RESEARCH ARTICLE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
www.rspsciencehub.com
Vol. 07, Issue 09 September

Check for updates

http://dx.doi.org/10.47392/IRJASH.2025.084

Balancing the Load in a Multi-Paradigm Era: A Comprehensive Survey of Algorithms from Cloud to Quantum

Suma Surasingh[1], Dr. Pradeep S[2], Shravya G[3], Deepika P S[4]

[1]Research Scholar, Dept of Computing Technologies, SRM Institute of Science and Technology Kattankulathur, Chennai, India.

[2]Associate Professor, Dept. of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur, Chennai, India.

[3]UG Student, Dept of CSE, SRM Valliammai Engineering College, Chennai- 603 203, India.

[4]UG Student, Dept of Manufacturing Engineering, Anna University-CEG, Guindy, Chennai, India.

Email ID: *ss1399@srmist.edu.in[1], pradeeps1@srmist.edu.in[2], shravyagadupuri@gmail.com[3], deepika21745@gmail.com[4]*

Article history

Received: 13 August 2025
Accepted: 27 August 2025
Published:04 September 2025

Keywords:
Cloud computing, Edge computing, Load balancing algorithms, Quantum computing.

Abstract

Load balancing is a pivotal facet of distributed computing systems that significantly influences the performance of the system by achieving optimal resource utilization, reduced response time, and enhanced system reliability through the even distribution of workloads to computing nodes. The rate at which the computing paradigms are changing has resulted in the increased complexity of the load balancing problem, which is evident in the cloud, fog, edge, grid, and quantum computing since they are particularly characterized by the issues of latency sensitivity, energy efficiency, heterogeneity, scalability, and quantum decoherence. This paper surveys load balancing algorithms implemented in the five computing paradigms, highlighting the main operating principles, architectural differences, scheduling strategies, and performance evaluation criteria applicable to each context. The cloud computing section is devoted to the classification of static and dynamic algorithms, and we also touch on the weighted round-robin, honeybee foraging, and VM migration strategies. The paper also surveys the load balancing of the fog and edge computing wherein we pinpoint the latency-aware and mobility-aware load balancing strategies that are best for the resource-limited, geographically distributed infrastructures. In the grid computing section, peer-to-peer, and decentralized scheduling methods are discussed, which are best suited to loosely coupled networks. Quantum computing is also discussed in the paper as an early stage of load balancing in hybrid classical-quantum systems, the partitioning of quantum jobs, and the coherence limitation of qubit. In a unified taxonomy of algorithms, the paper describes how various algorithms can be mapped depending on their design philosophy, decision criteria, and applicability to different paradigms. Further, the paper lists evaluation charters that are widely utilized in the literature and underscores open issues like cross-paradigm interoperability,

scheduling with context awareness, and the trade-offs between energy and performance. By bringing together understanding from these different places, this survey acts as a base for researchers and practitioners who want to create load balancing strategies that are not only adaptable but also paradigm-specific, and cross-platform, and that fit into the changing face of distributed computing.

