid having work delayed while waiting for the next available virtual machine (VM), it is preferable to assign tasks to VMs with the lowest workloads.

The results show that paralleled to the modified TA and RR, the LBA HB algorithm has a noticeably slower response time. However, since FCFS is being used, there is no requirement for jobs to be completed in any particular order. A similar approach, involving the allocation of weights to virtual machines, is proposed by an algorithm called improved HB Behavior Load Balancing (Enhanced HBBLB). Priority of the task is deliberated already allocating it to a virtual machine. According to the results, employing Enhanced HBBLB decreases response time through 0.15 ms, although it may rise waiting time on behalf of low-priority operations.

Reduced time spent searching for available tasks is a main goal of the Honey Bee behavior Load Balancing (HBB-LB) algorithm. The public cloud is then segmented into three regions based on how much load each virtual machine (VM) contributes: underloaded, balanced, and overloaded. The system then determines whether or not to relocate any virtual machines by calculating the load differential and rejecting the option if the result is larger than zero. It also considers VMs' priority values, such as performance and cost, but does not address the case where these values are both equally significant[100]. .

The method is supposed to resolve the significance issue with the HB algorithm, but consuming RR would static present concerns when dealing with a large number of processes because the quantum employed by this algorithm is fixed.

The concept behind this technique, the Genetic Algorithm (GA), is based on evolution in the wild. As a result of addressing the issue of limited resources, this approach is effective in achieving multi-objective improvement. However, as the size of the search space grows, the GA algorithm tends to get more complex and run time increases. Using the centralized controller, the cloud is divided into four sections, allowing for more effective scheduling and distribution of workloads. Partition Based Load Balancing Algorithm has a programmed that looks with an best division for work allocation, and it is one half of Cloud Partition Based Load Balancing (CPBLB)[101]. The Determination of Reload Period Algorithm calculates the optimal time between reloads for a given system. These two approaches are often employed in tandem with one another.

While this approach can reduce processing time, it does not dictate how the split should be structured. Singh and Prakash created a technique called Weighted Active Monitoring Load Balancing (WAMLB). Since the relative importance of each virtual machine is calculated based on its bandwidth, number of processors, and speed, this method workings fine for a heterogeneous network. The Data Centre receives the ID and updates the allocation table accordingly. This technique attempts to reduce cloud response times without considering the use of virtual machines (VMs).

This approach is introduced to handle the importance of virtual machines. A common term for this device is "Central Load Balancer" (CLB). CLB weighs factors like VM memory and CPU speed when deciding which ones to priorities. The approach is fantastic because it links all users and makes full use of the VMs' resources; conversely, the priority that is allocated for VM has stable, so it might not function capably if there were too various variations to the system's servers and imperative priority must be taken into account. This algorithm, called Dynamic Load Management, makes sure the workload is evenly distributed based on things like the VM's present health. The result is a quicker response.

With dynamic load balancing, we can make better use of our resources while cutting down on the time it takes to complete a Make period. Using an algorithm called bubble sort, it ranks jobs based on how long they take to complete and how quickly they can be handled. After that, the VMs' workloads are divided up using the First-In-First-Out (FCFS) method. Next, load balancing is performed, during which time the workload of each virtual machine (VM) is determined. The method is efficient at making the most of available resources, but since First-Come, First-Served (FCFS) is applied, priority is not considered[102].

A heuristic algorithm known as HBLBA is proposed for use with the IaaS model in data center configuration based on the amount of size, tasks, and also VM compatibility; this is done in an effort to address the issue of inefficient task allocation. This will enable better VM selection and higher productivity. While the method performs admirably when applied to a small set of jobs, it may not be as helpful when applied to a enormous set of tasks within a system. Further setup data could potentially increase the time required for the operation[103].

For count value is below the usual, the virtual machine with the lower value will be given the assignment.

While this method provides a beautiful representation of DLB, it does little more than track score and ignores key job criteria. To improve the system's overall QoS, the authors of presented a new cloudlet allocation mechanism that makes use of a more efficient load balancing technique. The plan's objective is to decrease the total amount of time it takes for cloudlets, VMs, and hosts to accomplish their goals. The cloudlet was given to the VM that could take the most simultaneous requests. The large Make span value for both the VMs and the hosts is unfortunate, but necessary to maintain system activity and equilibrium. That's a major issue. The experiment is not scalable in a high-traffic environment because it was built to support no more than three VMs.

To ensure that all Data Centre VMs are equally taxed, Cloud Analysts should implement a policy that prioritizes the least-used VM mechanism. It may be inferred from the findings that the method outperforms the majority of previously-standing load-balancing techniques in Cloud Analyst. Conversely, nearby is no non-consistent methodology performance testing available on this platform[104].

This idea can be seen in action with the updated load balancing strategy. After determining the user's location, it selects the Data Centre that is physically nearest to them and measures how much that facility is being accessed.

The author provides a cloud computing job scheduling strategy based on reinforcement learning that can adapt to changing user requests and increase system performance. When compared to alternative scheduling algorithms, the suggested solution beats them with respect to of response time as well as the basis of makes pan while also enhancing the effectiveness of resources[105].

This study provides a Microservice Allocation strategy centered around Reinforcement Learning to overcome the difficulty of choosing a suitable node to house containers housing microservice applications in Cloud infrastructures. The proposed method is tested on real Google cluster traces, where it is proven to successfully lower SLA violations and cut down on energy use.

In order to strike a good balance between energy efficiency, diagnostic accuracy, and processing latency, the research offers a reinforcement learning-based solution for scheduling tasks in mobile cloud computing [106].This paper provides a comprehensive overview of deep reinforcement learning (DRL)-based scheduling techniques for use in the cloud. The authors explore the obstacles and possible future directions of DRL in the context of cloud computing scheduling. The paper finds that DRL has great potential in cloud computing resource scheduling, where it may be utilized to boost service quality, cut costs, and eliminate pollution.

To combat the difficulty of cloud elasticity, the research suggests a new reinforcement learning strategy based on adaptive state space partitioning. The method begins with a single environment representing a global state and, depending on the workload as well as how the system operates, divides it into increasingly granular states. The suggested algorithm may operate on a multiple dimensions continuous state space by dynamically partitioning the state space when necessary, and it can adapt the state universe dimension in accordance with the quantity of data collected on the operation of the system.

The research suggests an innovative approach for management based on interactive learning that has great potential for highly adaptive control systems. A proposed reinforcement learning strategy to controlling a partially-observed system in real time does not necessitate any prior model specification. The system's autonomous managerial skills come courtesy of the learned agent.

The host is then partitioned into an overloaded portion and a normal portion. The approach involves shutting down the host machine that isn't being heavily used after process migration. However, the effectiveness of the system has not been evaluated. In order to perform load balancing, the technique Roulette: A Selection method for Wheels (RWSALB) takes into account the weight value for respectively virtual machine. First, it collects VM metadata, and then it sends requests to the VM with the highest priority. Each VM has a weighted selection probability that reflects its relative importance in the model. In that instance, requests will wait in a queue until they can be serviced. While the algorithm has the potential to reduce data transfer costs, it still takes a long time to respond to and service requests.

To ensure a consistent and high level of service, the authors of this paper suggested a dynamic algorithm which accounts for network latency and is dubbed the Dynamic Cost-Load Aware Service Broker (DCLASB). The VMs are sorted by their processing speeds, and the VM used is selected founded on the size of the request[107].

Consideration is given to selecting the Data Centre with the shortest processing time, the Data Centre with the lowest latency, and the Data Centre with the lowest cost. The algorithm can get good results in the cloud; however, it doesn't consider the priority of the user's requests. The authors of this study propose an innovative algorithm that exemplifies the fundamental method for confederacy VMs in a cloud-based distributed environment; they call it the Flexible Load Sharing (FLS) algorithm. Thanks to cloud nodes distributing work and exchanging data, the percentage of overworked virtual machines is going down. In addition to improving cloud performance, this technique also reduces the amount of time required by individual VMs.

Starvation Threshold-Based Load Balancing (STLB) is a proposed technique that limits load balancing to times when there was at least one idle VM (starvation state). Task waves of immigration through one VM to the next are reduced, and unnecessary overhead costs are avoided, because it only attempts to deal with simple nodes for the purpose of workload balance. The technique, however, can only be used successfully in implementations that are completely decoupled from one another[108].

Continued development of the LB (Priority Basis) algorithm, which classifies tasks as either "cancelable," "strong cases," or "non-preemptable," and which needs the type of task if two or more organizations expectations are the same. The enhanced LB algorithm (Priority Basis) gives higher priority to jobs that can be halted at any time and executes them before those that cannot. There is a high risk of failure with this algorithm, but it has the improvement of dropping response and waiting time as well as processing cost[109].

In this approach, we employ a space-sharing policy for Cluster-Based LB. In the event that one virtual machine (VM) is unable to finish the task at hand, the burden can be divided among numerous VMs, resulting in a quick response. While this approach uses k-means clustering, it does so exclusively for VMs and not for tasks. Both the minimum and maximum capacity of virtual machines are set by the cluster.

The system's responsiveness increases due to the algorithm's ability to reduce the time spent scanning the complete list in VMs. In contrast to the Active load balancer, the authors has study developed a load balancer that, before allocating user requests, examines the previously allocated request in a least loaded VM. This technique is known as VM-Assign Load Balancing. The VM-Assign Load Balancing load balancer is what you need. A request is subsequently issued to the VM, and the Data Centre is provided with the VM's identifier[110]. If not, the process of finding the VM with the lightest workload continues.

2.3 RESEARCH GAP

Enhancing the performance and reliability of cloud databases is crucial for providing efficient and scalable data management services. To achieve this, dynamic load balancing techniques are employed to distribute the database workload among multiple instances effectively. In cloud database environments, the demand for data access can be highly variable, and individual instances may become overloaded during peak times. As a result, there is a need for load balancing techniques that can efficiently distribute the workload across multiple database instances to prevent performance bottlenecks and ensure smooth operations. The "No Instance" scenario arises when a single instance is unable to handle the incoming queries or transactions efficiently, leading to degraded performance and response times.

Dynamic Resource Allocation: Reliable dynamic load balancing techniques dynamically allocate resources to database instances based on their workload and performance metrics. This may involve scaling up or down the resources allocated to each instance in real-time to meet the changing demands effectively. To enhance reliability, load balancing techniques should consider fault tolerance and high availability. This involves detecting and handling instance failures, redistributing the workload to healthy instances, and ensuring data consistency during failover scenarios.

However, Autoscaling is a key resource management technique that automates the process of adding or removing resources in response to changes in the workload. Autoscaling policies are defined based on performance thresholds, and the load balancer dynamically adjusts the number of instances or resources to meet these thresholds. the workload is partitioned into smaller tasks, and these tasks are distributed among different database instances to achieve a balanced load distribution. The load balancing algorithm should consider factors like query complexity, data access patterns, and resource availability to determine the optimal task distribution.

Finally Predictive resource management techniques use historical data and machine learning algorithms to forecast future workload patterns. By predicting demand spikes or resource requirements, load balancers can proactively allocate resources and handle workload variations effectively. Different applications or services may have varying QoS demands, such as response time, throughput, and availability. QoS-aware resource allocation ensures that the load balancer meets these requirements while maintaining resource efficiency.

2.4 CONTRIBUTIONS

The main contribution of the thesis is as follows:

Initially proposed system self-healing in load balancing includes the attractiveness scaling factor, which improves its convergence speed, exploration-exploitation balance, and handling of complex optimization problems. Using the firefly algorithm comparing with various load balancing algorithm finds the objective fitness of the attractive various instance nearby. we find the effective resource management techniques are vital for dynamic load balancing in the cloud environment. These techniques optimize resource utilization, enable seamless scaling, and ensure high availability, contributing to the overall effectiveness and adaptability of load-balancing mechanisms. The combination of resource monitoring, autoscaling, VM migration, task scheduling, and predictive resource management empowers load balancers to efficiently handle varying workloads and changing demands in a cloud-based infrastructure.

Compared to traditional methods, the Dynamic Queue ML model based on Actor-Critic Deep Reinforcement Learning algorithms demonstrates superior performance in terms of minimum task completion time, response time, and reduced CPU utilization. This makes it a promising solution for optimizing various systems and processes, ensuring efficient resource allocation and overall improved performance. Furthermore, the Dynamic Queue ML model also outperforms the state-of-the-art in various metrics such as energy savings, cost, strength index improvement and total system throughput. By leveraging the power of Actor-Critic Deep Reinforcement Learning algorithms, the Dynamic Queue ML model has proven to be a significant advancement over traditional methods.

CHAPTER - 3

THE FIREFLY TECHNIQUE WITH COURTSHIP TRAINING OPTIMIZED FOR LOAD BALANCED WITH TASK SCHEDULED IN THE CLOUD

3.1 INTRODUCTION

Load balancing is a key performance indicator that can be used to optimize a system. Scheduling and load balancing in the cloud are difficult because of the heterogeneity and dispersed nature of cloud resources. Although there are a variety of ways for allocating jobs to suitable virtual machines (VMs), load balancing remains a significant challenge due to variations in workload and VM specs. The firefly method is used in a meta-heuristic scheme to distribute work among numerous servers. The primary result of this study is a method for balancing workloads to maximize the speed with which user-submitted tasks can be processed. Dependent and independent activities are used to implement the expected multi-server load balancing. The jobs are a collection of related errands, each of which may require processing in many virtual machine cores or even in other virtual machines[111]. The Firefly algorithm decides which Virtual Machines (VMs) are best suited for task assignment and moves tasks from overburdened VMs to less busy ones. This technique greatly reduces the likelihood of uneven cloud-based workload performance. Existing LB techniques like Dynamic LB (DLB), Hybrid Dynamic LB (HDLB), and Honey Bee Behavior LB (HBB-LB) are compared with the performance of the suggested Firefly Algorithm method to see which is more capable of handling a given workload. This adaptability is particularly valuable in real-time systems, where the workload is unpredictable, and the load balancing solution must dynamically adapt to the changing environment. Future research can explore the effectiveness of the Firefly Algorithm in dynamic load balancing scenarios and identify strategies to improve its performance further in such settings. Exhaustive search-based solutions to the job scheduling problem are impracticable in a cloud setting. Scheduling issues in the cloud can be tackled with the help of metaheuristic approaches like Ant Colony Optimizations (ACO), Particle Swarm Optimizations (PSO), Genetic Algorithm (GA), Artificial Bee Colony (ABC), Crowd Search Algorithm (CSA), and Penguins Search Optimizations Algorithm

(PeSOA)[112]. Furthermore, quantitative analysis of PeSOA, GA, PSO, ACO, CSA, and ABC must be performed according to a number of factors. When simulating a cloud system, it is important to take into account performance restrictions like as load, resource utilization cost, makes pan, and quality of service as a means of zeroing down on the best metaheuristic approach to use.

3.2 LOAD BALANCING IN A CLOUD COMPUTING ENVIRONMENT

Load balancing is a critical aspect in cloud computing, as it ensures the effective utilization of resources and prevents the overloading of individual resources. By distributing the workload dynamically across all nodes, load balancing maximizes transmission, minimizes data center processing time, and improves user response time. Additionally, load balancing in cloud computing addresses the issue of scalability. As the number of cloud users continues to grow rapidly, load balancing becomes a key challenge for cloud computing. Without effective load balancing, the heavy workload on cloud servers can result in performance issues, such as slower response times, decreased throughput, and even system crashes. Load balancing algorithms play a crucial role in achieving efficient resource utilization and maintaining optimal performance in cloud computing[113]. The Enhanced Firefly algorithm, developed by incorporating key concepts from the Firefly approach, aims to improve the search capability and understand global optimized solutions. By implementing an enhanced firefly feedback mechanism and improved communication, this algorithm is able to reduce variance in execution time, make span time, and the number of migrations. This enhancement in performance and efficiency makes the Enhanced Firefly algorithm a valuable tool in optimization tasks, especially in the Cloud Sim simulation environment. The Enhanced Firefly algorithm, through its incorporation of key concepts from the Firefly approach, offers an improved solution for optimization tasks. By enhancing the firefly feedback mechanism and improving communication, the algorithm is able to achieve better results in terms of reduced variance in execution time, make span[114].

