Soft Computing Techniques to Analyze the Load Balancing in Cloud Environment

Arulmurugan Ramu
Department of Computer Science, Mattu University, Metu, Ethiopia.
arulmr@gmail.com

Correspondence should be addressed to Arulmurugan Ramu**:** arulmr@gmail.com.

Article Info
Journal of Computing and Natural Science (http://anapub.co.ke/journals/jcns/jcns.html)
Doi: https://doi.org/10.53759/181X/JCNS202303001
Received 02 June 2022; Revised form 20 July 2022; Accepted 05 September 2022.
Available online 05 January 2023.

Published by AnaPub Publications

Abstract – An emerging method of digital computing known as "cloud computing" has recently gained immense popularity. While there are many benefits to using the cloud over the internet, there are also serious challenges that must be addressed in order to boost the efficiency of this method. Challenges to cloud computing via the internet include load balancing, work scheduling, fault tolerance, and several security concerns. The effectiveness of the cloud may be enhanced by fixing a number of problems, one of the most pressing being load balancing. To prevent any one node from being too overburdened or underused, a system employs a technique known as load balancing. In order to increase utilization and minimize overall task execution time, load balancing algorithms are developed to distribute work fairly across available resources. In this article, a review and comparison of different load-balancing algorithms is provided. This paper provides a basis of the application of different load-balancing techniques, which utilize the approach of soft computing within the cloud computing environment.

Keywords – Stochastic Hill Climbing (SHC), Genetic Algorithm (GA), Ant Colony Optimization (ACO), Load Balance Improved Min-Min Scheduling Algorithm (LBIMM).

I. INTRODUCTION

Today, the concept of the cloud is becoming increasingly popular with users all around the globe using cloud computing for a variety of purposes. Different types of activities, from business to schoolwork to private life, may benefit from the hardware and software made available by third parties via the cloud computing infrastructure. The term "cloud computing" refers to the delivery of services over an external network. E-mail, data storage, document, enterprise software, social media, and other services are all examples of cloud computing. Users may use the services offered by cloud computing from any location with an Internet connection since cloud computing offers a centralized location from which users may access a variety of services, such as a network, a computer cluster, business software, and data storage. By utilizing the cloud's on-demand services, users may purchase and control their own resources independently. Cloud computing may be divided into three primary service types i.e. Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS), each of which offers its own unique advantages to end users.

Software-as-a-Service (SaaS)

Software-as-a-Service (SaaS) alludes to a software authorizing and delivery framework where software is accredited based on a certain subscription and then hosted in a centralized location [1]. SaaS represents a framework of delivering computer application through the web, and is a type of computing, which also integrates Information Technology-as-a-Service (ITaaS), Integration Platform-as-a-Service (iPaaS),Infrastructure-as-a-Service (IaaS), Data Center-as-a-Service (DCaaS), Mobile Backend-as-a-Service (MBaaS), Platform-as-a-Service (PaaS) and Desktop-as-a Service (DaaS). In many cases, web browsers are all what are require accessing SaaS software. The messaging software, office software, payroll processing software, DBMS software, administration software, CAD software, development software, gamification, bookkeeping, virtualization, Management Information System (MIS), Customer Relation Management (CRM), field service management, invoicing, Enterprise Resource Planning (ERP), Talent Acquisition (TA), and Human Resource Management (HRM), and many other applications are delivered through SaaS framework. Using the SaaS approach, you may access prebuilt apps that already include an OS, hardware, and networking infrastructure.

Platform-as-a-Service (PaaS)

Platform-as-a-Service (PaaS) is a form of cloud computing service, which allows users to create, enhance and launch applications without any form of complexity of constructing and maintaining the requisite infrastructures [2]. By abstracting the difficulties of infrastructure management, PaaS makes it possible for enterprises and developers to build, host, and release applications (set up, configure, and manage components such as databases and servers). Using PaaS aids speed up the development process of applications so that the developer can concentrate on the app itself. With PaaS, the client is responsible for managing the applications and data, while IT department (private PaaS) or providers (public PaaS) handles the infrastructure. The vendor's development tools are tailored to each individual customer's requirements. According to El-Deeb [3], the customer has the option of either maintaining the software themselves or having the vendor do it. Services in this case include web service marshaling, database integration, scalability, security, persistence, application versioning, state management, developer communication facilitation, and application instrumentation. In addition to traditional service engineering features, PaaS also provides tools for managing services, such as discovery, reservation, and workflow management. The client in the PaaS framework installs or creates its software applications and makes use of the underlying platform's OS, hardware, and networking.

Infrastructure-as-a-Service (IaaS)

The term "infrastructure as a service" (IaaS) defines a specific class of cloud computing services, which provide on-demand accessibility to computing, networking, and storage resources on the pay-as-you-go basis. Alongside SaaS, serverless computing, and PaaS, IaaS composes of four main classes of cloud computing services. By switching to IaaS services, a firm may reduce the timeframe and funds spent on in-house data center maintenance, save on the cost of purchasing hardware, and enhances accessibility to real-time data for strategic decision making. With IaaS, it is possible to increase or decrease the amount of resources allocated to IT infrastructure in response to fluctuations in workload. IaaS aids in the speedy deployment of new applications and the improvement of the stability of the underlying infrastructure.

Enterprises may save money and time by using IaaS instead of purchasing and maintaining their own servers and datacenter hardware. Each resource is presented as a standalone service, and firms only pay for it for as long as they really use it. Using Azure or another cloud service, the underlying hardware is updated and maintained by the providers, but companies are obliged to ensure procuring, managing, configuring, and installing all software, including operating systems, middleware, and applications. IaaS solely sells its hardware and networks to its customers and builds its own applications and software.Users have access to the cloud computing system's resources through four major deployment servers. **Table 1** below shows the descriptions of four different types of cloud deployment servers

Table 1. Types of cloud deployment servers

Cloud deployment servers	Description
Public Cloud	Social networking sites, electronic mail services, and cloud storage spaces are all examples of this kind of service that can be accessed over the internet and are made open to users at no cost.
Private Cloud:	These clouds are operated by certain businesses and are only accessible to authorized users.
Community Cloud	The community cloud is hosted within the cloud system and shared amongst a select group of businesses.
Hybrid Cloud	Two clouds are used, and multiple forms of resource sharing are combined in this approach.