1. Introduction

Computing systems in the digital landscape have started to operate at a higher level of efficiency, minimal latency, and absolute reliability, together with processing data of an increasingly higher volume while at the same time servicing user demands that are growing rapidly. The rapid rise in data- centric apps, cloud-based services, and real-time decision- making has put a huge strain on infrastructure providers, who are now forced to optimize performance while still being cost-effective and scalable. Load balancing in this venue turn out to be a primary role of distributed computing architectures. It allows for the smart and fair allocation of the computational load amongst various resources such as physical servers, virtual machines, or decentralized nodes, thus avoiding resource bottlenecks, guaranteeing high availability, and improving system responsiveness overall [1]. Load balancing has gone far beyond merely performance tuning in ensuring system resilience during too heavy or unpredictable workloads, it also reduces downtime and increases user satisfaction. The development of distributed computing paradigms has led to a great increase in complexity to accomplish a satisfactory load balancing. This is most notably illustrated in various modern computing ecosystems such as cloud, edge, fog, grid, and quantum, every one of them bringing along the whole new set of their own constraints, for instance, sensitivity, energy consumption, device heterogeneity, and for quantum systems, they face the issue of decoherence to name a few. The primary focus of traditional load balancing algorithms is centralized environments, and extensive research has been conducted on them; however, the new decentralized and hybrid architectures' applicability to these algorithms is still a matter of ongoing research. New paradigms require flexible and context-aware algorithms that can function economically under changes in network topologies and resource conditions. It follows that a multi-faceted approach is necessary for the evaluation of load balancing strategies, one that not only takes scheduling decisions into account, but also architectural dependencies, mobility support, and scalability factors. This survey intends to offer a comprehensive bulletin covering the most important load balancing techniques implemented over five notable computing paradigms: cloud, edge, fog, grid, and quantum computing. It lays out a comprehensive categorization of algorithms, delves their mechanisms and talks about design ideologies and measurement of decisions. The article further carves out a consolidated classification of an algorithm for the purpose of comparison and it also points out issues that have been there all along like problems in interoperability across layers, the trade-off between energy and performance as well as scheduling that is dependent on the context. We thus aspire to provide assistance to scientists and professionals through this research in not only conceiving and applying but also adjusting and personalizing load balancing strategies for distributed computing in future which is inherently more diverse [2] a key factor in cloud environments because it manages the fair distribution of customer requests among several servers with the goal of increasing system throughput performance, reducing latency, and guaranteeing high availability. Various algorithms are used in cloud-based load balancing, and each one has its own advantages and disadvantages depending on the infrastructure and workload features. a key factor in cloud environments because it manages the fair distribution of customer requests among several servers with the goal of increasing system throughput Figure 1 shows Cloud Computing Reference Model