3.3 FIREFLY MODEL FOR ATTRACTIVE MECHANISM

The impact of different parameter settings on the performance of the Firefly Algorithm in load balancing scenarios. Identifying the optimal parameter values, such as the firefly attractiveness, randomness factor, population size, and termination conditions, could enhance the algorithm's convergence speed, accuracy, and overall performance[115]. Additionally, comparison studies could be conducted to assess the Firefly Algorithm's performance in relation to other well-established load balancing algorithms, providing insights into its comparative strengths and weaknesses.

- **Objective Function**: The Firefly Algorithm aims to optimize a given objective function. The objective function represents the problem to be solved and is typically defined by the user based on their specific requirements.
- **Light Intensity**: Fireflies communicate with each other through the intensity of their flashing light. In the algorithm, light intensity represents the fitness or quality of a solution. Higher intensity indicates a better solution.
- **Attraction Mechanism**: Fireflies are attracted to each other based on their relative brightness or fitness. The attractiveness is determined by the distance between fireflies and their light intensity. Fireflies move towards brighter fireflies in search of better solutions.

$$EC = \sum_{j=1}^{\text{Nmber of task}} \left[\frac{\text{Execution time} \times \text{Communication time}}{\text{Number of task}}\right] \qquad Equ\ (3.1)$$

*Equation 3.1 explains***,** The Energy Consumption (EC) for a set of tasks running in a cloud computing environment. As shown in the below Table 3.1 . For each task j, the formula computes the energy consumption for that specific task, and then sums up the energy consumption for all tasks. After calculating the energy consumption for each task, the formula takes the sum (Σ) of these values over all tasks from j = 1 to the total number of tasks.

Table 3.1. Parameter Used for the load Balancing

Table: Parameter used in this proposed approach	
Parameter	**Variable Description**
PM_i	Physical machine i , $1 \le i \le n$
VM_i	Virtual machine i , $1 \le i \le m$
T_i	Task i , $1 \le i \le n$
ET_i	Execution time
EC_i	Execution cost
L_i	Load
T_m	Number of Tasks
CPU_i	Number of CPU
x_i	Weight value $1 \le i \le 3$

This work develops multi-objective-based load balancing utilizing ADA to address this issue. The natural swarming patterns of dragonflies served as inspiration for DA, a metaheuristic algorithm. For both hunting (a static swarm) and migration (a dynamic swarm), dragonflies gather in large groups. Exploration occurs when the dynamic swarm's many dragonflies travel together over vast distances and unfamiliar terrain. In the static swarm, dragonflies migrate in larger groups and in one direction, with close modifications and sudden shifts in the flying pattern that make sense during the exploitation phase. Here, FA is combined with DA to enhance searching capability and prevent being stuck at a local optimum.

3.4 OPTIMIZATION OF FINDING THE FITNESS FUNCTION

Here, we introduce a bit of randomness into the dragonfly population. A group of answers makes up the population. The solution is developed in response to a set of user tasks and VMs. At first, VM's assignments are completely at random. The solutions are then revised in light of the fitness function. The first answer is shown in. Each dragonfly stands for a set of free nodes, and the length of the solution indicates the total amount of work. If there are ten jobs and five nodes, for instance, the population size would be ten, and each dragonfly's prediction would range from one to five.

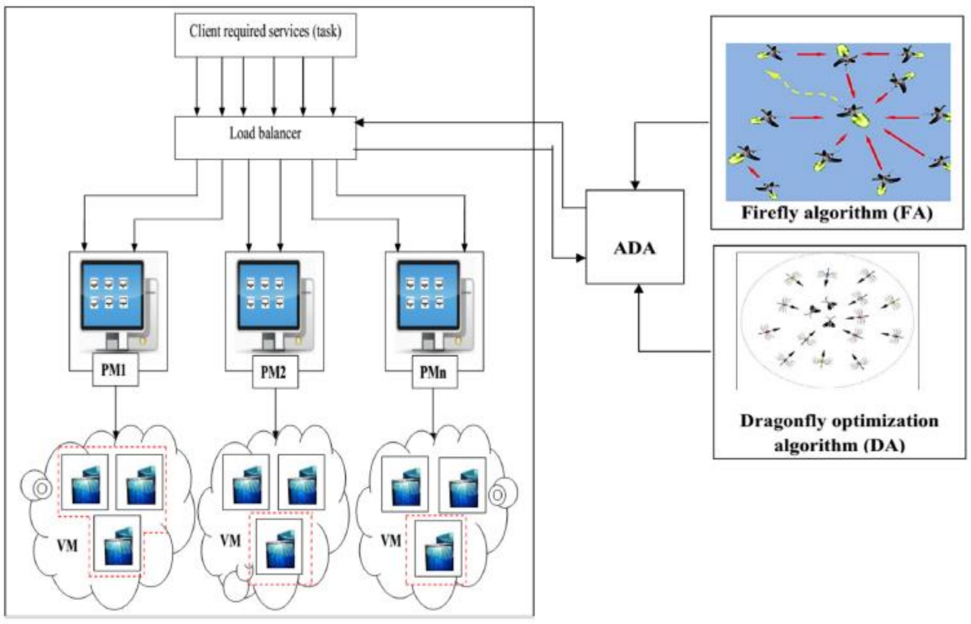

Client required services (task)
Load balancer
PM1
PM2
PMn
VM
VM
VM
ADA
Firefly algorithm (FA)
Dragonfly optimization algorithm (DA)

Figure 3.1. Firefly Optimization techniques for Load balancing

Choose your own female. If the collection is continuously dependent on the significance order minus the probability collection, FA is allowed to slide into a local optimum, making it more difficult to obtain the global optimum. As shown in above figure 3.1 Selecting the female firefly is a discrete process, so the roulette method is used after the selection probability for each female firefly is calculated using formulas to prevent the local optimum. Invention of a new dance formula.

According to equation (1), the attractiveness factor i, j falls as the If the light output of the randomly selected male firefly xj is higher than the light output of the currently active male firefly xi, then the male professional firefly xj will not perform an undertaking procedure in this iterative process, but instead will perform a movement operation consistent with the equation. However, when there is a great deal of space between any two fireflies, the attracting force is quite weak, which can easily result in the moving process being aborted and the best solution not being reached.

3.5 EXPERIMENTAL ANALYSIS

Separation, harmony, cohesiveness, attraction to food, and diversion from the opposing team is the five key parameters that allow this method to update the solution.

$$SP_i = -\sum_{j=1}^{N} X - X_j \qquad Equ \ (3.2)$$

Equation 3.2 defines where X- is the current person's location, Xj is the jth neighbor's position, and N is the total number of people in the immediate area.

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \qquad Equ\ (3.3)$$

Equation 3.3 explains where Vj represent the velocity of jth the neighboring individual

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X \qquad Equ\ (3.4)$$

Equation 3.4 explains where X represents the current location, N the total number of neighbourhoods, and Xj the location of the j-th nearest neighbour. This is how we determine how drawn we are to a particular food source:

$$F_i = X^{+} - X \qquad Equ\ (3.5)$$

Equation 3.5 explains Where X represents the current individual's location and X indicates the food source's location.

While the State distribution function, which was introduced as a new measure of desirability, shares some characteristics with the Gaussian distribution, it differs in that its peak value is lower at the origin and its allocation at the ends persists for a longer period of time. Since the 8745 desirability rapidly drops to almost 0 when r gets too large, this feature makes it easy to generate random numbers over a large region out from the origin. The distinct male firefly can keep the engagement and affiliation of community knowledge near a female, even when the distance between them is considerable, and can obstruction out once have trapped in a local optimum.

Attraction Equation: The attractiveness of a firefly (A) toward another firefly (B) is determined by their objective function values (f A and f B) and the distance between them (r). The equation to calculate the attractiveness is typically defined as follows:

$$\text{A} = \beta * \exp(\gamma * r^\wedge 2) \qquad Equ\ \ (3.6)$$

Equation 3.6 shows where β is the attractiveness scaling factor and γ is the absorption coefficient. The distance between fireflies can be calculated using any appropriate distance metric, such as Euclidean distance.

Movement Equation: The movement of a firefly (A) toward another firefly (B) is determined by the attractiveness (A), the distance (r), and a randomization factor (ε). The movement equation is typically defined as follows:

$$\Delta x = \alpha * (x_0B - x_0A) + \beta * \exp(\gamma * r^\wedge 2) * (x_0B - x_{0\alpha}A) + \varepsilon \qquad Equ \quad (3.7)$$

Equation 3.7 explains, where Δx represents the movement of firefly A, α is the step size or step control parameter, x_A and x_B are the positions of fireflies A and B, and β is a randomization factor to introduce randomness in the movement.

```
Algorithm 1: Improved firefly algorithm through engagement learning method
Initially - Randomly produce Np male fireflies
population {Xi|i = 1, 2, ··· , Np};
Prepare the external archive A = {Xi|i = 1, 2, ··· , Np}, selection possibility p, t=0;
Whereas (t < t_max) do
If p = 1 : Np else
If q = 1 : Np else
if Tb (Xj) < Tb Xi) then
Modernize the location for firefly in male
Comparing the Eq. (3);
Optimizations solution;
else
Use the roulette strategy to calculate the brightness for nominated female firefly
Xk in a external archive;
if Tb (Xk) < Tb (Xj) then
Exchange male firefly Xj concerning female firefly Xk :
Optimizations solution; end
end
end
end
Archiving the female modernization with Archive with A:
Possibility of P with Female firefly corresponding to P;
u = u + 1;
end while
```

The number of occupations is proportional to the length. The tasks 2, 6, 10, and 12 would be completed on machine 1, the tasks 4, 6, 7, and 8 would be completed on computer system 2, and the tasks 1, 3, and 8 would be executed on machine 3. This scenario assumes ten jobs and three computers. Sequencing Work: The second step is to catalogue the sequence in which each machine performs its functions.

Algorithm 2: Pseudocode of the IFACL I terms of UPMSP through setup times for sequences dependents

```
while (p < q_max) do
if i = 1 : Np else
if j = 1 : Np else
for fb (Xj) < fb (Xi) then
substitute the male firefly on the applicable equ3;
find the optimal solution;
else
fix in the other applicable equ.14;
for calculation of roulette strategy brightness for designated female firefly Xk
into external archive;
if fb(Xk) < fb(Xj) then
Exchange male firefly Xj concerning female firefly Xk allowing to Eq. (19);
Update an optimal solution;
end
end
end
end
check for archive A with female;
Check the selection possibility with female firefly P;
u = u + 1;
end.
```

A matrix with the same number of columns and rows as the vector provided by the computer could be used to express the categorization. Therefore, the job sequencing suggested by Seq2 might be represented as a M N matrix showing the collection of processes on to each machine. In the preceding example, the Seq2 matrix, which builds on the Seq1 vector, designates that task 9, 5, 10, and 2 pertain to models 1 and 2, while tasks 4, 7, and 6 pertain to systems 2 and 3, and tasks 3, 1, and 8 pertain to systems 3 and 4.

3.6 PERFORMANCE ANALYSIS

To begin, a preliminary male firefly community (represented by matrix X) of Np male firefly beings was established by the IFACL algorithm's indiscriminate allocation of N purposeful jobs to virtual machine accessible handling machines As shown in above figure 3.2. In the section titled "IFACL is practical to UPMSP without sequence-dependent operation durations," each male firefly cell corresponds to a job sequence for a given machine. In outside archive A, male firefly populations can be used to seed female firefly populations.

Table 3.2: Experimental Parameters

S. No	Parameters	Description values
1	Data centre Region	From 1 to 5
2	Number of Data centre	6
3	Host machine used	50 VM
4	Manager	Time and space
5	Bandwidth	5Gbps
6	Size of the packets	Range 20,000-30,000 Mips

We start by allocating a fly-based initial population at random. The fitness function is then determined using equation (11). Equation 3.7 explains When one firefly (FF)i sees another firefly (FF)j that it finds more appealing (brighter), FF(i)'s flight path is set by

$$x_{i+1} = x_i + B_0 e^{-\gamma r^2}\left(x_j - x_i\right) + \varphi\left(\text{rand} - \frac{1}{2}\right) \quad Equ\ \ (3.7)$$

Attraction accounts for the second component in, while the third term introduces randomization with a 0/0 and "rand" is a random number that is evenly distributed and ranges from 0 to 1. After determining the fitness function, the optimal solution is chosen Table 3.2.

The effectiveness of the IFACL approach to solving the UPMSP by means of sequence-dependent setup times was validated using two benchmark test datasets. We developed the first benchmarking test dataset (named Dataset1) based on the aforementioned publications, and it has been utilized in several related studies. To avoid randomness in test results, we ran each method 18 times as per Fig.3.2 . All other parameters were also taken straight from the study, and the maximum number of iterations was set to Max _ It = 2000, with Np = 75, As shown in above figure 3.1 so that a practical solution could be found in a practical amount of time.

Table 3.3 Experimental Parameter for comparison of different Algorithm

Node VM	**GA**		**Min Max**		**Firefly Algorithm**	
	Mean Value (MV)	**Standard**	**Mean Value (MV)**	**Standard Deviation Value (SDV)**	**Mean Value (MV)**	**Standard Deviation value (SDV)**
20	1892.47	13.761	1893.73	12.16	1891	15.84
40	3922.13	17.39	3985.33	9.3	3936.33	20.06
60	5876.07	33.47	5866.93	26.32	5869.8	28.91
80	7974.93	41.16	7951.67	18.65	7941.87	31.74
100	9915.27	23.09	9854.73	43.85	9848.73	31.46
120	12026.87	37.7	11969.13	35.27	11950.8	24.97

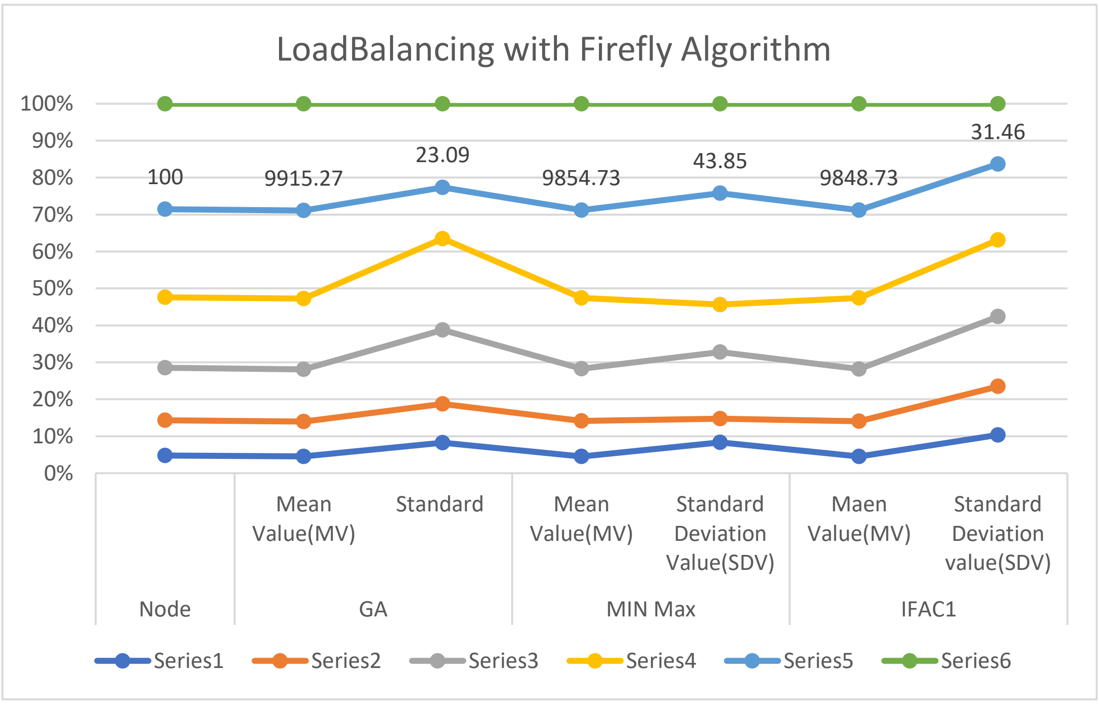


Figure 3.2 Comparison of various Load Balancing with Firefly Algorithm

By repeatedly applying it to Dataset2 and contrasting the results with state-of-the-art studies, the efficacy of the proposed approach is further confirmed. Experiments were recorded in Cloud Sim Plus 7.0.0 software on a Windows platform improved performance, and an enhanced API for simulating cloud computing environments.