This article presents a review of different load-balancing methods [GA, ACO Algorithm, SHC Algorithm, Fair RRAlgorithm, TA, BEE-MMT and LBIMM], and presents a performance comparison between these methods. The remainder of the article is arranged as follows: Section II presents an analysis of load balancing in the cloud computing environment. Section III focuses on a review of cloud load balancing algorithms from a soft computing approach, as well as a comparative study of the algorithms. Section IV draws final remarks to the research.

II. LOAD BALANCING IN CLOUD COMPUTING ENVIRONMENT

When it comes to cloud computing, it's crucial that client requests, which might occur at varying intervals, be processed in a timely manner. This emphasizes the critical need of spreading the workload over many servers. A system's load is proportional to the sum of its Central Processing Unit (CPU) load, memory use, and network delay load. Load balancing is an approach of distributing work throughout a model such that no one component is overburdened, underused, or idle. The goals of a good load balancing strategy are to maximize throughput, optimize resource use, prevent resource overload, and reduce reaction time as much as possible. One may use either static or Dynamic Load Balancing (DLB).

Static Load Balancing

Static Load Balancing Definition

In order to keep individual servers from getting overwhelmed, a load balancer may be implemented in the form of hardware or software. Algorithms are utilized by load balancers to divide the traffic of networks across multiple services. The algorithms of load balancing may be either static or dynamic, depending on the requirements of the system. When distributing work, Static Load Balancing (SLB) algorithms in the cloud do not take into consideration things like the system's current status or metrics like the processors' current load levels. Static algorithms are established for models with minimal load variability, and they do this by dividing traffic evenly across servers or according to other principles that are insensitive to system variables. In order to optimize processor performance, SLB methods need in-depth knowledge of server resources at the time of implementation. In order to prioritize the server with the least amount of work, DLB algorithms need constant network connectivity. The dynamic algorithm adjusts the workload according to the current conditions of the system, shifting traffic from idle to busy nodes in real time. **Fig 1** represents the process of SLB.

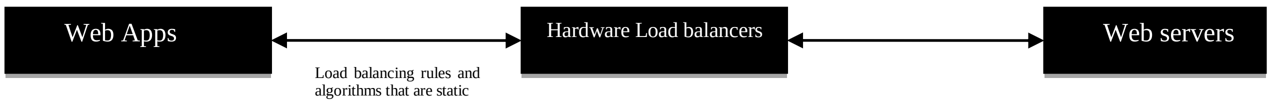

Fig 1. SLB process flow

Approaches to Static Load Balancing

There is often not a lot of foresight into the activities involved in static load distribution. Even if the execution duration is unknown, static load distribution may be performed. **Table 2** highlights some of the approaches to SLB.

Table 2. SLB approaches

Approach	Brief description
Round Robin (RR)	The RR load balancing method is one of the most used and basic load balancing techniques, distributing requests from clients in a randomized fashion among available application servers. It sends requests from clients to application servers in the sequence they were received, without taking into account the servers' individual capabilities.
Weighted RR	Within the framework of the original RR load balancing algorithm, a weighted RR load balancing method takes into consideration a number of characteristics of the application servers involved. The algorithm "weights" units according to predetermined criteria, most often their capacity to handle traffic, and distributes requests accordingly. Weighted units would get more requests than the traditional round robin 1 by 1 allocation to each server in turn.
Opportunistic/Randomized Static	Opportunistic/randomized algorithms distribute workload among servers at random, regardless of how busy those servers currently are. Performance decreases with increasing task size; therefore it is best used for lesser jobs. Sometimes bottlenecks develop because of the unpredictability of the distribution. Adding insult to injury, a high-demand machine might be given more work at any moment.

As a result of this method, the algorithm is pre-aware of all available system resources, and the workload is spread uniformly across all nodes.

Dynamic Load Balancing:

Dynamic Load Balancing Definition

Dynamic Load Balancing (DLB) [4] is a software solution that guarantees even system load by letting each parallel process handles its own load balancing at the application level. As long as the parallel work includes executable code, the DLB can balance the load among computers with diverse architectures and operating systems. Both the DLB system and its users play important roles in the DLB ecosystem. If a person has an account on any of the computers in the system, they will have access to all of those machines. The system keeps track of information like how many people are logged in, how fast their connections are, and the overall status, so that users may access this data whenever they need it for purposes like load balancing and monitoring. The users themselves launch parallel tasks and submit them to the DLB environment using a graphical user interface. **Fig 2** shows the processes and outline of DLB, which **Table 3** highlights the approaches to DLB.

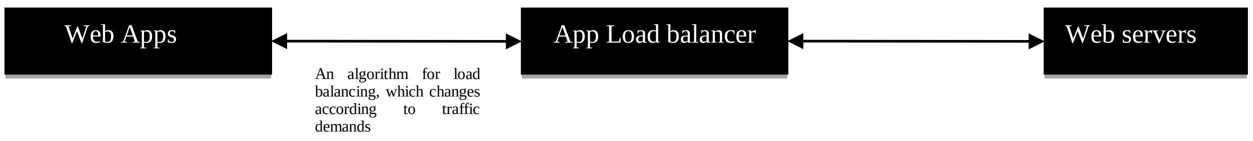
Web Apps
App Load balancer
Web servers
An algorithm for load balancing, which changes according to traffic demands

Fig 2. DLB process flow

Table 3. Approaches to DLB

Approach	Brief Description
Least connection	Determines which servers currently have the fewest active connections and directs traffic there. This presupposes that the computational needs of all connections are substantially equivalent.
Weighted least connection	Allows administrators to prioritize some servers over others if they can handle a higher volume of traffic.
Weighted response time	Uses a weighted average of each server's response time and the number of active connections to make routing decisions. In order to provide consumers with speedier service, the algorithm directs traffic to the servers with the shortest response time.
Resource-based	The available resources on each server are taken into account before deciding how to distribute the workload. The load balancer determines which server to send a certain amount of work to base on the agent software's report of the server's available processing power and memory.