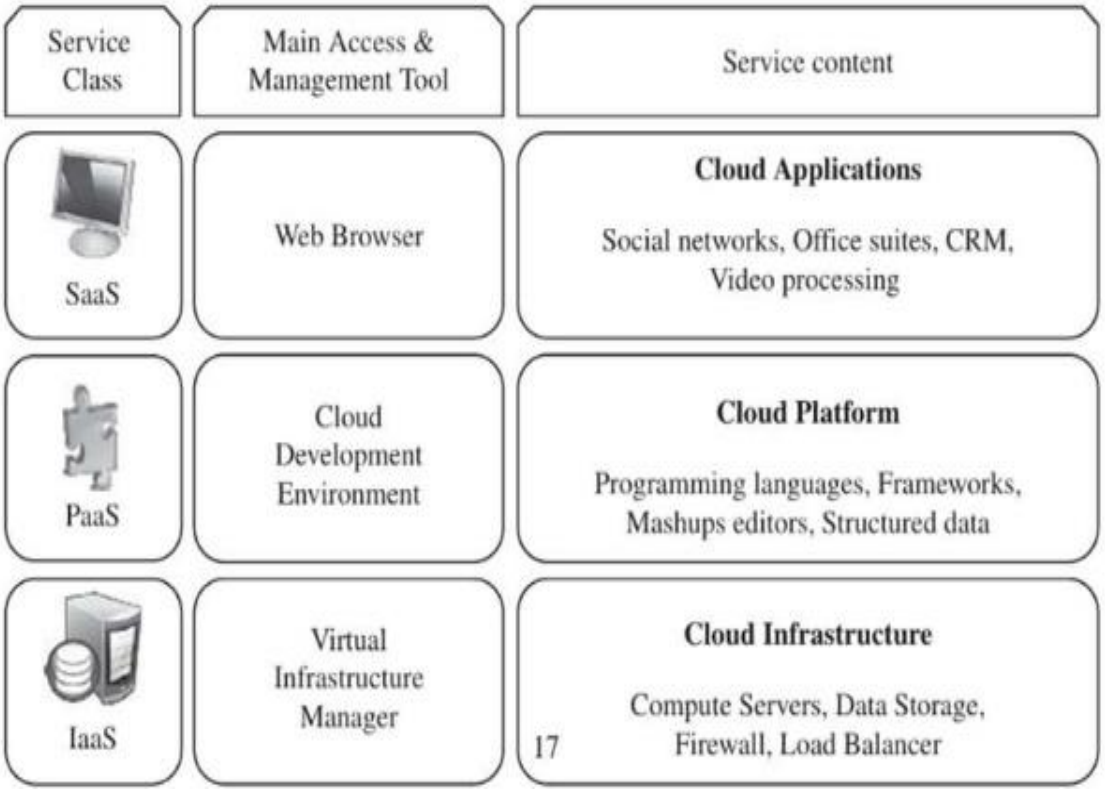

Figure 1 Cloud Computing Reference Model

2. Relevance of Load Balancing

In the realm of contemporary distributed computing environments, load balancing has become an indispensable instrument for preserving the integrity of the particular services of the web. As the number of users increases and applications require more current responsiveness, computational infrastructure has to be able to continue to meet performance standards. Load balancing solves this problem by intelligently allocating workloads to the resources available, for example, virtual machines, physical servers, or edge nodes, so that the resources are not overused, the response time is lowered, and the throughput is substantially more improved. For example, in cloud computing where multitenancy and on- demand resource provisioning are standard practices, load balancing guarantees that resources are distributed evenly among several servers or clusters and thus, it prevents the overloading of a particular server or cluster. Besides, this not only allows the scaling of the system but also assures it that the highest availability will be maintained and the service will continue uninterrupted even during unanticipated traffic peaks or failures in hardware. And also in edge and fog computing instances, where the resources become more dispersed and they are often scarce, efficient load distribution is of paramount importance to giving services to users without delay and also keeping the real-time processing capability intact. Additionally, containerization and microservices as well as hybrid deployment models got the more detailed way that load balancing has to operate so that it could make the right decisions. When balancing is not good, the performance of the backend services will be worse, the user experience will be bad, and the infrastructure will be more susceptible to a chain reaction of failures. Hence, effective load balancing is very crucial for the system to be reliable, the users to be satisfied, and it will also become a factor in minimizing the operational costs [3]. Boosted dependence on electronic means—covering the areas of electronic commerce, cloud storage, live health monitoring, and intelligent traffic systems—has transformed load balancing from just a performance enhancer to a business-critical function. This in turn makes the study of load balancing algorithms over different computing paradigms not only relevant but also necessary for the creation of adaptive, robust distributed systems of the future [4]. Figure 2 shows Cloud Storage.