3.7 SUMMARY

At the end of the study, this approach can improve the social sharing of information amongst fireflies as well as the computation capacity to break out from the local optimal solution in high-dimensional circumstances. In expressions of accuracy on the dataset, the Firefly approach clearly outperforms GA and MIN -MAX, but it will also be vastly superior to other optimizing algorithms. the Firefly algorithm is presented with the goal of reducing the UPMSP problem's completion time. The technique takes advantage of the gender-neutrality of the Cauchy density function to boost collaborative problem-solving and computational power. This study introduces the Firefly algorithm, which use a reduction in the defined completion time as the step to prepare the target, to solve the UPMSP using a set of images. The firefly algorithm can effectively optimize load balancing in distributed systems, considering both server load and network delay. The proposed model outperforms traditional load balancing algorithms in terms of response time, throughput, task scheduling and CPU utilization. indicates its potential value in optimizing resource allocation and improving system performance. The research contributes to society by enhancing resource utilization and system efficiency, leading to better user experiences and increased productivity. Further research can focus on exploring the algorithm's effectiveness in dynamic load balancing scenarios, evaluating its performance in large-scale environments, and identifying optimal parameter settings. These advancements can pave the way for more effective load balancing solutions in various domains, benefiting society as a whole.

CHAPTER 4

OPTIMIZED DYNAMIC LOAD BALANCING TECHNIQUE IN THE CLOUD BASED ON Q-LEARNING FOR DYNAMIC LOADS

4.1 INTRODUCTION

The delivery of cloud services can be broken down into three primary architectural models, each of which can then be further subdivided into a variety of derivative models. The acronym "SPI Model" is commonly used to refer to the three primary service models. "SPI" stands for "Software, Platform, and Infrastructure," in that order. Because this approach is delivered in the form of a service, it is referred to by the following names: Software-as-a-Service, which is also referred to as SaaS; Platform-as-a-Service, which is also referred to as PaaS; and Infrastructure-as-a-Service, which is also referred to as IaaS. In the IaaS model, the level of abstraction is at its lowest, which enables consumers to exercise a greater degree of control over the infrastructure that they utilize. On the other hand, the level of abstraction is at its highest point for the SaaS model, which is characterized by a lack of direct user influence over the underlying platform and infrastructure[116]. The public cloud offers the least amount of control that an organization can leverage over the different cloud models, while the hybrid cloud model offers the most control. Every day, more of the available cloud resources are being utilized as a direct result of an increase in demand. To store data in a variety of formats, techniques from cloud computing are combined with those from other fields. The complexity of the computing equipment is increased and their workload is increased when they are required to handle both structured and unstructured data forms.

Massive systems in today's world are expected to operate in a manner that is more effective while consuming less power and taking up less space. The design of a modern processor ought to have an emphasis on both power and energy economy. Multicore processors make true multitasking feasible by enabling users to carry out numerous complex activities concurrently. This enables users to do more in a shorter amount of time and get more done overall. Multicore processors, which pack two or maybe more processing cores into a single chip, offer improved performance and novel features that keep systems running at lower temperatures and with greater economy. These advantages are because multicore processors can fit more processor cores onto a single chip.

Cloud computing is a game-changing approach to the delivery and utilization of services based on information technology that is accessed via the Internet. Techniques, methods, strategies, and algorithms for load balancing can each be placed into one of several distinct categories according to the qualities, features, and functionalities that they possess[117].

The following is an in-depth classification of several of the load-balancing approaches that are regularly employed in cloud computing: Static Load Balancing vs. Dynamic Load Balancing Techniques for load balancing can be broken down into two categories: static and dynamic load balancing. The choice as to how the workload should be distributed is made based on a pre-determined set of rules when using static load balancing methods, which means that these approaches are pre-determined. On the other hand, dynamic load balancing modifies the workload allocation such that it takes into account both the present load and the availability of the resources.

Both push-based and pull-based approaches to load balancing may be utilized. In pull-based load balancing, the workload is pulled by the servers based on their current load, whereas in push-based load balancing, the load balancer proactively distributes the task to alternative servers based on their current load. Load balancing methods can either be heuristic or analytical. Heuristic load balancing is more common than analytical load balancing. When using heuristic load balancing, the choice of how to divide the workload is made using a set of rules or heuristics as the basis for making the decision. The choice in analytical load balancing is made using mathematical or statistical models that take into account a variety of criteria, including the size of the task, the number of resources that are being utilized, and the network latency[118].

Cloud computing is not a sophisticated method for providing wanted, the customer required, adaptability approaches a collection of computational assets that are customizable and can be quickly provisioned and unloaded without exhausting considering effort or administration that analyses the unique sequencing of jobs for expert algorithms. Cloud computing is a worldview that delivers needed, adaptable ways to a collection of computational assets that are adjustable and can be rapidly provisioned and released with depleted considered effort or administration. This is needed by the user. Cloud computing is a term that was coined to describe this worldview. In a cloud computing environment, various virtual machines (VMs) share the same physical resources (memory, bandwidth, and CPU) on a single physical host.

A large number of virtual machines (VMs) can share the throughput of a host farm thanks to system virtualization. It can be tough to develop a suitable timetable for task scheduling that takes asset consumption as well as foundation execution into consideration.

This is because the resources of the framework are shared by several applications and consumers. Memory space, the bandwidth of the system, and processing power are all framework boundaries that can affect how efficiently tasks are scheduled. These framework constraints can have an impact in a variety of ways. When it comes to the cloud, the most important goal that task scheduling algorithms strive to achieve is to maintain the same level of load across all of the processors at all times by taking into account the available bandwidth in the system. This is done to enhance the productivity and utilization of the processors, as well as to cut down on the amount of time it takes to finish the task. Moreover, this is done to reduce the amount of time it takes to do the task.

4.2 A DYNAMIC ALLOCATION FOR CLOUD COMPUTING ENVIRONMENT

Virtual Machine and Data Centre Scheduling in the Cloud Task scheduling in cloud virtual machines (VMs) and Data Centre virtual machine (VM) generation scheduling are both well-studied topics. Examined the trade-offs between performance and cost savings by simulating the execution of different workloads on several VM types. This model may then be used to determine which cloud-hosted VM will provide the best balance of price and performance for a given workload. A green distributed data center (DGDC) work scheduling technique with two objectives was suggested. By jointly determining the division of tasks among multiple ISPs and the task service rates of each, they proposed a scheduling method called matching and multi-round allocation (MMA) to optimize the make span and total cost for all submitted tasks subject to security and reliability constraints, thereby maximizing the profit of DGDC providers and minimizing the average task loss possibility of all applications. Unlike cloud VM task scheduling or data center VM provisioning, the big data cluster scheduling problem is addressed in this study.

Since virtualization is crucial to cloud computing, determining where to deploy virtual machines (VMs) is a critical issue. The broker-based architecture and algorithm were created to allocate virtual machines to physical servers.

It was suggested to use a resource management system that would include both resource provisioning and VM deployment. Virtual machine (VM) placement and consolidation methods were investigated, namely those that make use of the min-max and sharing capabilities offered by hypervisors. In, a mechanism for dynamic consolidation using constraint programming was created.

The initial purpose of this method of consolidation was. Made for groups of similar people. However, the presence of heterogeneity, which is typical in a setting with many cloud service providers, was disregarded. In addition, they failed to factor in the unpredictability of future needs and costs. It was suggested to use a dynamic VM placement. The ideal answer may not be guaranteed, however, because the placement is heuristic. Production planning, financial management, and capacity planning are just a few examples of the types of problems that stochastic programming has been designed to address.

To plan for electrical power generation and transmission line expansion while accounting for uncertainties, the authors, for instance, used the stochastic programming approach. Optimal decision-making in a random setting is demonstrated to be a problem that stochastic programming can solve. Stochastic programming has been widely used in other fields, but to the best of our knowledge, it has never been examined specifically in the context of providing computer resources.

4.3 DYNAMIC Q-LEARNING ALGORITHM

Load balancing in the cloud is typically accomplished with the help of the Dynamic Queuing Algorithm (DQA). The most important things that this work has done are: Determine how the present workload and available resources influence DQA's allocation of resources, and use this knowledge to improve cloud performance and utilization.

Efficient task allocation: DQA allocates tasks to resources in a way that makes the most use of both their skills and the resources at their disposal. The flexibility to add or remove resources based on the changing workload is made possible by DQA's flexible scalability, which leads to efficient resource utilization.

DQA can help cut down on cloud computing expenses by increasing productivity with existing infrastructure and decreasing the amount of time and money spent on maintenance and upgrades. Real-time monitoring: DQA continuously monitors the performance and resource utilization, and adjusts the resource allocation in real-time to maintain optimal performance. The purpose of this article is to introduce new readers to Autoscaling strategies and the underlying technology related to cloud platforms. In the field of machine learning, Reinforcement Learning (RL) is used to get notable results in the execution of dynamic decisions. The performance of the model can be fine-tuned by using different regularize and optimization parameters and values. The study discusses previous works and investigates the proof of RL in cloud settings for load balancing and resource allocation. Utilizing resources and services effectively is a difficult undertaking that calls for experience and a flexible algorithm that can adapt to new information and reallocate assets as needed. Performance and optimization of cloud services can be enhanced by using reinforcement learning (RL) strategies, which operate on a trial-and-error basis. Five areas were monitored using an RL method, together with six data centers and 40 Virtual machine hosts.

4.4 ARCHITECTURE

The Q-learning approach uses a reinforcement technique that takes into account the current state to determine the optimum course of action to maximize reward. Its "off-policy" label comes from the fact that it operates arbitrarily and without regard to predetermined policies or norms. This method favors the course of action that solves the problem best and delivers the most benefits.

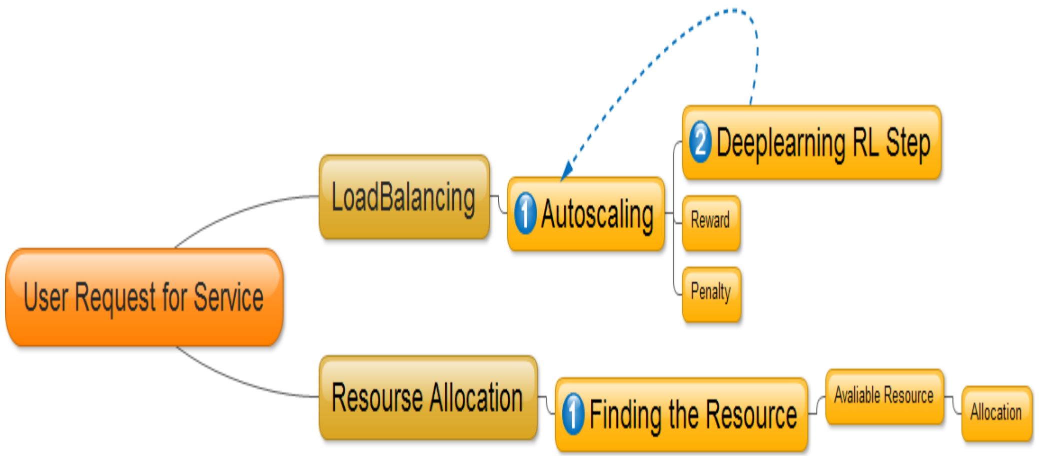


Figure 4.1 Q- learning approach for Autoscaling techniques

Adopting the Q-learning approach in a cloud setting helps the load-balancing activity make better use of the available resources. While handling the users' concurrent requests, we randomly implement priority scheduling and round-robin ideas. As per the above Fig.4,1. Q-Tables are used to introduce the Q-Learning approach to the cloud. The Q-Tables include the necessary states and activities for reaching the goal. The parameter, whose initial value is zero, is incremented with each new decision. It helps the agent make good decisions about what to do next based on the current value of Q.

Additional emphasis was placed on energy efficiency and load distribution parameters, which together now account for the similar method of generalized co-optimal control described mathematically below.

$f(x) = \Sigma w11 f(x1); 0 < 1 \leq$ Equ **(1)**

In the Eq. (1) Here $w$ll signifies weights allotted and $f(x$l) represents individual appropriateness function at 0<l≤n.

For a well-organized solution,

So every VM's load can be used to estimate the total load on the data center.

A virtual machine with the task set P= {a1, a2,…an} with n tasks in the job queue and VM set VM={b1,b2,…bm} with m VMs in a VM pool set, Here based on the processing time as well as the completed task, the impartial parameters can be determined.

Completion time: $CTij$=ΣFti−Sti, Ni=1

Response time: $RTij$=Σ$Subti$−Wti, Ni=1

Throughput : $Thij$=Σ$Succ\ tasks\ Total\ time$,Ni=1

2) Processing time of multiple VM

If network bandwidth is constant then:

SD=$\sqrt[n]{\sum_{n}^{Fxj} x}$ Equ **(2)**

In Eq. (2) n indicates the attributes depending on the global and local abilities of the number of nodes we are connecting and F is the functional value corresponding to x values and y values in summation of various virtual machine task values in resource utilization and execution time.

The task implementation on a VM machine through the energy assessment is determined with resource utilization and execution time.

Energy consumed Hij of i-th task on j-th VM is expressed as

Hij=Y(Uij)×COij ***Equ* (3)**

In Eq. (3) U_{ij} and CO_{ij} represent relative intermediary variations to current and earlier Virtual machines (VMs) where the ith task and jth task will currently maintain in the product of both processes of the element in a virtual machine. Task scheduling problem definitions. (Virtual machines (VMs)) Usually, each virtual machine can be represented as a tuple/row (VM = {id; mips; bw; pes number}),

$Di = \frac{Fmax - Fmin}{Fa}$ **Equ (4)**

Equation (4) defines the Degree of Imbalance (Di), which is a metric used to assess the distribution of load across virtual machines in terms of their performance and execution capabilities. A lower Di value indicates a more stable (balanced) distribution of load. Di is calculated based on the period and is derived using the maximum execution time (Fmax), minimum execution time (Fmin), and average complete execution time (Fa) achieved across all virtual machines.

$T = \sum_{k=0}^{n} \frac{H_{i\ length}}{in\ number \times Vm\ in}$period **Equ (5)**

Where k denotes the number of virtual machines and the n value is within functional value 1; 2; 3-time value denotes the number of tasks/jobs, and the j value is within { 1,2,3,…n}.