When using this method, load balancing is dynamic and adapts to the operational conditions of the system. When one of the nodes becomes overburdened, the workload is transferred to the other nodes.

III. DISCUSSION AND COMPARATIVE STUDY

Cloud Load Balancing Algorithms from a Soft Computing Approach

Load balancing in cloud computing is a relatively new technique that utilizes a number of different soft computing methodologies to distribute workloads among available resources, to the extent that no available resource goes unused. In this section, a number of different load balancing methods that use a soft computing approach will be reviewed.

Genetic Algorithm

The GA-based Load Balancingwas introduced by Saputra et al. [5] in 2013. In this research, we show how evolutionary algorithms may be used to balance workloads in the cloud in order to reduce the mean completion time for groups of tasks. To maintain optimal performance when cloud deployments grow in scale, service providers use load balancing techniques such as RR, Min-Min, SHC, and First Come First Served (FCFS). The GA, however, has been utilized here because of its faster reaction time compared to the aforementioned methods. Increased performance and efficient load balancing are achieved by the GA's usage of a technique based on natural selection. Increasing the number of VMs and the size of the data sets allows the GA to improve its performance, which in turn decreases the reaction time.

A GA is a kind of search algorithm that takes inspiration from evolutionary theory and natural selection. GAs takes use of both known and undiscovered information in the search space. A GA may approximate the inventiveness of a human search in certain respects by using survival of the fittest methods in conjunction with a structured but random flow of information. A group of synthetic organisms that have a common ancestor (strings). Each successive generation generates its own unique collection of strings by incorporating data from earlier iterations. Although GA has an element of randomness, they are not the same as a purely random path. Effectively using past data, they may make educated guesses about where to go for future improvements in search performance. The vast majority of optimization strategies involve jumping from one decision point to another in the decision space, with each jump being determined by a different transition rule. In multimodal (many-peaked) search spaces, such as the human brain, this point-to-point approach may easily identify spurious peaks. In contrast, GAs use a database of points at once (a population of strings), concurrently ascending numerous peaks.

This algorithm has a lower chance of discovering a false peak than point-to-point approaches. When broken down to their bare bones, the mechanics of a basic GA consist of nothing more than copying strings and changing out fragments of strings. The GA method is appealing due to its ease of use and potency of impact. A balance between discovery and use is crucial to the success of the GA.

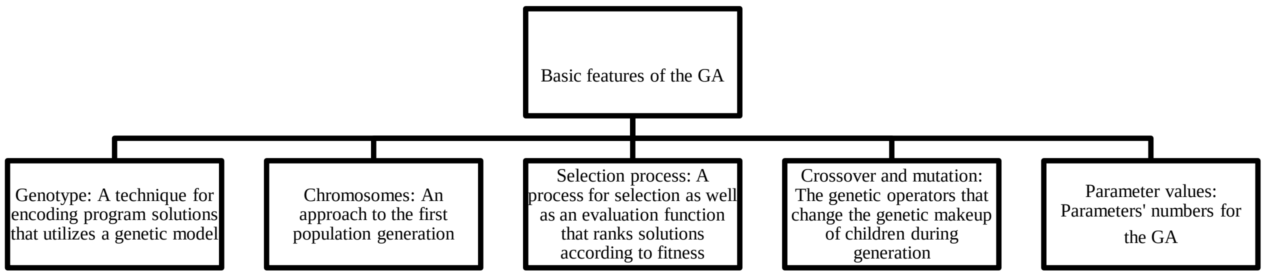
Basic features of the GA
Genotype: A technique for encoding program solutions that utilizes a genetic model
Chromosomes: An approach to the first population generation
Selection process: A process for selection as well as an evaluation function that ranks solutions according to fitness
Crossover and mutation: The genetic operators that change the genetic makeup of children during generation
Parameter values: Parameters' numbers for the GA

Fig 3. Basic components of the Gas

Selecting, crossing, and mutating are three tools that may help do this. Exploitation stems from the process of selection based on fitness. Mutation and crossover operators are the pioneers. They need to wreak havoc on some of the strings they rely on in order to go exploring. Mutation best exemplifies the exploration vs. exploitation tradeoff. As the rate of mutation grows, mutation becomes more disruptive, eventually nullifying the beneficial benefits of selection.

GAs, the most well-known form of evolutionary computing technology, is effective stochastic search and optimization tools with wide-ranging applications. In most cases, a GA will consist of the following five features as depicted in **Fig 3** below:

Ant Colony Optimization Algorithm

Ahmid, Dao, and Van Le [6] introduced the ACOAlgorithm in 2012, which is based on a soft computing approach to load balancing. The study proposes using ACO. According to this research, ants are described as being rather simple in terms of their behavior. Because of their short-term memory and reliance on individual action, ants are notoriously unpredictable. It has been shown that ants can do a variety of complex activities more reliably and consistently than humans can. Ants use pheromone trails to guide each other from the nest to the source of food and back, and this inspired a technique known as ACO, which optimizes for the minimized route. In the same way as honeybees use a complex foraging system to signal the quality and availability of food to their fellow foragers, so too can we use similar approaches to decipher the bees' messages. The ant colony method uses pheromone tables, where each column reflects the chance of selecting a neighbor as the next hop based on the destination. The shortcomings of this method may be mitigated by using it in a cloud-based setting.

Network connection load balancing is an NP- Complete problem; hence it is not viable to employ a standard GA, as was explained in a 2013 presentation by Afzal and Kavitha [7]. Toascertain the best possible answer for the whole problem, quickly and efficiently, the use of a GA is highly recommended. Darwin's idea of natural selection is the basis for GA's assessment criteria. Connection bandwidth link usage constraints, delays, and other variables are taken into consideration in link load balancing in order to evenly distribute network demand. Here, the standard GA's Roulette Wheel selection mechanism has been upgraded with an improved GA that makes use of both optimum retention and the roulette selection mechanism. An enhanced evolutionary algorithm is designed to address the network link load issue, which is essential for balancing traffic throughout the network. They were able to achieve their goals of maximizing link usage, minimizing latency throughout the network, and finding the optimal global routing solution by using a refined evolutionary algorithm.