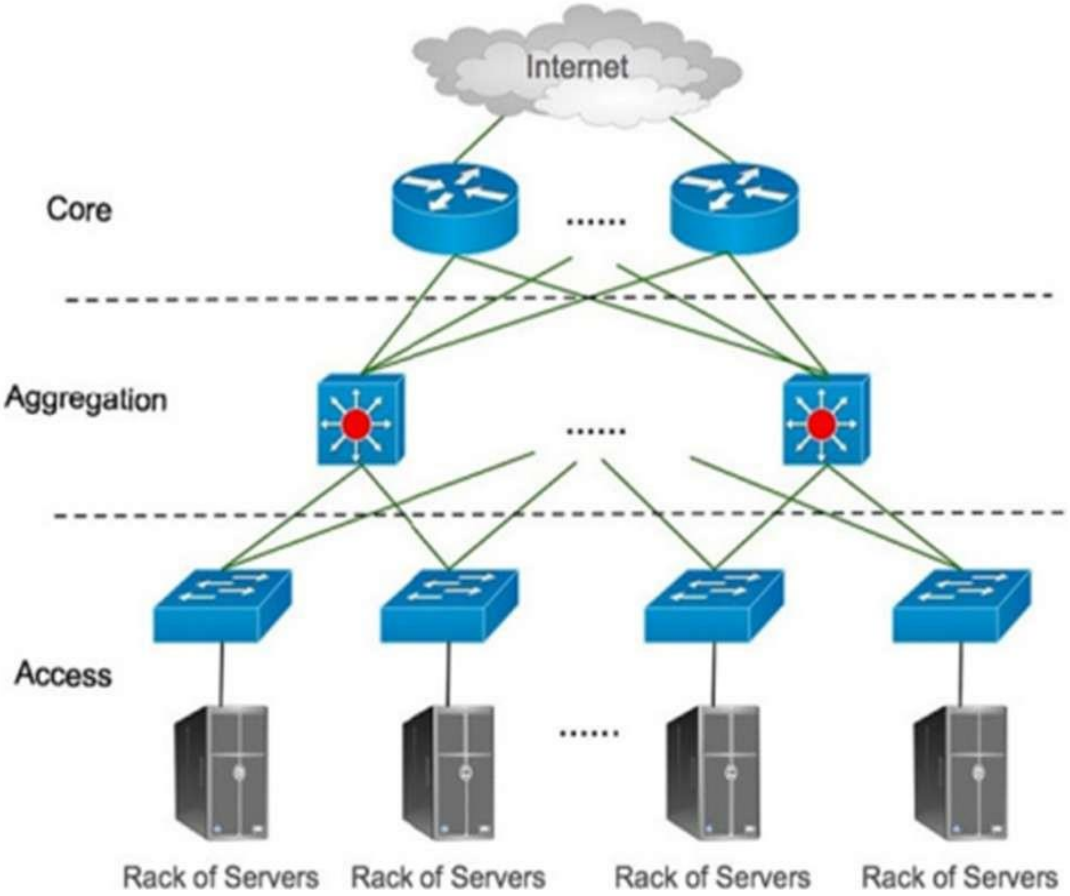

Figure 2 Cloud Storage

3. Evolution of Load Balancing Strategies

Load balancing approaches have significantly changed over time which clearly shows the changes in the computing world from centralized mainframes to dramatically distributed and hybrid infrastructures. Initially, load balancing was a manual operation that was mostly done by system administrators using very simple techniques like static configurations or DNS-based round-robin methods. These initial techniques were very basic and operated with limited context-awareness. They were very effective only in small-scale, mainly static environments [5]. A big increase in data and user interactions on the internet has made these strategies insufficient very easily and they were not able to maintain system responsiveness and reliability. When client-server architectures came about and dynamic web applications became popular in the late 1990s and early 2000s, they introduced more advanced software and hardware-based load balancers. These load balancers were capable of dynamic resource allocation, running health checks, session persistence, and SSL termination, which in turn helped both performance and fault tolerance. But the fact that these strategies were centralized still limited them in terms of scalability and resilience [6]. The emergence of cloud computing represented a significant milestone in the design of load balancing strategies. Virtualization facilitated hardware abstraction and the dynamic allocation of resources. Load balancing at hypervisor level became the most popular method, as it allowed the distribution of workloads across virtual machines based on real-time resource utilization metrics such as CPU, memory, and network I/O [7]. Cloud-native solutions, such as Amazon ELB, Microsoft Azure Load Balancer, and Kubernetes Ingress Controllers, started to add policies that could auto-scale according to thresholds, thus increasing both the elasticity and cost-efficiency. Load balancing evolved to fit the new conditions as the computing has reached fog and edge nodes besides data centers. In fog computing, which is located more in the network's edge, load balancing strategies were latency-aware, mobility-aware, and energy-efficient, because of the