Equation (5) introduces the concept of Make Span, which represents the total time required to complete all tasks. In the context of manufacturing, Make Span is the time interval between the start and finish points of a sequence of jobs or tasks. A low Make Span value indicates that the scheduler is providing optimal and efficient task scheduling to the virtual machines, while a high Make Span value indicates that the scheduler is not effectively scheduling tasks to the machines.

$R_u = \frac{max}{1<i<m} \{CT\}$ E***qu* (6)**

Equation (6) defines Resource Utilization (Ru) as a performance metric used to measure the extent to which devices or resources are being utilized. A higher Ru value indicates a greater level of resource utilization, which can lead to increased profits for the cloud provider.

$S_U = \frac{\sum_{i=1}^{m} \boldsymbol{CTi}}{\boldsymbol{Su} \times \boldsymbol{k}}$ *Equ* (7)

Equations (7) and (8) introduce the concepts of Schedule Cost (SC) and Execution Cost, which refer to the expenses incurred by cloud computing users when utilizing devices for task execution in the cloud environment. The primary goal for cloud computing users is to minimize their costs while also maximizing device utilization and minimizing Make Span.

$\boldsymbol{SC = \sum_{n=1}^{b} costi * CTi}$ *Equ* (8)

Algorithm:1 Dynamic Programming Algorithm to Load Utilization Corresponding To Bandwidth And Network Availability

Data centre = ∑ Load, Let VMid = the VM which will start for every data in PT = load in to DC

Capacity in DC

Input VMid, Thres_bottom, Thres_stop, and n.

Calculate the load of the VM.

If the VM load is less than Thres_bottom, allocate the VM to a host with the lowest load.

```
If the VM load is greater than or equal to Thres_bottom and less than Thres_stop, allocate the VM to a host that already has at least one VM running and has the lowest average load.
If the VM load is greater than or equal to Thres_stop, allocate the VM to a host with the lowest number of running VMs.
End if     Loop
For {Get the t hours load predictions of the starting VM}
VM Preload<-Get-Load Prediction (VMid)
{Get load prediction of every VM on host}
HRes<-Get_ResFromLoad (VMs,PreLoads,eachhost)
    End For
For: each server PM in the datacenter
        PM.Tcpu > β
Workload balance in Datacenter ()
End Function
```

The cloud provider can achieve maximum profit by attaining high resource utilization rates.

$$\boldsymbol{ECT_{ab}} = \frac{\boldsymbol{task\text{-}length_i}}{\boldsymbol{mips_l}} \qquad \textbf{Equ (9)}$$

$$S_U = \frac{\sum_{u=1}^{n} \boldsymbol{CT_i}}{\boldsymbol{mspan \times m}} \qquad \textbf{Equ (10)}$$

Equations (9) and (10) define ECTab as the required execution time for a MIPSi task on a virtual machine with a specific task length. By calculating the combined fitness value of each collection using a multi-objective function, the evolutionary strength of the organism to the ecosystem can be determined.

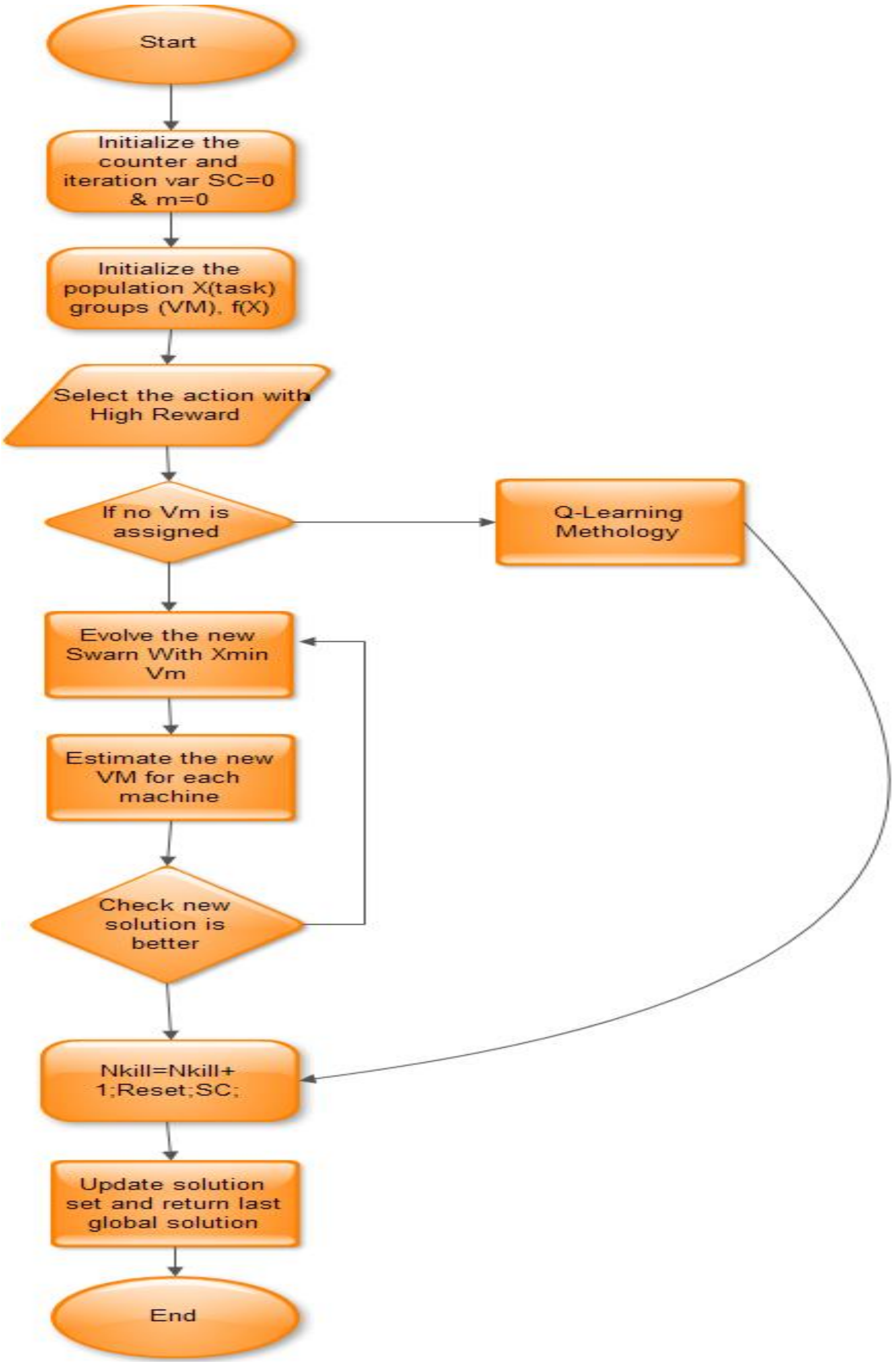

Start
Initialize the counter and iteration var SC=0 & m=0
Initialize the population X(task) groups (VM), f(X)
Select the action with High Reward
If no Vm is assigned
Q-Learning Methology
Evolve the new Swarn With Xmin Vm
Estimate the new VM for each machine
Check new solution is better
Nkill=Nkill+ 1;Reset;SC;
Update solution set and return last global solution
End

Figure 4.2 Flow diagram of Dynamic Q learning

The n=1 Master node Virtual machine is used in this Algorithm's minimum set up, and all machine settings are calculated anew. Because of this, we must verify the upper bounds of Virtual nodes' values as shown in above figure 4.2. Allocating resources for virtual machine startup and migrating VMs for load balancing is the next stage. Our procedure needs to compute this load-balancing factor and choose the right host at this point. The complexity of the host selection approach is O(n), where n is the number of hosts sharing a given set of resources, as was mentioned earlier. Allocating and relocating virtual machines will take an excessive amount of time if there are too many hosts. Failure energy is calculated using equation (13), where (_Time) is the entire time the VM contributed to the energy consumed while the load was kept constant.

$$\varphi_{Time} = \sum_{i=0}^{m} \varphi_{cp} + \varphi_L + \varphi_F \qquad \text{Equ (11)}$$

4.5 EXPERIMENTAL ANALYSIS

Utilizing the VMs' resources (CPU and bandwidth) is seen to increase energy consumption as well. The proposed DQ-Theory seemed to function best with fewer objectives, as indicated by the values. displays the algorithm results for the DQ theory, demonstrating that the DQ theory proposed has superior outcomes. In light of the new requirements (between 200 and 1000 activities), this is essential. Table.4.1. The performance of algorithms for scheduling tasks and balancing workloads has been shown to decline as work volume grows. Because the proposed DQ theory continues to function well even when subjected to greater loads, it is now considered one of the best scheduling algorithms available. Table.4.2.

Table.4.1. Performance Results of Dynamic Q-learning Theory on RIN method with range from 200-1200 tasks

Algorithm	**Load balancing index**					
	200 Tasks in number	400 Task in Number	600 Task in Number	800 Task in number	1000 Task in number	1200 Tasks in number
GA	350	0.061	0.078	0.075	0.7845	0.7845
DSOS	352.1	0.052	0.065	0.062	0.7458	0.7654
MSDE	421	0.051	0.059	0.051	0.6589	0.7124
PSO	450.23	0.48	0.49	0.056	0.6235	0.6784
WOW	520.3	0.49	0.491	0.561	0.6221	0.6641
MSA	550.31	0.461	0.045	0.499	0.5784	0.6612
DQL	572.592	0.451	0.4386	0.495	0.5629	0.6521

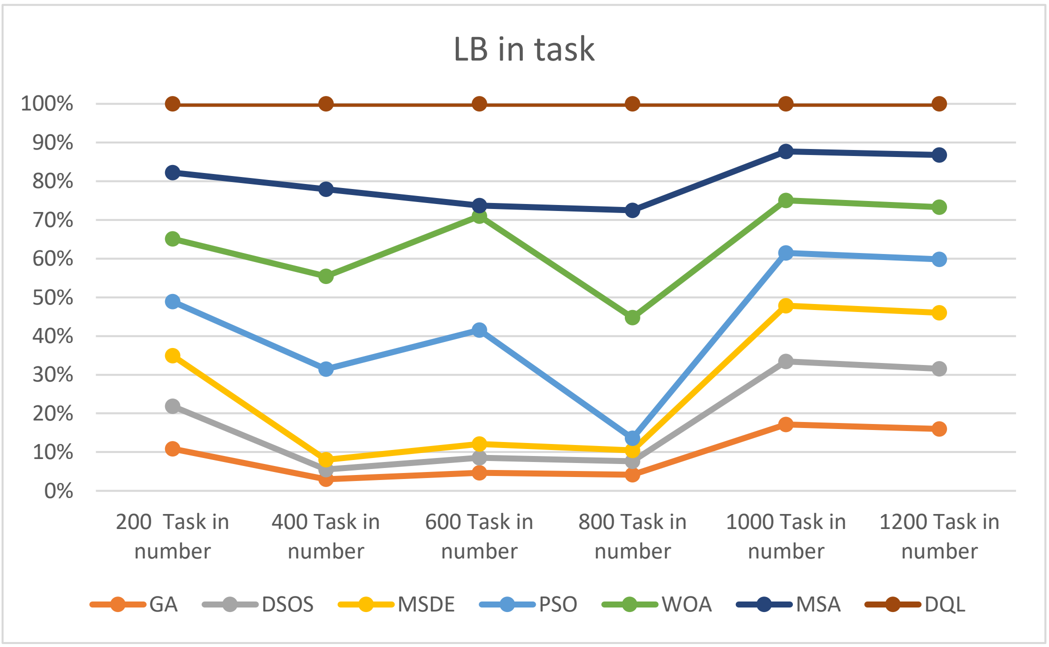


Figure 4.3. Performance Results of Dynamic Q-learning Theory on RIN method with range from 200-1200 tasks

Table 4.2. Performance Results of Dynamic Q-learning Theory on Throughput

Throughput (bps)						
Algorithm	200 Tasks in number	400 Tasks in number	600 Tasks in number	800 Task in number	1000 Tasks in number	1200 Tasks in number
GA	350	0.061	0.078	0.075	0.7845	0.7845
DSOS	352.1	0.052	0.065	0.062	0.7458	0.7654
MSDE	421	0.051	0.059	0.051	0.6589	0.7124
PSO	450.23	0.48	0.49	0.056	0.6235	0.6784
WOW	520.3	0.49	0.491	0.561	0.6221	0.6641
MSA	550.31	0.461	0.045	0.499	0.5784	0.6612
DQL	572.592	0.451	0.4386	0.495	0.5629	0.6521

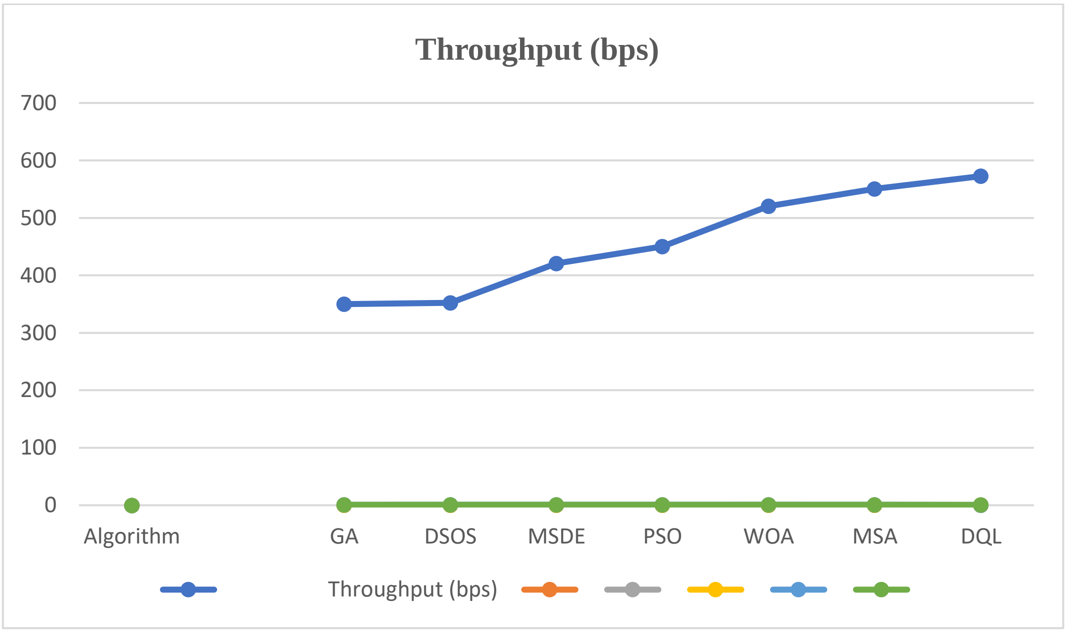


Figure 4.4. Performance Results of Dynamic Q-learning Theory on Throughput

Table 4.3. Performance result on DQ learning on Task completion time

Task completion time (ms)						
Algorithm	200 Tasks in number	400 Tasks in number	600 Tasks in number	800 Task in number	1000 Tasks in number	1200 Tasks in number
GA	1.023	28.36	31.21	38	46.12	49
DSOS	2.021	26.01	33.12	39	45.12	45.5
MSDE	2.031	27.01	30.56	37	46.18	46
PSO	2.452	25.02	32.89	40	45.13	47
WOW	3.021	26.12	31.45	41	44.98	48
MSA	2.089	27.36	30.15	36.5	44.5	46.5
DQL	0.0131	25	29.79	36	44.19	45