ACO represents a probabilistic methodology employed in computer engineering and operational research to identify optimal pathways in graphs, which are the building blocks of many computing problems. Multi-agent approaches that mimic the actions of actual ants are sometimes referred to as "artificial ants." Commonly, the pheromone-based communication of actual ants is employed as a model. Multiple optimization problems, such as vehicle routing and internet routing, have grown to favor the use of hybrid approaches that combine artificial ants with local search methods. For instance, there is a group of optimization techniques known as ACO that mimic the behavior of a colony of ants. Artificial "ants" (such as the simulated agent) might potentially identify optimum solutions by evaluating the parameter spaces reflecting the potential solutions. Actual ants utilize pheromones to signal locations of different sources of food while away from the food hunt.

The simulated 'ants' recall their positions and the precision of solutions they identity, so that successive simulation cycles may have more ants locate optimal solutions. The honey bee is another social creature that serves as inspiration for a variant of this method called the Bees algorithm. This method is a member of the ant colony algorithm family, which is widely employed in swarm intelligence approaches and represents a number of metaheuristic optimizations. In 1992, Marco Dorigo introduced the first algorithm, which took its cues from the scavenging behavior of ants, in his PhD dissertation. Its goal was to identify the shortest route in a graph. Since the initial notion was developed to address a broader category of numerical problems, other issues have arisen, each of which borrows in some way from ant behavior. ACO, like several other algorithms for estimating distributions, uses models to guide its search.The best features of the ACO algorithm are depicted in **Fig 4**

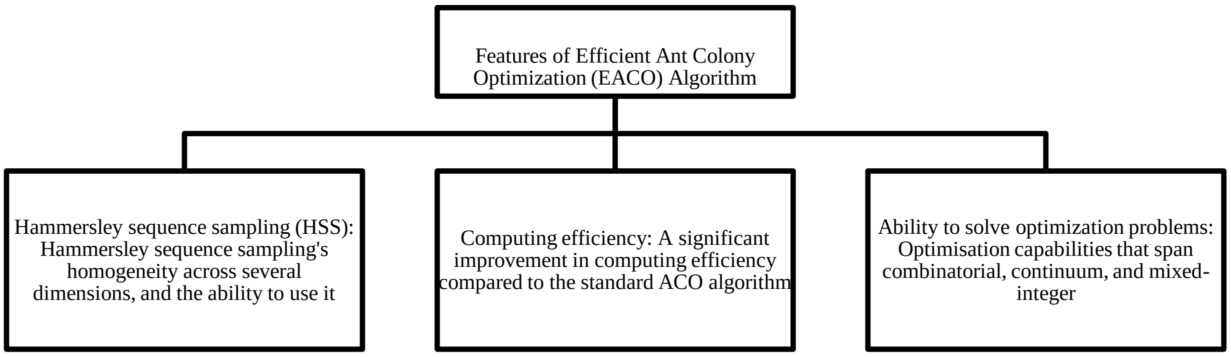
Features of Efficient Ant Colony Optimization (EACO) Algorithm
Hammersley sequence sampling (HSS): Hammersley sequence sampling's homogeneity across several dimensions, and the ability to use it
Computing efficiency: A significant improvement in computing efficiency compared to the standard ACO algorithm
Ability to solve optimization problems: Optimisation capabilities that span combinatorial, continuum, and mixed-integer

Fig 4. Features of EACO Algorithm

Stochastic Hill Climbing Algorithm

When it comes to numerical analysis, hill climbing is a kind of local search that uses mathematics to achieve optimization goals. It is an iterative method that begins with a chosen solution and iteratively modifies that solution in an effort to improve it. If the adjustment improves the situation, it is repeated with the new answer until no more adjustments can be made. The hill-climbing analogy may be used to help solve the traveling salesman issue, for instance. You can quickly generate a solution that travels to each city, but keep in mind that it probably isn't going to be very good compared to the best possible answer. The method takes such a solution as input and modifies it in small ways, such as switching the sequence in which cities are explored. Ultimately, a quicker and more direct method will be found.

However, SHC only finds local optima (structures that cannot be improved upon by any adjacent reconfiguration) for non-convex problems, but it always finds the optimum solution (the globalized optimum) for convex problems (search spaces). Examples of algorithms that employ hill-climbing to solve convex problems include the simplex technique for regression analysis and binary search. It is possible to avoid being stuck in local optimization by using strategies like restarts (i.e., repeated localized search), iteration-based systems (such as iterated local optimum), memory-based schemes (such reactive search optimisation and tabu searches), and memory-less probabilistic changes (such as simulated annealing).

The SHC Algorithm was introduced as a new method for load balancing by Stubbs, Wilson, and Rostami [8] in 2013. This study employs a load-balancing method based on soft computing. One that focuses on maximizing performance in a certain area, allocating incoming workloads to physical or virtual computers using SHC maximizes efficiency. The algorithm's ease of use makes it a go-to option for those just getting started with optimizing systems. To go from one state to another in artificial intelligence, it is often used to traverse graphs. Related algorithms employ a wide variety of options for both the next and initial nodes. In certain cases, SHC is equally as effective as more sophisticated algorithms like simulated annealing or tabu search. As long as a modest number of adjustments generally converges on a good solution, SHC could often produce a better outcome than other approaches when the opportunity available to execute a search is limited, as it often is in real-time applications (the optimal remedy or a closer approximations). Another way to look at bubble sort is as a SHC algorithms (each close element exchange lowers the number of disorganized element pairs), although this approach is expensive for even small N owing to the quadratic rise in the number of exchanges required.

The Load Optimization and SHCAlgorithm is a recursive approach, which climbs steadily in a single direction (uphill) until it reaches a peak beyond which no neighbors climb. To better distribute workloads, Kumar, Kumar, Bajaj, and Singh [9]proposed a max min strategy. In this work, the authors propose a new variation of the max-min algorithm that uses predicted time as its foundation and makes its selections accordingly. With the max-min method, resources are prioritized for higher-priority activities while working in parallel on lower-priority ones. The duration of the whole process is determined by how long each individual job takes to complete.