geographical distribution and limited resources of fog nodes [8]. Likewise, edge computing environments provided a necessity for real-time decision-making with the least dependence on central controllers. From here, decentralized and hierarchical load balancing algorithms came into being, which could operate by themselves, be receptive to mobility and connectivity, and trust their own fault tolerance even in the case of intermittent network conditions [9]. At the same time, with the growth of big data and the increase of AI workloads, special load balancing mechanisms were necessary for distributed processing frameworks such as Hadoop, Spark, and TensorFlow. These systems not only required a balanced task execution but also had to be aware of data locality, network topology, and the heterogeneity of computing nodes [10]. The quantum computing field has opened up new opportunities for load balancing scientists. Unlike classical systems, quantum computing uses qubits that are highly vulnerable to decoherence and have only a few physical resources available. Hence, load balancing should solve problems resulting from the limited circuit depth, connectivity of qubits, and the combining of classical and quantum parts during orchestration. Research in the initial stages of productivity is relating to task division between classical and quantum components, job scheduling for quantum hardware, and also necessitating implementing of error during the experiment [11]. Considering that quantum computers are moving from the NISQ devices to fault-tolerant ones, the matter of load balancing will greatly influence both the production rate and resource utilization to be efficient. The progression of load balancing techniques has reflected a shift to greater decentralization, environmental sensitivity, and targeted efficiency. The advent of novel computing paradigms such as serverless, federated learning, and quantum-cloud integrations has only served to broaden the reach of load balancing. Here, load balancing will have to go beyond being just reliable, it will have to incorporate more flexible, smarter, and compatible agents capable of performing seamlessly across the diverse and mixed systems.

4. Load Balancers in Cloud Environments

With the cloud computing paradigm, the provisioning of computational resources has changed beyond recognition through the use of scalable and on-demand services. Load balancing is a key factor in cloud environments because it manages the fair distribution of customer requests among several servers with the goal of increasing system throughput performance, reducing latency, and guaranteeing high availability. Various algorithms are used in cloud-based load balancing, and each one has its own advantages and disadvantages depending on the infrastructure and workload features. Figure 3 shows Cloud Simulator

4.1. Round Robin Algorithm

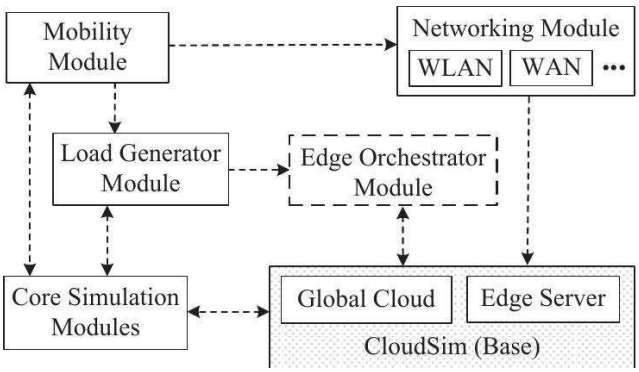

Figure 3 Cloud Simulator

One of the earliest and simplest algorithms that Round Robin was used for load balancing because of its deterministic character and easy implementation. It distributes requests to a series of servers in a round-robin fashion. After the last server in the list receives a request, the process is repeated for the first server and so on. This cyclic nature of the process guarantees that over time, each server will receive an equal number of incoming requests. On the other hand, Round Robin considers that all servers are of equal capacity and the load generated by the requests is of equal weight, which rarely happens in actual cloud environments. Differences in server processing power, current load, or task complexity can lead to imbalanced distribution and longer queue times for certain servers [12]. Due to its simplicity, Round Robin may, however, become ineffective in heterogeneous environments where server capabilities and workloads differ significantly [13].

4.2. Least Connections Algorithm

The Least Connections algorithm provides a more dynamic approach by taking into account the load on each server in real- time. It does not distribute requests randomly but routes each new request to a server that has the smallest number of active

connections at that moment. This is especially suitable to instances with continuous sessions as in video streaming, database transactions, or stateful web sessions. This flexible trait allows for a more balanced load and increases response times, particularly in cases where there are different durations for processing requests. The algorithm, though, still needs to keep track of all open connections, which may lead to extra work being done [14]. This approach is more efficient in systems with uneven workloads and also helps scalability by not overloading any single node. [15] Figure 4 shows Load transfer strategy