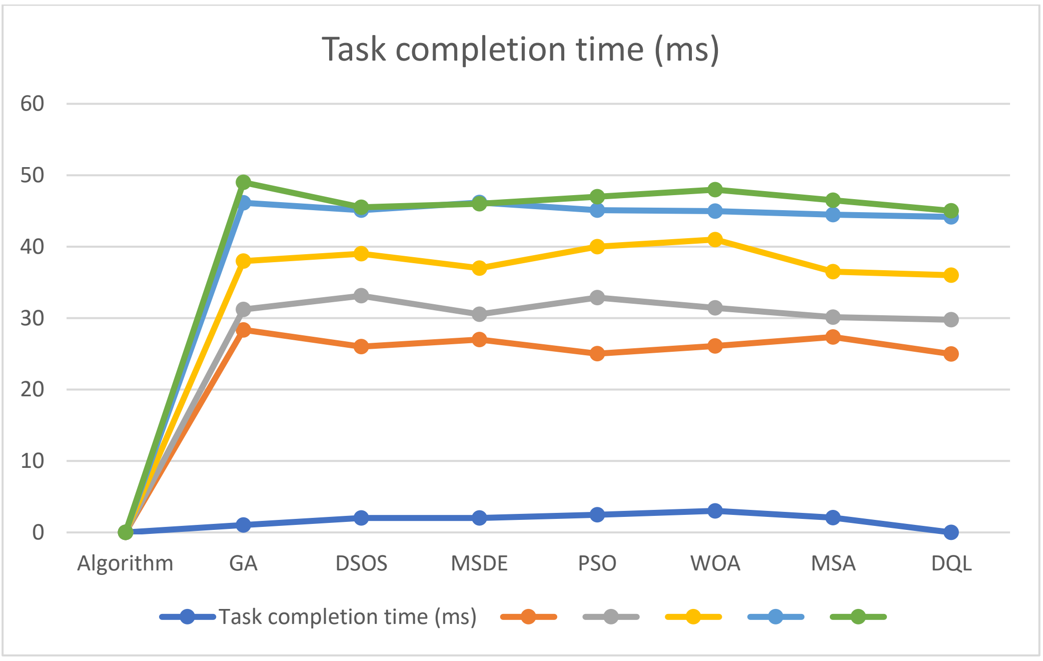


Figure 4.5. Performance result on DQ learning on Task completion time

4.6 RESULT ANALYSIS

When combining various types of virtualization technologies, full virtualization uses more CPU resources than paravirtualization. One reason for this is because comprehensive employs VM machines to simulate infrastructure I/O, which is both more efficient and requires more CPU cycles than virtualization technology's response to an increasing technique for achieving network virtualization. DQL's impact on the Reinforcing learning strategy. The proposed D-Q theory outperformed the other algorithms studied. This enhancement is a direct outcome of the faster convergence of the D-Q theory algorithm, which reduces queueing-related waiting time and resource loss. Comparison of Optimization-Based Task Scheduling Algorithms with D-Q Theory for Throughput Evaluation As shown in the above Figure 4.3,4.4. The offered figure shows how the suggested D-Q theory outperforms competing algorithms in terms of throughput thanks to its load-balanced and energy-aware scheduling.

The D-Q theory can outperform competing approaches because it provides superior global search capability and convergence rate. The suggested system has a CPU utilisation of 35% on average, with peaks reaching 45%, and a utilisation of 15% or less during the night. The above figure 4.5 show that compared to the current system, the suggested system is more stable and has a higher collection rate according to the D-Q theory. The research shows that DQ learning, which makes use of reward and penalty techniques, outperforms conventional application-oriented algorithms when applied to event-based and time-critical tasks. Different statistical metrics and situations with a set number of rounds were used to evaluate the performance of different algorithms including GA, DCOS, MSDE, PSO, WOA, and MSA. All of the examined algorithms were identical in theory and function.

4.7 SUMMARY

To evaluate the effectiveness of the proposed Q-Learning, we gathered metrics and compared them to those from previously used enhancement packet scheduling. The results showed that when compared to the state-of-the-art, Q-Learning-based RL job scheduling improved upon it in every measurable way: energy savings, cost, strength index improvement, task completion time, turnaround time, and total system throughput. More QoS factors can be added in the future to reduce the complexity and overhead of the proposed algorithm. Another option is to improve the suggested Q-Learning-based RL such that jobs can be scheduled while keeping high availability and security in mind. In this paper, we look into the many factors that must be considered to provide quick, dependable cloud services. The suggested model's performance is greatly enhanced by the use of Scheduling algorithms and Q-Learning-based RL techniques, allowing for superior results on highly scalable jobs. The use of such hybrid methods takes cloud performance to a new level and allows for on-the-fly decision-making. The proposed model achieves 20% higher performance than the state-of-the-art. DQL completes indexing the LB values of 15% more documents than the next best algorithm. Throughput has dropped by no more than 20% compared to alternative algorithms. DQL's average task completion time is also very low. The trial findings also indicated that response time increased by no more than 10% for any of the other parameters of the method. In the end, while most algorithms use up to 35% of the CPU, DQ learning only needs 15%. In the future, it will be necessary to incorporate machine learning for dynamic load-balancing algorithms in a larger number of task variables, even though the results of the current study are better than those of the state-of-the-art techniques. Changing the application's load on the fly is more practical for real-time applications. Only averages for bandwidth, CPU usage, and throughput are considered in this work. Network security costs and data transfer costs must also be considered for future development. We wanted to improve the model's performance in the future so that it might function better in a multitasking setting. In this paper, we solely took into account memory and CPU usage when balancing the load. Therefore, we need to make sure that our load-balancing technique takes into account the burden of network and disc I/O.

CHAPTER 5

ACTOR-CRITICAL DEEP REINFORCEMENT LEARNING FOR OPTIMAL LOAD BALANCING IN CLOUD DATA CENTRES

5.1 INTRODUCTION

Many service providers now do the vast majority of their processing in the cloud, thanks to the proliferation of sophisticated data centers and networks. Managing multiple virtual resources in a cloud setting to optimize resource allocation is a challenging topic due to constraints on available network bandwidth and hardware or virtual resources. OpenStack's Heat service is one example of a default auto-scaling and orchestration method available in the cloud, however it can only adapt to changes in one parameter, such CPU utilization, and is therefore not suitable for dynamic networks. A crucial orchestration feature, auto-scaling automatically modifies the number of servers in response to the current network load. This method allows service providers to quickly reboot unused servers and free up valuable bandwidth when network congestion hits. In order to facilitate the realization of data ownership and resource management, many cloud infrastructures provide simple auto-scaling and orchestration techniques by default[119]. Auto-scaling can be provided by the heat orchestration service in OpenStack, for instance, by establishing a threshold for a specific resource metric, such CPU utilization. There may be a lag in the response time of a single resource (particularly CPU utilization) based heat orchestration service due to the fact that servers typically require some time to react to network track. changes. When RL is combined with deep neural networks, more nuanced policies can be learned. In cloud computing, it has been used for load balancing to more efficiently assign virtual machines to underlying real servers. In order to learn a policy and assess its worth, Actor-Critical Methods combine value-based methods (like Q-learning) with policy-based methods (like policy gradient methods). Load balancing in the cloud makes advantage of this to distribute virtual machines across available servers efficiently, cutting down on wasted resources and power usage. The jobs are prioritized according to their features (such as the resources they need and the projected length) and scheduled in the order of importance. Appointments of higher importance are planned first, followed by those of lesser importance. However, there may be circumstances where this straightforward method excels. Programming using Natural Selection, or Genetic Algorithms In this method, a genetic algorithm is used to develop a schedule that minimizes the cost of some metric of interest (such as resource utilization or work wait times) As shown in the below Figure 5.1 in order to maximize productivity. A population of schedules is generated by the genetic algorithm, and their fitness is measured against the goal function before the most successful schedules are bred to form the next generation. A considerable number of iterations may be necessary to converge to an ideal solution, however this method can be useful for optimizing difficult scheduling problems.

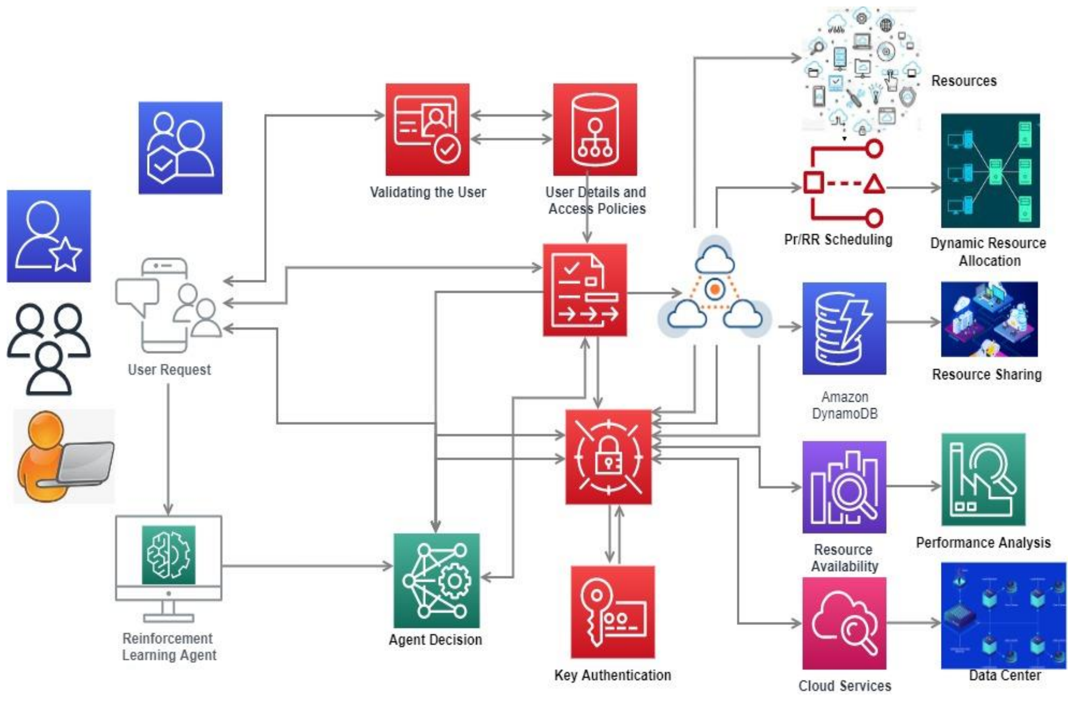


Figure 5.1. Reinforcement Learning model for -Actor critic Method

Optimisation strategies inspired by ant colonies use an algorithm to determine the most effective daily routine. For this algorithm to operate, it must first generate a population of "ants" that probe the search space and then use pheromones to lead other "ants" to the most promising solutions they discover. The algorithm converges to a solution with a high concentration of pheromones since the pheromones dissipate over time.

Using the actor-critic framework, we model the interaction between the job scheduler and the cloud data centre as a Markov Model Process (MMP). To reduce computation expenses when time and resources are limited, we employ this method in a cloud resource allocation scenario.

An approach to scheduling tasks is developed by combining Deep Q-learning with target networks and involvement rerun. They are investigating potential algorithms to lessen offloading's drain on processing power.

a) To analyses and evaluate the potential applications of various ML-based load-balancing technologies in data centers.

b) A Compute-Intensive Workload Allocation Scheme based on the Actor-Critic framework, with a number of parameters optimized for use in the cloud.

c) Increasing emphasis is placed on the challenges and potential future research avenues that stem from this study. Users are able to conserve network resources while still receiving excellent service thanks to auto-scaling.

5.2 ENHANCED RESOURCE ALLOCATION AND WORKLOAD MANAGEMENT USING REINFORCEMENT LEARNING METHOD FOR CLOUD ENVIRONMENT

However, the typical actor-critic's use of a linear functional form to estimate the action-value function results in both excessive variance and an incorrect policy gradient. To solve this problem, an advantageous actor-critic was created. The actor makes arbitrary choices about what to do, the critic gives him feedback in the form of scores, and the actor modifies the likelihood that he will pick a given action based on the ratings. On the flip side, the advantage functions can be a huge help in bringing the policy gradient variance down. Modelling radio access network offloading as a Markov Decision Process leads to function approximation error when the neural network attempts to estimate the value function or policy function in an actor-critic approach but fails to do so accurately.

We developed a double-deep Q-network based reinforcement learning strategy. To prevent state-space explosion and identify the optimal operational loading method in real time, they went to a sophisticated neural network-based amount of extra. The authors suggest a deep learning method using the State-Action-Reward-State-Action (SARSA) paradigm to improve the effectiveness of computational dumping. In simulation studies, the authors (GA) compare and contrast the proposed system with a number of scheduling methods, including the first-come, first-served (FCFS) algorithm and a genetic algorithm. In comparison to alternative algorithms, the suggested method makes better use of available resources and has a shorter makes pan (total execution time)[120]. In high-dimensional or continuous action spaces, it is suggested that function approximation error might lead to instability and poor performance in actor-critic approaches.

To solve this problem, they offer the Twin Delayed Deep Deterministic (TD3) policy gradient technique, which makes three key adjustments to boost the approximation accuracy of the function. The actor in an actor-critic deep reinforcement learning algorithm makes recommendations for how resources should be allocated, while the critic provides feedback on how effective those recommendations are. Both the actor and the critic are neural networks that undergo a hybrid supervised/reinforcement learning training process. The technique is tested in a simulated cloud data center. In terms of resource utilization and job completion time, the results suggest that it is superior to more conventional resource allocation methods. The research finds that the proposed method has the potential to increase cloud data centers' efficiency and flexibility in allocating resources.

5.3 ACTOR CRITIC MODEL FOR INTENSIVE WORKLOAD ALLOCATION

The suggested method uses deep reinforcement learning to discover a fair and efficient method of allocating available resources to tasks. The two halves of an actor-critic deep RL system are the actor, which suggests resource allocation decisions, and the critic, which analyses the quality of those suggestions. Fig.2. Using a mix of supervised and reinforcement learning, neural networks play the roles of both the actor and the reviewer.

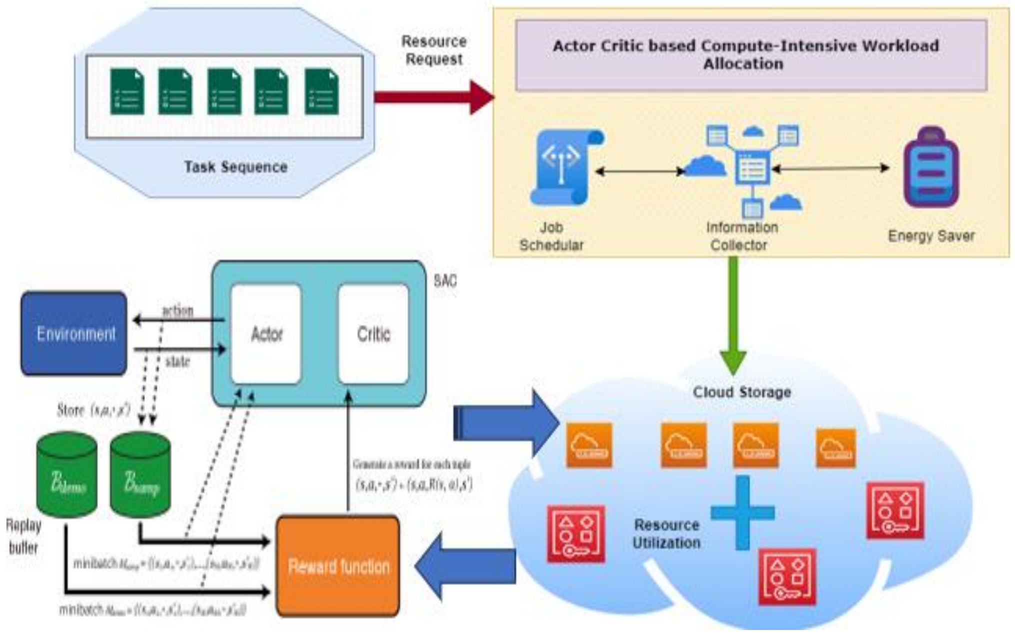


Figure 5.2 Resource utilization of Actor critic-based computing

An artificial cloud data centre is used to test the algorithm. In terms of resource utilization and job completion time, the results reveal that it is superior than conventional resource allocation methods As shown in the above figure 5.2. The maximum discounted Q-value of the next state (f_t+1) and all actions (l_t+1) that can be taken from it constitutes the optimal Q-value for a state-action pair (f_t, l_t), where the factor of discount () defines the weight given to future rewards:

$$Q*(f\_t,\ l\_t) = \sum [Rx\_t+1\ +\ \beta \max\ 〖\_(l\_t)〗\ \ +1\ Q*(f\_t+1,\ l\_t+1)\ |\ f\_t,\ l\_t](1)\ \text{Equ 1}$$

The best Q-value in RL is described by the Bellman equation. It allows you to get the best Q-value for a state and action by recursively calculating it from the best Q-values of the subsequent state and all possible actions that can be taken from it.