By having larger works completed by slower resources and smaller projects completed in parallel by the quickest resources, the waiting time for smaller jobs has been reduced. This allows a greater number of smaller jobs to be completed while at least one larger job is completed. There is a better chance of finding complementary resources. The algorithm divides up the input tasks among the available resources and estimates how long it will take to complete each one. After identifying all tasks, assigning the one with the greatest projected execution time to a capacity with the shortest anticipated completion time is performed, and the process is repeated until all tasks have been assigned.

SHC is a kind of Local Optimization algorithm that belongs to the broader category of Stochastic Optimization (compared to Global Optimization). As no further steps are taken to broaden the search area, this technique has the feel of a straight search. Predictable hill climbing algorithms such as Simple Hill Climbing (first-best neighbor) and Steepest-Ascent Hill Climbing are extensions of SHC, which is the grandmother of methods such as Random-Restart Hill Climbing and Parallel Hill Climbing. The SHC method works by repeatedly picking a neighboring candidate solution at random and accepting it if and only if it improves the situation. Because of their greedy acceptance of surrounding movements, deterministic hill-climbing methods are prone to become trapped in local optimums, motivating the proposal of this solution to escape this trap. **Fig 5** depicts some of Hill Climbing Algorithm's most distinguishing features:

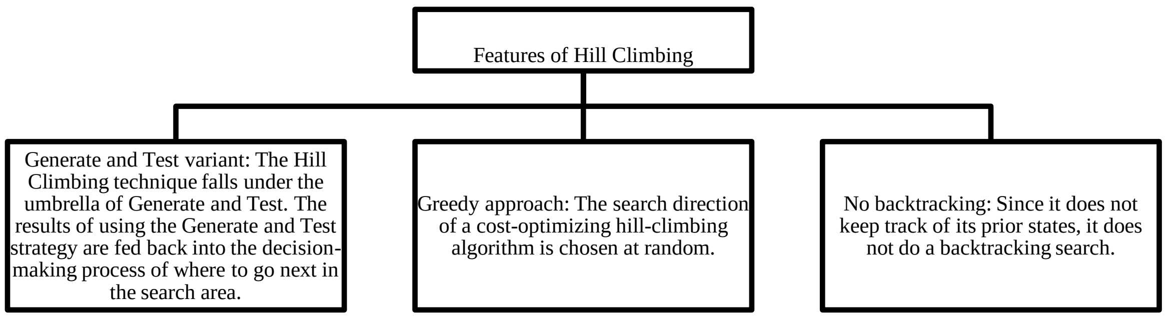
Features of Hill Climbing
Generate and Test variant: The Hill Climbing technique falls under the umbrella of Generate and Test. The results of using the Generate and Test strategy are fed back into the decision-making process of where to go next in the search area.
Greedy approach: The search direction of a cost-optimizing hill-climbing algorithm is chosen at random.
No backtracking: Since it does not keep track of its prior states, it does not do a backtracking search.

Fig 5. Features of Hill Climbing

Fair Round Robin Algorithm

Round robin was proposed in 2014 by Prasad [10] for load balancing amongst cloud-based VMs. In this study, the author lays out the details of an enhanced RR Algorithm known as theFairRR Algorithm, which is the quickest and easiest method for balancing workloads in the cloud. Time-scheduling is used to distribute incoming requests to available VMs. If a request's associated job is finished within its given time window, the request is marked as "done," and the VM's resources are made available to other processes. To determine the intelligent time,RR algorithm is used, and one of its components is called Priority Based Round Robin (PBDRR). PBDRR allocates distinct time Quantum to each task, and the processes in priority order. The author recommends the FairRR algorithm because it keeps the focus on ease of use. More work is required to execute a complex algorithm, which in turn slows things down. If the incoming request load tends to spike at unpredictable times, it is recommended to rely on the FairRR to get the most out of the schedule.

It is common practice for computer process and network schedulers to use Round-Robin (RR) as one of their algorithms. In the common usage of the term, time slices (or time quanta) are distributed uniformly across all running processes in a clockwise fashion (also known as cyclic executive). To avoid starvation while still maintaining efficiency, use a RR schedule. Scheduling issues outside of just data packets in computer networks can be solved with RR scheduling. It's the idea behind an OS. The algorithm is called "round robin" because it follows the same principle used in other areas, where everyone gets a turn at doing something. TheRR scheduler uses time-sharing to schedule processes fairly; jobs are allotted a certain amount of CPU time, or "quantum," and are terminated if they haven't finished in that time. Whenever that process's allotted time comes around again, the job picks back up where it left off. The scheduler will choose the first process in the ready queue to run if the process terminates or transitions to the waiting state within its assigned time quantum. A process that produces large tasks would be favored over others if time-sharing was not an alternative or if the quanta were large in comparison to the job sizes. **Fig 6** presents the features of the RR algorithm.

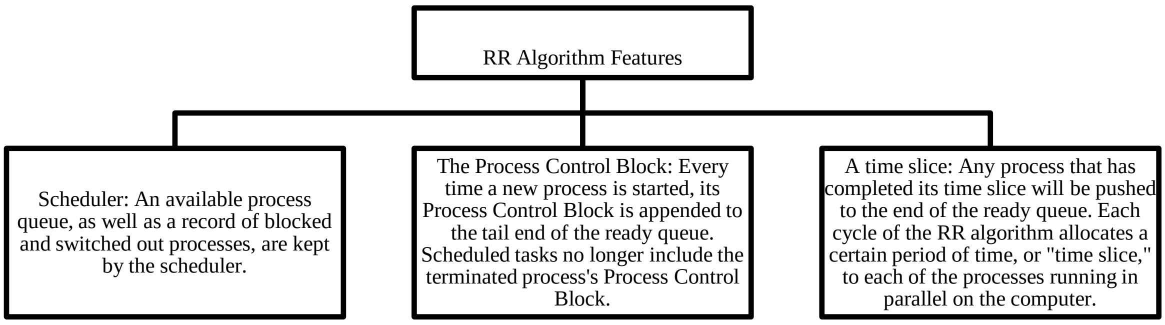
RR Algorithm Features
Scheduler: An available process queue, as well as a record of blocked and switched out processes, are kept by the scheduler.
The Process Control Block: Every time a new process is started, its Process Control Block is appended to the tail end of the ready queue. Scheduled tasks no longer include the terminated process's Process Control Block.
A time slice: Any process that has completed its time slice will be pushed to the end of the ready queue. Each cycle of the RR algorithm allocates a certain period of time, or "time slice," to each of the processes running in parallel on the computer.