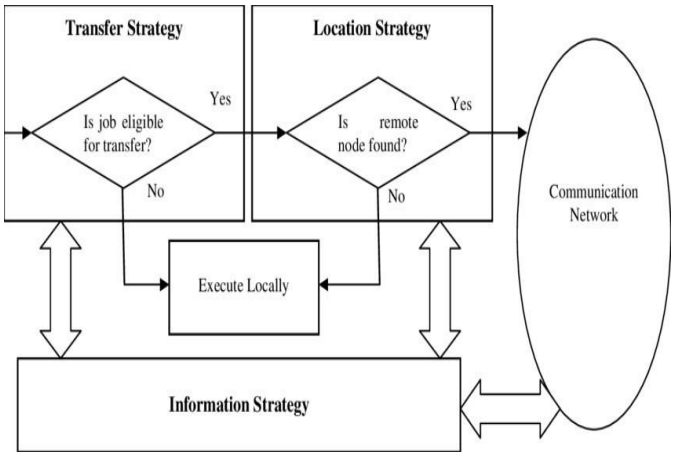

Figure 4 Load Transfer Strategy

4.3. Weighted Round Robin

Weighted Round Robin (WRR) takes the basic Round Robin protocol a step further by incorporating weights which enable the non-uniform distribution based on the server's abilities. Servers are given a weight value which is proportional to their computing power (CPU, RAM, network bandwidth). The algorithm's main idea is that more powerful servers are given a bigger portion of the incoming requests, thus resource usage is consistent with server capacity. Thus, for instance, a server with a weight of 3 will get three times as many requests as a server with a weight of 1. The WRR algorithm walks the server list over and over while going each server as many times as its weight, resulting in proportional load distribution in heterogeneous environments [16]. Such a method is the most efficient for the case of cloud platforms with heterogeneous server instances and it also gives a guarantee for better performance by decreasing the response time and not causing the server to be overloaded [17]. The proportional distribution of requests in Weighted Round Robin can be mathematically expressed as:

$$R_i = (W_i / \Sigma W_j) \times T$$

Where:

- R_i = Number of requests assigned to server i
- W_i = Weight of server i
- ΣW_j = Sum of weights of all n servers (j = 1 to n)

T = Total number of incoming requests in one cycle This equation ensures that servers with higher capacities receive a proportionally greater share of the workload, thus optimizing system performance in heterogeneous cloud environments [18].

4.4. Load Balancers in Edge Computing

With the rapid shift of computing power from a data center to a data source for ultra-low latency and real-time processing, edge computing has become a paradigm-shift mechanism in the most recent distributed systems. The edge environment, which is at the opposite end of the spectrum from centralized cloud infrastructures, carries features like decentralization, spatial dispersion, and resource limitations. The nature of these systems is latency-sensitive applications such as real-time video analytics, autonomous vehicles, industrial IoT, and wearable health monitors. So in this case, the role of load balancing is essential not only for safeguarding the fair distribution of resources but also for providing the necessary service continuity in the case of user mobility and heterogeneous edge node capabilities. The characteristics of edge environments that make them tiny, short-lived, and volatile mean that conventional methods of balancing load cannot meet the situations. Consequently, algorithms written for edge require load balancing, with the potential to carry out the real-time monitoring of the current situational conditions and decentralized decision-making, and to be able to respond rapidly to changes in the environment. Here, we explore the three most common load balancing protocols that are usually followed in edge ecosystems.

4.5. Edgecloudsim Dynamic Algorithm

EdgeCloudSim is the simulation toolkit built upon the CloudSim platform. This is a simulation environment primarily targeting the emulation of edge computing scenarios. Specifically, the integration of dynamic load balancing methods is one of its principal contributions that constantly

gather input from the environment such as server status, users movement, and network fluctuation throughout the process. In EdgeCloudSim's algorithmic model, the decisions are not fixed but rather changes are implemented dynamically while the mobile users are moving from one location to another or if the edge servers are unreachable for some time. The system redirects the requests to the best available nodes in real-time, thus not only eliminating service handoff delays but also extending the task execution without interruption and increasing the network's overall response speed. Such features are especially necessary in the cases of vehicular communication systems, healthcare monitoring, and mobile video surveillance, where a delay in the task allocation may cause the service to fail or important data to be lost [19].