An explanation of the equation's parts follows:

As seen in Eq.1, The expected total reward that can be gained by following the optimal policy from the current state and action (f_t, l_t) onwards is represented by the optimal Q-value Q*(f_t, l_t).In state f_t, the immediate payoff for doing l_t is Rx_t+1. This is the payoff from the next clock cycle (t+1). The discount factor, denoted by, influences how much emphasis is placed on future rewards in comparison to immediate rewards. It's a number between zero and one that takes into account not only the potential delay and degree of uncertainty in getting future incentives, but also the rewards themselves. max_l_t+1 The greatest Q-value for the next state f_t+1 and all actions from it is denoted by Q*(f_t+1, l_t+1). Total estimated benefit of continuing with the optimal policy from the current state and action forward

5.4 ARCHITECTURE MODEL FOR ACTOR AND CRITIC

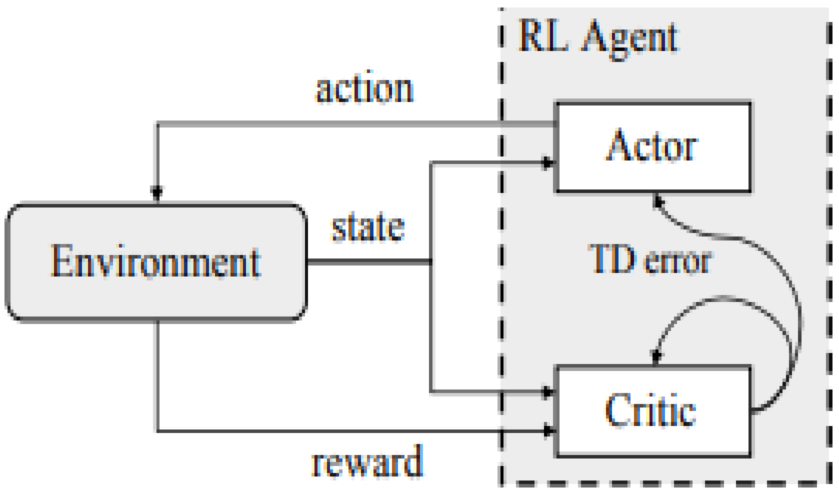


Figure 5.3 Architecture on action and reward agent policies

The maximum Q-value, representing the optimal next-state action given the current policy, is chosen by the max operator. In conclusion, the ideal Q-value is expressed in terms of the immediate reward, the largest expected real bonus from the next state, and all potential actions by solving the Bellman equation. This equation plays a crucial role in many Q-learning algorithms, which are used to refine an agent's policy over time by updating its Q-values. As shown in the below figure 5.3. It then presents self-adaptive systems, which may keep tabs on their own actions and modify their approach to get the desired results.

ALGORITHM 1: CALCULATION OF GRADIENT ACTOR-CRITIC RL ALGORITHM

```
**Initialize** actor network Va θ (s) and critic network Dπθ (o, p) with weights
Initialize actors and critics learning rate γn and γm, and TD error discount
for each training epoch n = 1, 2, ..., N do
    Receive initial state s1, where s1 = environmental. observe()
    for each episode l = 1, 2, ..., L do
Select action according to Su, where
        Action (at) = actor. choose action(Su)
Execute action at, receive reward Rt and next.
    state st+1, where rt, st+1 = environmental. step(at)
    Calculate TD error in critic, where β
    Dπθ = r + βV Dπθ (st+1) − V πθ (st)
    Calculate policy slope in actor using advantage function, where
    J(θ) = Kπθ- logπθ(st, a)δ-πθ
    update state st = st+1
        End
            End
```

A neural network is used to represent the actor's policy and the critic's value function in the algorithm.

The algorithm consists of the following procedures:

Random weights and biases are used to initiate the actor network, (s), and the critic network, Q(s, a). Set the TD error discount factor,, and the actor and critic learning rates, a and c, respectively. The following procedures should be carried out for each training epoch: Observe your surroundings to obtain the starting state (s1). Follow these instructions for each episode: Choose an operation using the actor policy, at, where st is the current state. Carry out the action in the environment; you will be rewarded with rt and advanced to state st+1. Then, utilizing the current, action, compensation, and next state, determine the TD error in the critic,. The advantage function, defined as the TD error multiplied by the increase in the logarithm of the likelihood of the current action in the current state, is then used to derive the actor's policy gradient, J(). Last but not least, change st to st+1, the next possible state. It marked the end of the training period. Using neural networks as approximations of the procedure and the value function, this technique is meant to train an actor-critic RL model that can learn a policy that maximizes cumulative rewards over a sequence of states and acts in an environment as shown in the below Figure 5.4 . To strike a fair balance between short-term and long-term gains, the TD error discount factor is applied.

The advantage function is used to adjust the policy gradient based on how well the policy is performing relative to the predicted value. In reinforcement learning, actor-critic algorithms are used to optimize (the actor) and estimate (the critic) the value of an approach at the same time.

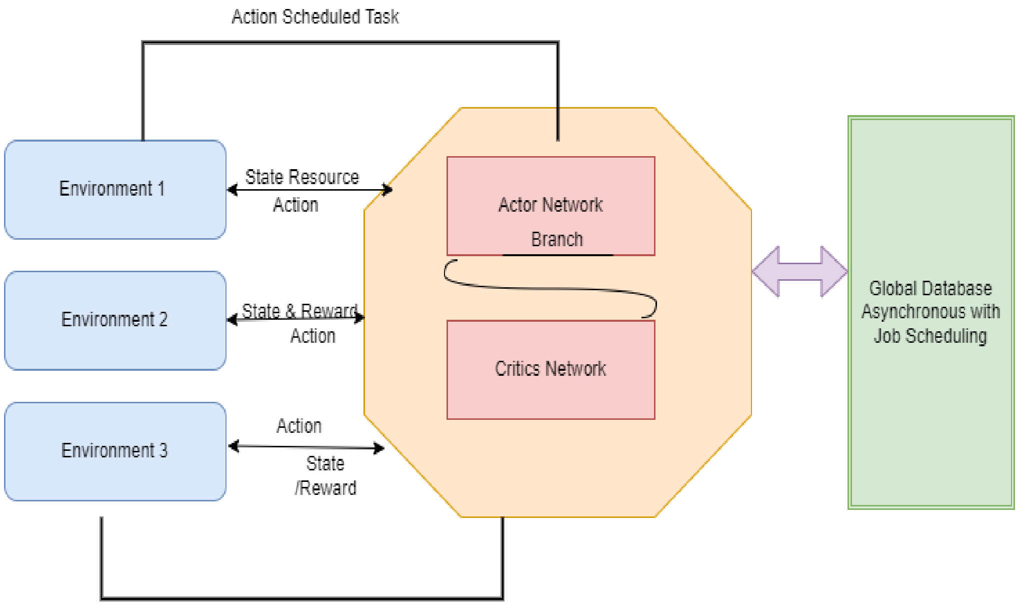


Figure 5.4 Actor and critics network for state and reward

Policy updates equation:

$$\forall\varphi = \omega\sigma\forall\_\varphi_{log\pi}(l_{\frac{k}{g}}\ k,\varphi \qquad \text{Equ (2)}$$

This equation 2. updates the actor's policy weights (φ) based on the advantage estimate (σ) and the log probability of selecting the action l_t in state k_t according to the policy π.

Value function update equation:

∀Z (l_t) = β(δ_t - V(l_t)) Equ (3)

This equation 3. updates the critic's value function (Z) for a state l_t based on the advantage estimate (δ_t) and a learning rate (α). Total reward equation .4:

$R_x = \sum_b^a r^i r_i$ Equ (4)

This equation computes the total discounted reward obtained over a sequence of time steps, where r_i is the immediate reward obtained at stage γ, and I am the discount factor.

Probability of acceptance:

$\mathrm{P}rob_x = \left(\frac{1}{(1+\exp(-f))}\right)$ Equ (5)

This equation 5. computes the probability of accepting a task assignment by a VM instance, where x is a weighted sum of features that capture the suitability of the example for the task.

Resource utilization equation:6

$Z = \sum_{b-1}^a a/(m * Z\_max)$ Equ (6)

To calculate the resource utilization of a cloud system, we can use the following equation, where a_i represents the utilization of resource I, b represents the total number of resources, and Z_max represents the maximum utilization. Actor-critic algorithms, which are used for load balancing of computationally heavy workloads in the cloud, can use a wide variety of formulas, some of which are shown here as examples. Parameters and equations can change based on the specifics of the task and the algorithm being used.

5.5 EXPERIMENTAL ANALYSIS

To test how well the proposed technique works in practice, we utilize Cloud sim, a Java-based cloud simulation tool, to create and configure a cloud environment. The recommended deep RL method is developed in Python and applied in a Cloud sim simulated cloud. We used Google's task events dataset as a reference while designing our user tasks. Our simulations were run on four Virtual Machines, each with 16 GB of RAM and 1 TB of storage. It can take anything from 5 GB to 100 GB of storage space for a virtual machine to complete a task. Using the Google Task Events dataset, we develop nine user tasks. To begin learning from its environment, reinforcement learning assigns its initial batch of charges at random and then evaluates the performance of each virtual machine.

The simulation results obtained after applying random scheduling are used to calculate the average computing efficiency of each virtual machine in terms of Megabytes per second as shown in the below Figure 5.5. Four virtual machines were used, and the prizes we could give them were split into four groups. If the calculation is run on the fastest available virtual machine, the bonus increases to +2. When the next-best virtual machine is used for the assignment, the action obtains a plus one. If the user's request is given to the VM with the lowest efficiency rating, the operation will be penalized by -2. If the action uses the VM with the third-fastest processing speed, a penalty of -1 is applied Table 5.1. The suggested deep reinforcement learning-based scheduling incentivizes the user's action of assigning a job to a virtual machine. So, VMs are rewarded for their efforts. Since the scheduling procedure's incentives and punishments are tied to the choice of virtual machine rather than the task itself, we call it "non-preemptive."

Table 5.1 Values of performance parameters assessed with a typical workload at regularly scheduled times

Algorithm	Response Time	CPU Utilization	Throughput	Task Completion Time (ms)
GA	350.0	0.061	0.078	0.075
DSOS	352.1	0.052	0.065	0.062
MSDE	421.0	0.051	0.059	0.051
PSO	450.23	0.480	0.490	0.056
WOW	520.3	0.490	0.491	0.561
DQL	572.592	0.451	0.4386	0.495
ACD-RL	510	0.445	0.4275	0.470

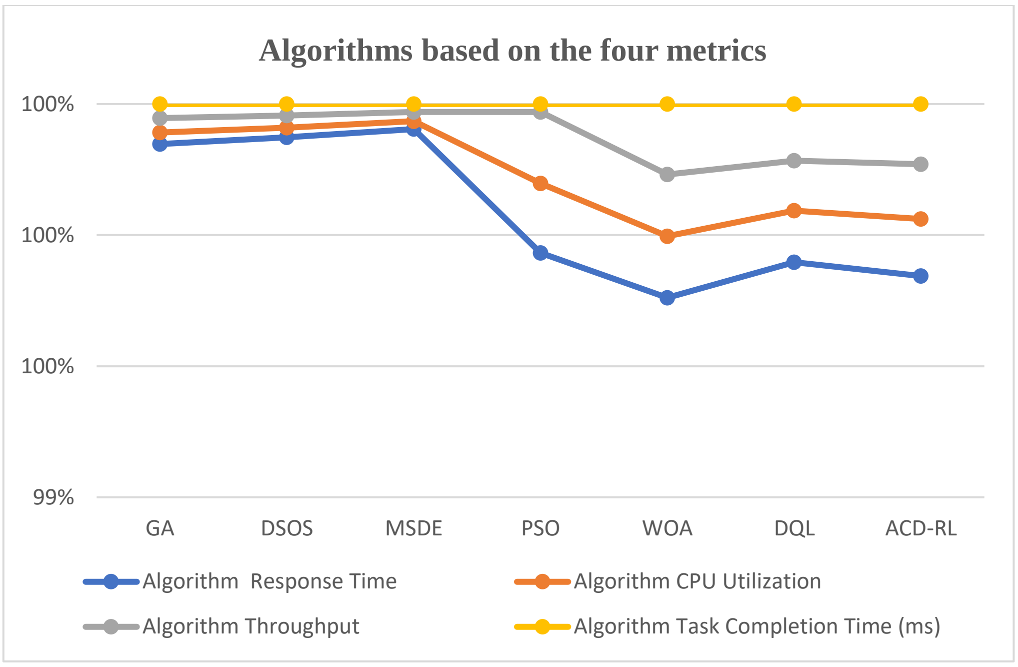


Figure 5.5 Algorithm based on Four metrics on ACD_RL

5.6 SIMULATION RESULTS

This table provides a comparison of the efficiency of various methods for performing certain tasks associated with resource allocation in a cloud computing setting. Response time, CPU utilization, throughput, and job completion time (in milliseconds) are the four metrics included in the table. GA, DSOS, MSDE, PSO, WOA, DQL, and ACD-RL are the algorithms being compared. According to the data in the table, GA has the slowest reaction time, at 350.0 ms, while DSOS has the fastest. The greatest throughput figures for PSO and DQL are also 0.49 and 0.4386, respectively, whereas WOA has the highest CPU usage at 0.490. The majority of algorithms take only a few milliseconds to do a task, with MSDE taking the cake at 0.051 ms. With a reaction time of 510 ms, CPU utilization of 0.445, throughput of 0.4275, and job completion time of 0.470 ms, ACD-RL fares relatively well in comparison to the other algorithms.

5.7 RESULT ANALYSIS

By efficiently allocating resources and distributing workloads over multiple servers, the proposed approach seeks to reduce the total execution time of operations. The proposed model will be enhanced by the incorporation of a priority order shortly. Given the relative weight of the tasks, it is required to reevaluate the state space, action space, and reward function. By analyzing the consequences of planning high-priority occupations on the rewards they receive, algorithms can be taught to learn scheduling policies that take the order of the tasks into account. The suggested ACL-RL agent outperforms the other six algorithms significantly. The ACL-RL agent's overall system cost is closer to both the D-Queuing method and the PSO algorithm for the four components we examined.

This occurs because processing time for each type of data is proportional. Since the search spaces of DSOS and MSDE are quite large as the amount of data points grows, they consume a lot of CPU and significantly degrade the performance of other algorithms. Compared to throughput and job completion, performance improves slightly with higher quantized levels, but not as much as with the suggested ACD-RL. The reason for this is that the noise introduced by the quantization process makes it difficult for the brain to properly process behavioral outcomes.