Fig 6. Features of the RR Algorithm

When compared to other non-preemptive schedulers, RR algorithm enhances due to the dramatic reduction in usual response times it provides. By assigning a time limit to each job, the operating system ensures that it will be able to cycle through all available tasks, allowing each one an opportunity to execute. By dividing the quantum time by the number of activities in the run queue, RR scheduling maximizes the throughput of interactive jobs. This algorithm's behavior closely resembles that of first-come, first-served scheduling if the photon is extremely large, since it is quite probable that a process may stall or finish before the terms of validity is up. With such a modest quantum, the system can rapidly cycle through operations. This works wonderfully for collaborative endeavors. Unfortunately, there is a cost to context switching, and the proportion of system time spent on it rather than actual work rises as the frequency with which it is required rises. The features of the RR Algorithm have been depicted in **Fig 6**.

Throttled Algorithm

One of the DLB methods that use throttled adjustable algorithmsis known as the throttled load balancing algorithm. This approach has the data center controller sending requests from users straight to a virtual machine load balancer that has its throughput limited. When a Virtual Machine (VM) is needed, the load balancer now verifies its availability by transmitting its ID to the master controller of the data centers. Generally, the throttled load balancer deals with numerous functions, e.g., reading the present condition of VM and linking it with the data center controllers to provide an identity of the prevailing or underused VMs. By producing the task overhead on a similar load balancer, all of these load balancing duties and other specialized activities are being performed on a single component. In order to provide efficient service, the load balancer's responsibilities should be distributed among many parts. This would reduce duplication and unneeded complexity. The database that controls the important VM statuses is likewise managed by the throttled load balancer. The load balancer always gives the VM ID of the first one it finds in the status table. When a request is made for a VM, the data center controller will only relay the most important VM lists from the available VM lists database.

Modifications to the TAwere made by Al-Maliki [11]for use in cloud-based load balancing. In this work, a new, improved version of the TA for cloud-based load balancing is presented. This algorithm is designed to intelligently allocate incoming workloads to idle VM. This technique keeps track of both the index table of VMs and the current VM state. A faster reaction time and more efficient resource use are two goals pursued by the updated TA. If a VM is available and meets the client's requirements, its id is sent to the data center, and if not, -1 is sent back. As soon as a request comes in, the next item in the index is retrieved. The VM is chosen, and the same action is requested again, maybe with different results, based on the VM's current state. The only variation between the modified and basic TA is that when dealing with an index table, the basic method always starts with the first index. **Fig 7** below depicts the features of the TA.

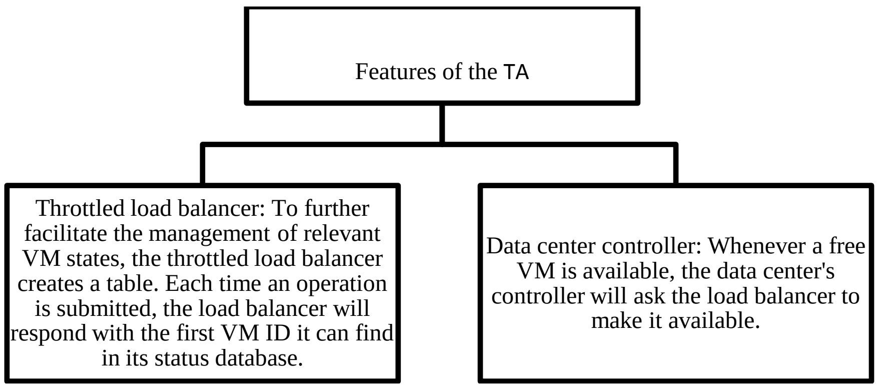
Features of the TA
Throttled load balancer: To further facilitate the management of relevant VM states, the throttled load balancer creates a table. Each time an operation is submitted, the load balancer will respond with the first VM ID it can find in its status database.
Data center controller: Whenever a free VM is available, the data center's controller will ask the load balancer to make it available.

Fig 7. Features of the TA

Artificial Bee Colony Algorithm Minimal Migration Time (BEE- MMT)

In 2013, Lian, Zaifeng, Guangfei, and Yi [12]introduced the BEE- MMT technique based on energy usage or CPU utilization PerformanceDegradation due to Migration (PDM) and the number of Virtual Machine Migration. The overloaded hosts are identified with the use of the Artificial Bee Colony (ABC) Algorithm, and then the MMT method is used to move one or more VMs off of those hosts. It also looks for hosts that aren't too busy and puts all the VMs assigned to them into hibernation if feasible.

When a host is identified as being in an overutilized state, one or more VMs are moved to less taxed hosts without taking the availability of the remaining hosts into account. The Minimum Migration Time (MMT) strategy is implemented to choose VMs from underutilized hosts for migration. Employing the Adaptive Neuro Fuzzy Inference System (ANFIS) and MMT policy, it is feasible to move VMs from over-utilized hosts to other hosts, hence reducing energy expenditures. Furthermore, a new cryptographic approach, elliptic curve encryption for data encryption, dramatically improves cloud security and data storage. MMT components have been depicted in **Fig 8**.

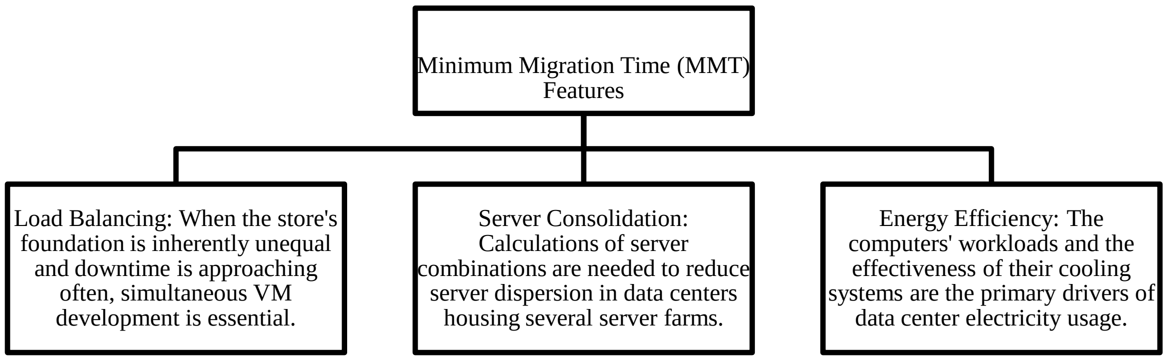
Minimum Migration Time (MMT) Features
Load Balancing: When the store's foundation is inherently unequal and downtime is approaching often, simultaneous VM development is essential.
Server Consolidation: Calculations of server combinations are needed to reduce server dispersion in data centers housing several server farms.
Energy Efficiency: The computers' workloads and the effectiveness of their cooling systems are the primary drivers of data center electricity usage.

Fig 8. Features of the MMT

Load Balance Improved Min-Min Scheduling Algorithm (LBIMM)

In order to perform load balancing in the cloud, Chen, Zhou, and Shi [13] recommended using a min-min scheduling method where user priority is used as a guiding principle. LBIMM Algorithm is a novel load balancing method built on the Min-Min algorithm that was developed by the author of this work to reduce the make span and increase resource utilization. Since the present workload is not considered in the traditional Min-Min approach, certain resources may be in continual usage while others lie idle. Priority Aware Load Balance Improved Min-Min Scheduling Algorithm (PA-LBIMM) is an upgraded algorithm that prioritizes user needs. It separates the work into two categories: those of greater and lesser importance. When it comes time to get things done, higher-priority tasks will run first, followed by the planned lower-priority ones.

Research on algorithms for load-balanced scheduling of cloud-based computing workloads has been ongoing for some time. Many methods for scheduling activities have been given in the corresponding literature. Thapliyal and PDimri [14]conducted in-depth analysis of the preexisting Min-Min algorithm and utilized it as a jumping-off point for developing the LBIMM and the PA-LBIMM. As a bonus, a simulated setting is provided for doing this analysis. Li, Mao, Xiao, and Zhuang [15]offer an enhanced Max-Min task-scheduling technique for a dynamical cloud, which maintains a task position table to calculate the real-time demand of virtual servers and predicted job completion time, is capable of distributing work among nodes and comprehend load balancing, and, as a result, improves resource usage and reduces task reaction times.

Assigning priority to the VMs and delivering the prioritized inputs to the GA might enhance load balancing and decrease response time in cloud computing, according to study by Shakeel and Alam [16]. They prioritized the VMs using a technique based on a logarithmic least squares matrix. Zheng, Wang, and Cai [17]describe a certain sort of work scheduling algorithm created by Rajput and Kushwah [18] known as the Improved Load balanced Min-Min (ILBMM). The ELBMM algorithm shortens maketime and improves resource utilization. This suggested algorithm consists of two parts. First, the predefined Min-Min algorithm is implemented, and therefore activities are rescheduled such that idle resources are put to good use.

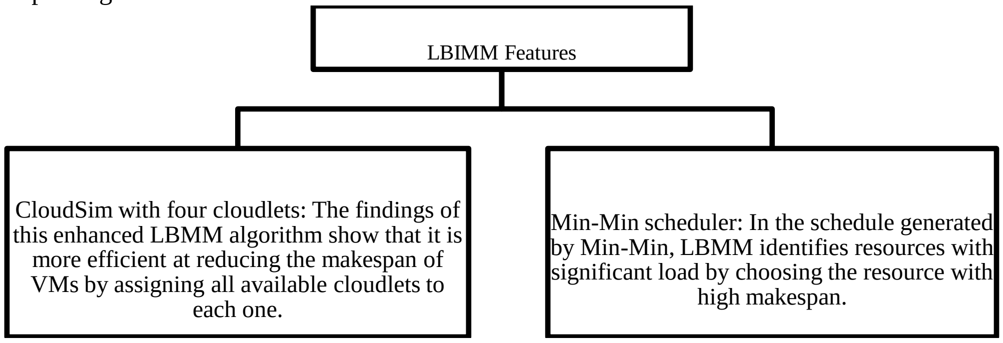
LBIMM Features
CloudSim with four cloudlets: The findings of this enhanced LBMM algorithm show that it is more efficient at reducing the makespan of VMs by assigning all available cloudlets to each one.
Min-Min scheduler: In the schedule generated by Min-Min, LBMM identifies resources with significant load by choosing the resource with high makespan.

Fig 9. Features of the LBIMM

Using a dynamic approach, Pourghaffari, Barari, and SedighianKashi [19] suggested allocating cloud resources to users. The Skewness Algorithm is used to evaluate the disproportional load distribution across VMs and to compute skew values accordingly. Combining resources of different types is simplified and resource efficiency is improved. Improved server-wide effectiveness in terms of accessible server resources is made possible by this repository's analysis of current and forecasted demand. Sen, Dey, and Bag [20] developed the VM-assign technique to assign user request to the closest available VMs. Further, the suggested method vastly enhances VM underutilization. The method "estimated the completion time load balancer" described in Schnor, Petri, and Langendörfer [21] is an improvement on traditional load balancing that guesses when a job will be finished processing, allowing cloud service providers to increase performance and make better use of their data center's VMs. Features of the LBIMM have been depicted in **Fig 9**.

Comparative Study

Table 4. Comparative study of various algorithms

Algorithms	Hierarchical Ecosystem	Distributed Ecosystem	Centralized Ecosystem	Dynamic Ecosystem	Static Ecosystem
Active Clustering	-	-	-	✓	-
OLB	-	✓	✓	-	✓
SHC	✓	-	-	✓	-
Max-Min	-	-	✓	-	✓
Min-Min	-	-	✓	-	✓
GA	-	-	✓	✓	-
ACO	-	✓	-	✓	-
RR	-	-	✓	-	✓

Table 4 above defines and contrasts a number of different types of ecosystems in order to better understand them. Earlier implementations of the techniques stated above for load balancing within the cloud are discussed. Many of them are revised and improved variants of algorithms that have already been described or found. There is more work to be done on enhancing some algorithms.