5.8 SUMMARY

The ACD-Rl optimizes the next move. The results of the suggested model improve by 23% in comparison to those of the earlier investigations. DQL completes 20% more quickly than another algorithm while indexing LB values. All the different algorithm values used in this experiment revealed that the reaction time could go up by no more than 10%, with throughput needing to drop by no more than 23% compared to other algorithms and the Task Completion Time of ACD-RI being concise on average. Finally, while other algorithms may utilize up to 38% of the CPU, ACD-RI learning only needs 12%. To facilitate future collaboration on computational tasks, we will investigate the feasibility of establishing an edge cloud computing network architecture. We will also examine ways in which computation and communication can be simplified during the training phase. We will investigate federated learning-based RL, which does not require local training data but rather a constant stream of new data entering a central data hub in order to distribute model parameters.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

While there are many benefits to using cloud computing, three of the most essential ones are cost savings, increased flexibility, and increased convenience. Despite this, it frees you from bothersome complexity like allocating resources, scheduling tasks, maximizing CPU use, and increasing throughput. Several strategies were developed to address these difficulties. In cloud computing, RA refers to the process of allocating available resources to necessary cloud applications across an online network. Poor allocation occurs in a cloud setting if resources were not allocated effectively. An optimal source allocation based on a multi-agent system is necessary to address such a problem. In order to effectively allocate diverse activities from cloud customers to the available VM, the Actor Critic Reinforcement Learning Model was suggested. The cloud equipment was managed by the suggested multi-agent system. A Dynamic Allocation was used to reframe the user's inquiry as a Machine Learning challenge rather than a task problem requiring various resources.

The use of the Firefly system's Metaheuristics algorithm in reliability analysis has shown improved results, especially in scenarios with a larger number of nodes and request counts. The Metaheuristics algorithm of the Firefly technique produces improved results as the number of nodes increases. For 50 nodes, the obtained reliability analysis is 79%, whereas the GA obtains a low reliability of approximately 64%. The Firefly system's Metaheuristics algorithm produces improved results as the number of nodes increases. For 2000 request counts, the number of failed requests is 380, whereas GA and Min-Max attain a low reliability of approximately 310. The completed requests analysis demonstrates that the Metaheuristics algorithm of the Firefly system achieves superior performance over GA, Min-Max as the number of requests increases.

After that, we have the real-time, optimal policy-learning Dynamic Q-Learning approach. RL methods optimize cloud services by using a trial-and-error policy to boost performance. In Dynamic Q-learning optimizes based on a model with several subgroups. The suggested Q-learning algorithm generates an ideal schedule via the mechanism of Reward and penalty condition, demonstrating autoscaling techniques for the allocation of newly created Virtual Machines. Which minimizes the make span and achieves efficient workload distribution across VMs according to their processing capabilities.

Getting an ideal timetable, the first time around reduces the need to reschedule tasks. The best possible schedule is designed with the help of the Dynamic Q-learning algorithm. Finally, proposed work introduces the use of deep reinforcement learning for optimal resource allocation in a cloud data center environment. Same in Reinforcement learning the previous Dynamic reward function is compared with another Algorithm Actor Critic leaning algorithm, ACD-RL finds the best course of action and achieves 23% better results compared to earlier studies. The response time could go up by no more than 10% for all different algorithm values used in the experiment. ACD-RI learning brings down the CPU utilization to 12% compared to other algorithms.

The compared algorithms include GA, DSOS, MSDE, PSO, WOA, DQL, and ACD-RL. Based on the four metrics (Task Completion Time, Throughput, CPU Utilization, and Response Time) results for GA and DSOS have the lowest response times, WOA has the highest CPU utilization, and PSO and DQL have the highest throughput values .Finally ACTOR -CRITIC Reinforcement Learning model make the better results on the basis of Observation is conducted with existing methodology and obtained results proved that the proposed mechanism delivers efficient scheduling results in the aspect of communication cost, bandwidth utilization, and overall execution time.

6.2 LIMITATIONS

In cloud settings, researchers are actively exploring various strategies to harness the benefits of reinforcement learning while addressing its limitations. Strategies are being studied to take advantage of RL's benefits while overcoming its limitations in cloud settings. These include model-based RL techniques, hybrid approaches that integrate RL with classic optimization methods, and applying RL in particular controlled elements of cloud administration.

Despite its use of successive agent's actions to generate gradient estimators and alter the agent's policy, the computational demands of these methods make them impractical for certain applications. Therefore, alternative approaches should be considered for real-time adaptive control tasks. One such approach is the evolutionary reinforcement learning algorithm, which utilizes an adaptive uncertainty handling technique to adjust the number of roll-outs per policy evaluation. This allows for a better ranking of candidate policies and ultimately leads to improved solutions.

6.3 FUTURE WORK

The future direction of the proposed scheme improves Load balancing algorithms can be enhanced to take into account the quality of service (QoS) constraints such as latency requirements, security considerations, and service-level agreements (SLAs). By incorporating QoS constraints, load-balancing decisions can be optimized to meet specific performance targets. By considering QoS constraints in load balancing algorithms, the performance targets of latency requirements, security considerations, and service-level agreements can be adequately addressed. This means that load-balancing algorithms can take into account factors such as the time it takes for a request to be processed, the level of security required for a particular task, and any specific agreements or guarantees that need to be met in terms of service levels. By incorporating these QoS constraints, load-balancing decisions can be optimized to ensure that the performance targets are met.

REFERENCE

[1] D. Tayouri, S. Hassidim, A. Smirnov, and A. Shabtai, "White Paper-Cybersecurity in Agile Cloud Computing–Cybersecurity Guidelines for Cloud Access," *Cybersecurity Agil. Cloud Comput. Guidel. Cloud Access*, pp. 1–36, 2022.

[2] S. Ahmad, S. Mehfuz, F. Mebarek-Oudina, and J. Beg, "RSM analysis based cloud access security broker: a systematic literature review," *Cluster Comput.*, vol. 25, no. 5, pp. 3733–3763, 2022.

[3] A. K. Alnaim and E. B. Fernandez, "The State of Security in Sdn, Nfv, and 5g," *Nfv, 5g*.

[4] K. Kishor, "Cloud Computing in Blockchain," in *Cloud-based Intelligent Informative Engineering for Society 5.0*, Chapman and Hall/CRC, 2023, pp. 79–105.

[5] S. Khatri, A. K. Cherukuri, and F. Kamalov, "Global Pandemics Influence on Cyber Security and Cyber Crimes," *arXiv Prepr. arXiv2302.12462*, 2023.

[6] G. Qiang, S. Tang, J. Hao, L. Di Sarno, G. Wu, and S. Ren, "Building automation systems for energy and comfort management in green buildings: A critical review and future directions," *Renew. Sustain. Energy Rev.*, vol. 179, p. 113301, 2023.

[7] E. Conchon, "Cyber Security Strategies While Safeguarding Information Systems in Public/Private Sectors," in *Electronic Governance with Emerging Technologies: First International Conference, EGETC 2022, Tampico, Mexico, September 12–14, 2022, Revised Selected Papers*, 2023, p. 49.

[8] M. Repetto, "Adaptive monitoring, detection, and response for agile digital service chains," *Comput. Secur.*, p. 103343, 2023.

[9] M. Z. Hasan, M. Z. Hussain, Z. Mubarak, A. A. Siddiqui, A. M. Qureshi, and I. Ismail, "Data security and Integrity in Cloud Computing," in *2023 International Conference for Advancement in Technology (ICONAT)*, 2023, pp. 1–5.

[10] N. Athirah, N. A. Syahadah, N. Naffisah, N. S. Sapiai, and N. Awang, "Cloud

Security: The Use and Significance in Information Security Management," in *Proceedings of 1st Glocal Symposium on Information and Social Sciences (GSISS) 2023*, 2023, p. 28.

[11] T. Wang, H. Huang, T. Tian, and Z. Zhou, "A Novel Elasticsearch Encryption Scheme for Intelligent Transportation System Applications," in *Proceedings of the 2023 4th International Conference on Computing, Networks and Internet of Things*, 2023, pp. 531–535.

[12] S. Balachandar and R. Chinnaiyan, "Intelligent Broker Design for IoT Using a Multi-Cloud Environment," in *Convergence of Deep Learning and Internet of Things: Computing and Technology*, IGI Global, 2023, pp. 23–41.

[13] M. A. Salitin and A. H. Zolait, "Evaluation criteria for network security solutions based on behaviour analytics," *Int. J. Syst. Control Commun.*, vol. 14, no. 2, pp. 132–147, 2023.

[14] B. Collier, "Considerations for Selecting and Implementing Cloud Security Solutions Using Cloud Access Security Brokers," Marymount University, 2023.

[15] A. R. Reddy, K. L. Flower, J. Anitha, and A. K. Kandru, "Detecting and Preventing Unauthorized User Access to Cloud Services by CASBs," in *2023 Second International Conference on Electronics and Renewable Systems (ICEARS)*, 2023, pp. 868–873.

[16] K. Ramesha, "Adaptive Cloud Access Security Broker," Dublin, National College of Ireland, 2023.

[17] A. Abbas, "Cloud Access Security Brokers (casbs): Enhancing Cloud Security Posture," 2023.

[18] A. Boru \.Ipek, "Multi-Objective Simulation Optimization Integrated With Analytic Hierarchy Process and Technique for Order Preference by Similarity to Ideal Solution for Pollution Routing Problem," *Transp. Res. Rec.*, vol. 2677, no. 1, pp. 1658–1674, 2023.

[19] Q. Chen, X. Ma, Y. Yu, Y. Sun, and Z. Zhu, "Multi-objective evolutionary multi-tasking algorithm using cross-dimensional and prediction-based knowledge transfer," *Inf. Sci. (Ny).*, vol. 586, pp. 540–562, 2022.

[20] Q. Wei, J. Yang, Z. Hu, H. Sun, and L. Wei, "A multi-objective multi-tasking evolutionary algorithm based inverse mapping and adaptive transformation strategy: IM-MFEA," *ISA Trans.*, vol. 135, pp. 173–187, 2023.

[21] R. Liu, P. Yang, H. Lv, and W. Li, "Multi-objective multi-factorial evolutionary algorithm for container placement," *IEEE Trans. Cloud Comput.*, 2021.

[22] D. Petcu, "Service deployment challenges in cloud-to-edge continuum," *Scalable Comput. Pract. Exp.*, vol. 22, no. 3, pp. 313–320, 2021.

[23] A. Hilali, H. Hafiddi, and Z. El Akkaoui, "Microservices Adaptation using Machine Learning: A Systematic Mapping Study.," *ICSOFT*, pp. 521–532, 2021.

[24] Y. Li *et al.*, "Sim-DRS: a similarity-based dynamic resource scheduling algorithm for microservice-based web systems," *PeerJ Comput. Sci.*, vol. 7, p. e824, 2021.

[25] M. M. Ghorbani, F. F. Moghaddam, M. Zhang, M. Pourzandi, K. K. Nguyen, and M. Cheriet, "Malchain: Virtual application behaviour profiling by aggregated microservice data exchange graph," in *2020 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2020, pp. 41–48.

[26] H. Mohamed and O. El-Gayar, "End-to-end latency prediction of microservices workflow on kubernetes: A comparative evaluation of machine learning models and resource metrics," 2021.

[27] H. Chen, K. Wei, A. Li, T. Wang, and W. Zhang, "Trace-based Intelligent Fault Diagnosis for Microservices with Deep Learning. In 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)." IEEE, 2021.

[28] C. Hou, T. Jia, Y. Wu, Y. Li, and J. Han, "Diagnosing performance issues in microservices with heterogeneous data source," in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, 2021, pp. 493–500.

[29] J. Cui, P. Chen, and G. Yu, "A learning-based dynamic load balancing approach

for microservice systems in multi-cloud environment," in *2020 IEEE 26th international conference on parallel and distributed systems (ICPADS),* 2020, pp. 334–341.

[30] S. Ji, W. Wu, and Y. Pu, "Multi-indicators prediction in microservice using Granger causality test and Attention LSTM," in *2020 IEEE World Congress on Services (SERVICES),* 2020, pp. 77–82.

[31] H. X. Nguyen, S. Zhu, and M. Liu, "A Survey on Graph Neural Networks for Microservice-Based Cloud Applications," *Sensors*, vol. 22, no. 23, p. 9492, 2022.

[32] M. Caporuscio, M. De Toma, H. Muccini, and K. Vaidhyanathan, "A machine learning approach to service discovery for microservice architectures," in *Software Architecture: 15th European Conference, ECSA 2021, Virtual Event, Sweden, September 13-17, 2021, Proceedings,* 2021, pp. 66–82.

[33] R. Yu, S.-Y. Lo, F. Zhou, and G. Xue, "Data-Driven Edge Resource Provisioning for Inter-Dependent Microservices with Dynamic Load," in *2021 IEEE Global Communications Conference (GLOBECOM),* 2021, pp. 1–6.

[34] M. S. A. Khan and R. Santhosh, "Hybrid Optimization Algorithm for VM Migration in Cloud Computing," *Comput. Electr. Eng.*, vol. 102, Sep. 2022, doi: 10.1016/j.compeleceng.2022.108152.

[35] S. Srichandan, T. Ashok Kumar, and S. Bibhudatta, "Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm," *Futur. Comput. Informatics J.*, vol. 3, no. 2, pp. 210–230, Dec. 2018, doi: 10.1016/j.fcij.2018.03.004.

[36] Q. Wang, G. Zhou, R. Song, Y. Xie, M. Luo, and T. Yue, "Continuous space ant colony algorithm for automatic selection of orthophoto mosaic seamline network," *ISPRS J. Photogramm. Remote Sens.*, vol. 186, pp. 201–217, Apr. 2022, doi: 10.1016/j.isprsjprs.2022.02.011.

[37] N. Manikandan, P. Divya, and S. Janani, "BWFSO: Hybrid Black-widow and Fish swarm optimization Algorithm for resource allocation and task scheduling in cloud computing," *Mater. Today Proc.*, vol. 62, pp. 4903–4908, Jan. 2022,

doi: 10.1016/j.matpr.2022.03.535.

[38] S. M. Hussain and G. R. Begh, “Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog–cloud environment,” *J. Comput. Sci.*, vol. 64, Oct. 2022, doi: 10.1016/j.jocs.2022.101828.

[39] P. M. Kumar, U. Devi G, G. Manogaran, R. Sundarasekar, N. Chilamkurti, and R. Varatharajan, “Ant colony optimization algorithm with Internet of Vehicles for intelligent traffic control system,” *Comput. Networks*, vol. 144, pp. 154–162, Oct. 2018, doi: 10.1016/j.comnet.2018.07.001.

[40] M. L. Chiang, H. C. Hsieh, Y. H. Cheng, W. L. Lin, and B. H. Zeng, “Improvement of tasks scheduling algorithm based on load balancing candidate method under cloud computing environment,” *Expert Syst. Appl.*, vol. 212, p. 118714, Feb. 2023, doi: 10.1016/J.ESWA.2022.118714.

[41] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, “Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends,” *Swarm Evol. Comput.*, vol. 62, p. 100841, Apr. 2021, doi: 10.1016/J.SWEVO.2021.100841.

[42] W. Li, J. Jing, Y. Chen, and Y. Chen, “A cooperative particle swarm optimization with difference learning,” *Inf. Sci. (Ny).*, vol. 643, Sep. 2023, doi: 10.1016/j.ins.2023.119238.

[43] S. Mangalampalli, G. R. Karri, and A. A. Elngar, “An Efficient Trust-Aware Task Scheduling Algorithm in Cloud Computing Using Firefly Optimization,” *Sensors*, vol. 23, no. 3, Feb. 2023, doi: 10.3390/S23031384.

[44] N. Venkataraman, “Threshold based multi-objective memetic optimized round robin scheduling for resource efficient load balancing in cloud,” *Mob. Networks Appl.*, vol. 24, pp. 1214–1225, 2019.

[45] S. Mangalampalli, G. R. Karri, and G. N. Satish, “Efficient Workflow Scheduling algorithm in cloud computing using Whale Optimization,” *Procedia Comput. Sci.*, vol. 218, pp. 1936–1945, 2023.

[46] A. Jyoti and M. Shrimali, “Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing,” *Cluster Comput.*, vol.

23, pp. 377–395, 2020.

[47] K. Karthiban and J. S. Raj, “An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm,” *Soft Comput.*, vol. 24, no. 19, pp. 14933–14942, 2020.

[48] A. Sunyaev, *Internet computing: Principles of Distributed systems and emerging internet-based technologies*. Springer Nature, 2020.

[49] R. Singhal and A. Singhal, “A feedback-based combinatorial fair economical double auction resource allocation model for cloud computing,” *Futur. Gener. Comput. Syst.*, vol. 115, pp. 780–797, 2021.

[50] J. Praveenchandar and A. Tamilarasi, “Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing,” *J. Ambient Intell. Humaniz. Comput.*, vol. 12, pp. 4147–4159, 2021.

[51] K. Sreenu and S. Malempati, “MFGMTS: Epsilon constraint-based modified fractional grey wolf optimizer for multi-objective task scheduling in cloud computing,” *IETE J. Res.*, vol. 65, no. 2, pp. 201–215, 2019.

[52] R. Liu, L. Peng, J. Liu, and J. Liu, “A diversity introduction strategy based on change intensity for evolutionary dynamic multiobjective optimization,” *Soft Comput.*, vol. 24, pp. 12789–12799, 2020.

[53] G. Kiruthiga and S. Mary Vennila, “An Enriched Chaotic Quantum Whale Optimization Algorithm Based Job scheduling in Cloud Computing Environment,” *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 6, no. 4, pp. 1753–1760, 2019.

[54] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, “An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments,” *Neural Comput. Appl.*, vol. 32, pp. 1531–1541, 2020.

[55] J. O. Gutierrez-Garcia and A. Ramirez-Nafarrate, “Agent-based load balancing in cloud data centers,” *Cluster Comput.*, vol. 18, pp. 1041–1062, 2015.

[56] A. Singh, D. Juneja, and M. Malhotra, “A novel agent based autonomous and service composition framework for cost optimization of resource provisioning in

cloud computing," *J. King Saud Univ. Inf. Sci.*, vol. 29, no. 1, pp. 19–28, 2017.

[57] D. Armstrong, K. Djemame, and R. Kavanagh, "Towards energy aware cloud computing application construction," *J. Cloud Comput.*, vol. 6, no. 1, pp. 1–13, 2017.

[58] L. Abualigah, A. Diabat, P. Sumari, and A. H. Gandomi, "Applications, deployments, and integration of internet of drones (iod): a review," *IEEE Sens. J.*, vol. 21, no. 22, pp. 25532–25546, 2021.

[59] P. P. G. Gopinath and S. K. Vasudevan, "An in-depth analysis and study of Load balancing techniques in the cloud computing environment," *Procedia Comput. Sci.*, vol. 50, pp. 427–432, 2015.

[60] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *J. Cloud Comput.*, vol. 7, no. 1, pp. 1–16, 2018.

[61] X. Xu *et al.*, "Dynamic resource allocation for load balancing in fog environment," *Wirel. Commun. Mob. Comput.*, vol. 2018, 2018.

[62] Y. M. Ko and Y. Cho, "A distributed speed scaling and load balancing algorithm for energy efficient data centers," *Perform. Eval.*, vol. 79, pp. 120–133, 2014.

[63] G. Rjoub, J. Bentahar, and O. A. Wahab, "BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments," *Futur. Gener. Comput. Syst.*, vol. 110, pp. 1079–1097, 2020.

[64] S. K. Panda, S. S. Nanda, and S. K. Bhoi, "A pair-based task scheduling algorithm for cloud computing environment," *J. King Saud Univ. Inf. Sci.*, vol. 34, no. 1, pp. 1434–1445, 2022.

[65] A. Ullah, I. A. S. Umeriqbal, A. Rauf, O. Y. Usman, S. Ahmed, and Z. Najam, "An Energy-Efficient Task Scheduling using BAT Algorithm for Cloud Computing." August, 2019.

[66] Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai, "An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning," *J. Supercomput.*, vol. 76, pp. 455–480, 2020.

[67] K. Lee and K. Kim, “A performance evaluation of a geo-spatial image processing service based on open source PaaS cloud computing using cloud foundry on OpenStack,” *Remote Sens.*, vol. 10, no. 8, p. 1274, 2018.

[68] C. Wang, S. Li, T. Cheng, and B. Li, “A construction of smart city evaluation system based on cloud computing platform,” *Evol. Intell.*, vol. 13, pp. 119–129, 2020.

[69] Q. Sun, Z. Tan, and X. Zhou, “Workload prediction of cloud computing based on SVM and BP neural networks,” *J. Intell. Fuzzy Syst.*, vol. 39, no. 3, pp. 2861–2867, 2020.

[70] L. Tang, B. Tang, L. Kang, and L. Zhang, “A novel task caching and migration strategy in multi-access edge computing based on the genetic algorithm,” *Futur. Internet*, vol. 11, no. 8, p. 181, 2019.

[71] K. Cui, B. Lin, W. Sun, and W. Sun, “Learning-based task offloading for marine fog-cloud computing networks of USV cluster,” *Electronics*, vol. 8, no. 11, p. 1287, 2019.

[72] X. Xu, R. Gu, F. Dai, L. Qi, and S. Wan, “Multi-objective computation offloading for internet of vehicles in cloud-edge computing,” *Wirel. Networks*, vol. 26, pp. 1611–1629, 2020.

[73] Z. Lv and W. Xiu, “Interaction of edge-cloud computing based on SDN and NFV for next generation IoT,” *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5706–5712, 2019.

[74] C. Ming, Y. Bingjie, and L. Xiantong, “Multi-tenant SaaS deployment optimisation algorithm for cloud computing environment,” *Int. J. Internet Protoc. Technol.*, vol. 11, no. 3, pp. 152–158, 2018.

[75] H. Kim, J. Kim, Y. Kim, I. Kim, and K. J. Kim, “Design of network threat detection and classification based on machine learning on cloud computing,” *Cluster Comput.*, vol. 22, pp. 2341–2350, 2019.

[76] C.-Y. Liu, C.-M. Zou, and P. Wu, “A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing,” in *2014 13th International Symposium on Distributed Computing and Applications to*

Business, Engineering and Science, 2014, pp. 68–72.

[77] M. Abbasi, E. Mohammadi Pasand, and M. R. Khosravi, “Workload allocation in iot-fog-cloud architecture using a multi-objective genetic algorithm,” *J. Grid Comput.*, vol. 18, no. 1, pp. 43–56, 2020.

[78] R. Sundarrajan and V. Vasudevan, “An optimization algorithm for task scheduling in cloud computing based on multi-purpose cuckoo seek algorithm,” in *Theoretical Computer Science and Discrete Mathematics: First International Conference, ICTCSDM 2016, Krishnankoil, India, December 19-21, 2016, Revised Selected Papers 1*, 2017, pp. 415–424.

[79] M. B. Gawali and S. K. Shinde, “Standard deviation based modified cuckoo optimization algorithm for task scheduling to efficient resource allocation in cloud computing,” *J. Adv. Inf. Technol*, 2017.

[80] R. Valarmathi and T. Sheela, “Ranging and tuning based particle swarm optimization with bat algorithm for task scheduling in cloud computing,” *Cluster Comput.*, vol. 22, pp. 11975–11988, 2019.

[81] H. Alazzam, E. Alhenawi, and R. Al-Sayyed, “A hybrid job scheduling algorithm based on Tabu and Harmony search algorithms,” *J. Supercomput.*, vol. 75, no. 12, pp. 7994–8011, 2019.

[82] E. Rani and H. Kaur, “Efficient Load Balancing Task Scheduling in Cloud Computing using Raven Roosting Optimization Algorithm.,” *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, 2017.

[83] D. Saxena, R. K. Chauhan, and R. Kait, “Dynamic fair priority optimization task scheduling algorithm in cloud computing: concepts and implementations,” *Int. J. Comput. Netw. Inf. Secur.*, vol. 8, no. 2, p. 41, 2016.

[84] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, “A comprehensive survey for scheduling techniques in cloud computing,” *J. Netw. Comput. Appl.*, vol. 143, pp. 1–33, 2019.

[85] R. Buyya *et al.*, “A manifesto for future generation cloud computing: Research directions for the next decade,” *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–38, 2018.

[86] C.-W. Tsai and J. J. P. C. Rodrigues, “Metaheuristic scheduling for cloud: A survey,” *IEEE Syst. J.*, vol. 8, no. 1, pp. 279–291, 2013.

[87] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, “A taxonomy and survey of energy-efficient data centers and cloud computing systems,” *Adv. Comput.*, vol. 82, pp. 47–111, 2011.

[88] S. S. Gill and R. Buyya, “A taxonomy and future directions for sustainable cloud computing: 360 degree view,” *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–33, 2018.

[89] S. Singh and I. Chana, “A survey on resource scheduling in cloud computing: Issues and challenges,” *J. grid Comput.*, vol. 14, pp. 217–264, 2016.

[90] M. Xu and R. Buyya, “Brownout approach for adaptive management of resources and applications in cloud computing systems: A taxonomy and future directions,” *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–27, 2019.

[91] M. Kalra and S. Singh, “A review of metaheuristic scheduling techniques in cloud computing. Egypt Inform. J. 16, 275–295 (2015).”

[92] M. Carillo, G. Cordasco, F. Serrapica, V. Scarano, C. Spagnuolo, and P. Szufel, “Distributed simulation optimization and parameter exploration framework for the cloud,” *Simul. Model. Pract. Theory*, vol. 83, pp. 108–123, 2018.

[93] A. K. Bhardwaj, Y. Gajpal, C. Surti, and S. S. Gill, “HEART: Unrelated parallel machines problem with precedence constraints for task scheduling in cloud computing using heuristic and meta-heuristic algorithms,” *Softw. Pract. Exp.*, vol. 50, no. 12, pp. 2231–2251, 2020.

[94] A. V. Lakra and D. K. Yadav, “Multi-objective tasks scheduling algorithm for cloud computing throughput optimization,” *Procedia Comput. Sci.*, vol. 48, pp. 107–113, 2015.

[95] X. Gao, R. Liu, and A. Kaushik, “Hierarchical multi-agent optimization for resource allocation in cloud computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 692–707, 2020.

[96] J. Bajo, F. la Prieta, J. M. Corchado, and S. Rodr\'\iguez, “A low-level resource allocation in an agent-based Cloud Computing platform,” *Appl. Soft Comput.*,

vol. 48, pp. 716–728, 2016.

[97] W. Wang, Y. Jiang, and W. Wu, "Multiagent-based resource allocation for energy minimization in cloud computing systems," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 47, no. 2, pp. 205–220, 2016.

[98] M. Adhikari and T. Amgoth, "Heuristic-based load-balancing algorithm for IaaS cloud," *Futur. Gener. Comput. Syst.*, vol. 81, pp. 156–165, 2018.

[99] S. Mustafa, K. Bilal, S. U. R. Malik, and S. A. Madani, "SLA-aware energy efficient resource management for cloud environments," *IEEE Access*, vol. 6, pp. 15004–15020, 2018.

[100] A. Marahatta, Y. Wang, F. Zhang, A. K. Sangaiah, S. K. S. Tyagi, and Z. Liu, "Energy-aware fault-tolerant dynamic task scheduling scheme for virtualized cloud data centers," *Mob. Networks Appl.*, vol. 24, pp. 1063–1077, 2019.

[101] Y. Sharma, W. Si, D. Sun, and B. Javadi, "Failure-aware energy-efficient VM consolidation in cloud computing systems," *Futur. Gener. Comput. Syst.*, vol. 94, pp. 620–633, 2019.

[102] T. Tamilvizhi and B. Parvathavarthini, "A novel method for adaptive fault tolerance during load balancing in cloud computing," *Cluster Comput.*, vol. 22, pp. 10425–10438, 2019.

[103] A. Belgacem, "Dynamic resource allocation in cloud computing: analysis and taxonomies," *Computing*, vol. 104, no. 3, pp. 681–710, 2022.

[104] T. Kaur and I. Chana, "Energy efficiency techniques in cloud computing: A survey and taxonomy," *ACM Comput. Surv.*, vol. 48, no. 2, pp. 1–46, 2015.

[105] A. Hameed *et al.*, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, pp. 751–774, 2016.

[106] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[107] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: a big picture," *J. King Saud Univ. Inf. Sci.*, vol. 32, no. 2, pp. 149–158, 2020.

[108] F. la Prieta, S. Rodr\'\iguez-González, P. Chamoso, J. M. Corchado, and J. Bajo, "Survey of agent-based cloud computing applications," *Futur. Gener. Comput. Syst.*, vol. 100, pp. 223–236, 2019.

[109] P. Lou, Z. Zhou, Y.-P. Chen, and W. Ai, "Study on multi-agent-based agile supply chain management," *Int. J. Adv. Manuf. Technol.*, vol. 23, pp. 197–203, 2004.

[110] A. Belgacem, K. Beghdad-Bey, and H. Nacer, "Task scheduling optimization in cloud based on electromagnetism metaheuristic algorithm," in *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, 2018, pp. 1–7.

[111] J. Dogani, F. Khunjush, M. R. Mahmoudi, and M. Seydali, "Multivariate workload and resource prediction in cloud computing using CNN and GRU by attention mechanism," *J. Supercomput.*, vol. 79, no. 3, pp. 3437–3470, 2023.

[112] T. Catena, V. Eramo, M. Panella, and A. Rosato, "Distributed LSTM-based cloud resource allocation in Network Function Virtualization Architectures," *Comput. Networks*, vol. 213, p. 109111, 2022.

[113] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, 2021.

[114] H. M. Khan, F.-F. Chua, and T. T. V. Yap, "ReSQoV: A Scalable Resource Allocation Model for QoS-Satisfied Cloud Services," *Futur. Internet*, vol. 14, no. 5, p. 131, 2022.

[115] Y. Wei, L. Pan, S. Liu, L. Wu, and X. Meng, "DRL-scheduling: An intelligent QoS-aware job scheduling framework for applications in clouds," *IEEE Access*, vol. 6, pp. 55112–55125, 2018.

[116] D.-Y. Lee, S.-Y. Jeong, K.-C. Ko, J.-H. Yoo, and J. W.-K. Hong, "Deep Q-network-based auto scaling for service in a multi-access edge computing environment," *Int. J. Netw. Manag.*, vol. 31, no. 6, p. e2176, 2021.

[117] H. Yuan, M. Zhou, Q. Liu, and A. Abusorrah, "Fine-grained resource provisioning and task scheduling for heterogeneous applications in distributed

green clouds," *IEEE/CAA J. Autom. Sin.*, vol. 7, no. 5, pp. 1380–1393, 2020.

[118] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM special interest group on data communication*, 2019, pp. 270–288.

[119] A. Moazeni, R. Khorsand, and M. Ramezanpour, "Dynamic Resource Allocation Using an Adaptive Multi-Objective Teaching-Learning Based Optimization Algorithm in Cloud," *IEEE Access*, 2023, doi: 10.1109/ACCESS.2023.3247639.

[120] H. Liu, "Research on cloud computing adaptive task scheduling based on ant colony algorithm," *Optik (Stuttg).*, vol. 258, p. 168677, May 2022, doi: 10.1016/J.IJLEO.2022.168677.