IV. CONCLUSION

The objective of load balancing in the cloud is to prevent any one server or computer from being overworked, underworked, or idle. Generally, cloud performance may be enhanced by using load balancing to optimize limited resources including execution time, response time, and system stability. Cloud computing's approach to load balancing involves deploying a load balance between servers and clients to control network traffic. To enhance the efficacy and dependability of cloud applications, load balancing in cloud computing distributes workloads, computing resources, and traffic uniformly in the entire cloud system. Through the use of load balancing within the cloud, businesses are able to control how their clients' requests are distributed among a collection of servers. Throughout this article, we have covered a number of different soft computing algorithms that are employed for load balancing within the cloud but have a lot of room for further exploration. The article also covers a comparison of algorithms depending on the context in which they run. Since system saturation may lead to subpar performance, load balancing is a critical concern in cloud computing. Due to this, there is still room for improvement in cloud load balancing. In this research, we explore the application of several load-balancing techniques, which utilize a soft computing technique within the cloud computing setting. Moreover, cloud sim may be used to run a simulation and investigate the graphical representation and outcomes.

Data Availability
No data were used to support this study.

Conflicts of Interests
The author(s) declare(s) that they have no conflicts of interest.

Funding
No funding was received to assist with the preparation of this manuscript.

Ethics Approval and Consent to Participate
The research has consent for Ethical Approval and Consent to participate.

Competing Interests
There are no competing interests.

References

[1]. N. Phumchusri and P. Amornvetchayakul, “Machine learning models for predicting customer churn: A case study in a software-as-a-service inventory management company,” Int. j. bus. intell. data min., vol. 1, no. 1, p. 1, 2024.
[2]. F. Wulf, M. Westner, and S. Strahringer, “We have a platform, but nobody builds on it – what influences Platform-as-a-Service post-adoption?,” Int. j. inf. syst. proj. manag., vol. 10, no. 1, pp. 49–70, 2022.
[3]. A. El-Deeb, “The holy grail of software products success: Great Customer Experience and the key elements needed to create one,” Softw. Eng. Notes, vol. 47, no. 2, pp. 8–9, 2022.
[4]. H. Singh and S. Kumar, “Dispatcher based dynamic load balancing on web server system,” Int. J. Syst. Dyn. Appl., vol. 1, no. 2, pp. 15–27, 2012.
[5]. V. W. Saputra et al., “An efficient load balancing using genetic algorithm in cloud computing,” in 2022 11th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), 2022.
[6]. A. Ahmid, T.-M. Dao, and N. Van Le, “Enhanced Hyper-Cube Framework Ant Colony Optimization for combinatorial optimization problems,” Algorithms, vol. 14, no. 10, p. 286, 2021.
[7]. S. Afzal and G. Kavitha, “Load balancing in cloud computing – A hierarchical taxonomical classification,” J. Cloud Comput. Adv. Syst. Appl., vol. 8, no. 1, 2019.
[8]. R. Stubbs, K. Wilson, and S. Rostami, “Hyper-parameter optimisation by restrained stochastic hill climbing,” in Advances in Intelligent Systems and Computing, Cham: Springer International Publishing, 2020, pp. 189–200.
[9]. S. Kumar, A. Kumar, V. Bajaj, and G. K. Singh, “A compact fuzzy min max network with novel trimming strategy for pattern classification,” Knowl. Based Syst., vol. 246, no. 108620, p. 108620, 2022.
[10]. V. K. Prasad, “Optimized Load Balancing using adaptive algorithm in cloud computing with round robin technique,” Int. J. Res. Appl. Sci. Eng. Technol., vol. 10, no. 7, pp. 134–149, 2022.
[11]. S. Q. A.-K. Al-Maliki, “Efficient cloud-based resource sharing through multi-tenancy and load balancing: An exploration of higher education and digital libraries,” Research Square, 2022.
[12]. L. Lian, F. Zaifeng, Y. Guangfei, and H. Yi, “Hybrid artificial bee colony algorithm with Differential Evolution and free search for numerical function optimization,” Int. J. Artif. Intell. Tools, vol. 25, no. 04, p. 1650020, 2016.
[13]. F. Chen, X. Zhou, and C. Shi, “The container scheduling method based on the min-min in edge computing,” in Proceedings of the 2019 4th International Conference on Big Data and Computing - ICBDC 2019, 2019.
[14]. N. Thapliyal and P. Dimri, “Load balancing in cloud computing based on honey bee foraging behavior and load balance min-min scheduling algorithm,” International Journal of Electrical and Electronics Research, vol. 10, no. 1, pp. 1–6, 2022.
[15]. X. Li, Y. Mao, X. Xiao, and Y. Zhuang, “An improved max-min task-scheduling algorithm for elastic cloud,” in 2014 International Symposium on Computer, Consumer and Control, 2014.
[16]. H. Shakeel and M. Alam, “Load balancing approaches in cloud and fog computing environments: A framework, classification, and systematic review,” Int. j. cloud appl. comput., vol. 12, no. 1, pp. 1–24, 2022.
[17]. T. Zheng, J. Wang, and Y. Cai, “Parallel hybrid particle swarm algorithm for workshop scheduling based on Spark,” Algorithms, vol. 14, no. 9, p. 262, 2021.
[18]. S. S. Rajput and V. S. Kushwah, “A genetic based improved load balanced min-min task scheduling algorithm for load balancing in cloud computing,” in 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN), 2016.

[19]. A. Pourghaffari, M. Barari, and S. SedighianKashi, “An efficient method for allocating resources in a cloud computing environment with a load balancing approach,” Concurr. Comput., p. e5285, 2019.

[20]. S. K. Sen, S. Dey, and R. Bag, “Study of energy efficient algorithms for cloud computing based on virtual machine migration techniques,” Int. j. mach. learn. networkedcollab. eng., vol. 03, no. 02, pp. 93–101, 2019.

[21]. B. Schnor, S. Petri, and H. Langendörfer, “Load management for load balancing on heterogeneous platforms: A comparison of traditional and neural network based approaches,” in Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 615–620.