4.6. Resource Aware Scheduling

Resource-Aware Scheduling (RAS) is a real-time metric that gets its data from the edge nodes and takes note of the CPU usage, free memory, and network throughput. However, RAS, which is a dynamic decision-making strategy, makes use of resource profiling of nodes periodically to pick the best candidate for a certain task, whereas static ones make only the initial assignment without subsequent evaluation. The scheduler always keeps track of node status and makes sure that it distributes the tasks that come in among those nodes, which can process them efficiently and do not cause resource overutilization or service bottlenecks due to the high workload. Such a scheme certainly guarantees that low-power or already heavily-loaded devices will not be part of the process and thus will not contribute to an increase in the failure rate, even though they still have some unused resources. RAS is a very efficient method in conditions in which the edge devices are of different types and the workload is changing in an unpredictable manner. In addition, it enables power saving, which is very important in battery-operated edge systems. Experimental works have demonstrated RAS is able to remarkably improve both Quality of Service (QoS) and system resilience while load changes occur [20]. Figure 5 shows Mobile Edge Computing

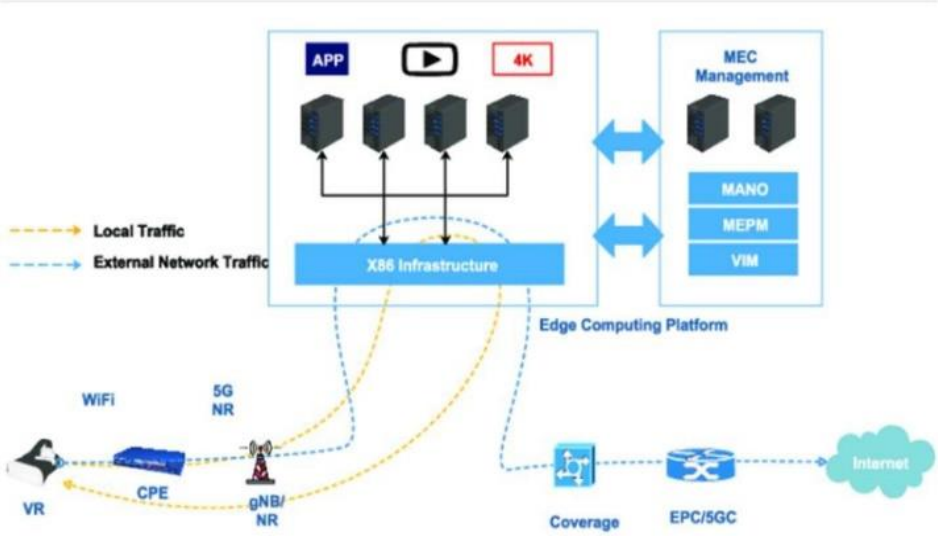

Figure 5 Mobile Edge Computing

4.7. Mec Proximity Algorithm

In the MEC realm, proximity is a major factor in load balancing thus the location and the distance to the user are the main factors that decide where the task is to be launched. The Proximity- Aware Load Balancing Algorithm in MEC, which is an algorithm loaded with the graph of the local state of resources of all machines in a cluster, primarily aims at routing tasks to the edge server that is physically closest to the user. This method outlines the basis for ultra-low-latency applications such as augmented reality, remote surgery, and autonomous vehicles where every millisecond counts. The proximity selection logic is not a purely distance-based one but it also takes into account network congestion and node workload which is done in the same way as performance improvement that is more holistic. The proximity function is generally given by the following expression:

$$P(u, e) = \alpha \cdot d(u, e) + \beta \cdot L(e)$$

Where:

- P (u, e) is the proximity cost of assigning user u to edge server e.
- D (u, e) represents the network distance (e.g., latency or hop count) between user u and server e.
- L(e) is the current load on edge server e, which may include processing, bandwidth, or memory usage.
- α and β are weighting coefficients to balance the importance of distance vs. load. [21]

This function ensures that the task is allocated to the most optimal edge node by balancing distance and real-time performance metrics. The proximity table is continuously updated to adapt to node failures or mobility events, maintaining system responsiveness and robustness. Figure 6 shows MEC Algorithm

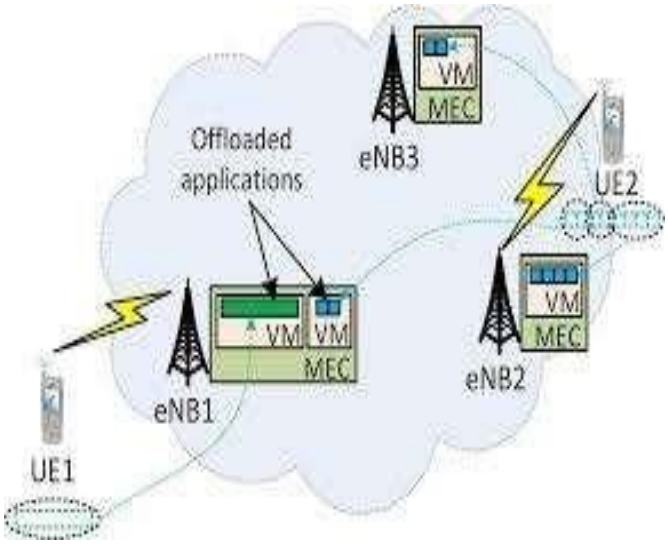

Figure 6 MEC Algorithm

5. Load Balancers in Fog Computing

Fog computing depicts a decentralized computing model that is intended to carry out and analyze data locally near the source of its generation, such as sensors, wearables, and embedded systems. In contrast to conventional cloud systems, fog environments are known by their geographic spread, low latency requirements, mobility, and resource restrictions. load balancing in such a case becomes a very important issue because of the diversity of fog nodes, limited processing and storage capacity, as well as the real-time nature of the applications they support. Since fog computing provides a link between the cloud and the edge, it has to manage the computational workloads in the most efficient way in order to keep the system responsive, minimize latency, and get rid of resource bottlenecks. Load balancing in fog computing environments needs context-aware, adaptive, and energy-saving strategies that are capable of responding to unpredictable changes in user requests, network dynamics, and node availability. Below are some of the most effective strategies employed in fog computing.

5.1. Adaptive Load Balancing (ALB)

Adaptive Load Balancing is a reactive and self-governing approach that reflects in real-time the alterations of workload distribution and node availability. ALB algorithms, which are dynamic, can redistribute tasks following network parameters that vary, e.g., CPU utilization, memory usage, bandwidth availability, and service latency, as opposed to static ones. This method is especially relevant in fog systems where nodes can appear and disappear without warning, or where the usage changes significantly during the day. Modern ALB techniques are blending predictive models to envision that resource congestion which has not yet happened but is on its way. The predictive models typically use past data and learning-based approaches like regression, time-series forecasting, and decision trees. In addition, the system keeps on improving its distribution of the load by making better decisions based on the current feedback. The efficiency of ALB has been proven in transportation which is smart, surveillance that is real-time, and automation of industries implementation examples [22], [23]. In addition, ALB aligns quite well with the concept of fog computing on the mobile. Properties of vehicular fog computing, for instance, include devices that are constantly moving, lack of leaders, and fluctuating connectivity as a result of mobility. Adaptability in situations like these is not only a plus but a requirement to guarantee the continuity of service and lower the number of tasks that are not accepted [24].

5.2. Fuzzy Logic-Based Scheduling

Conventional binary decision-making approaches come up short in fog environments owing to uncertainty in resource availability and workload predictability. Fuzzy logic-based scheduling is a probabilistic model that simulates human-like decisions, and it supports the making of more flexible balancing of load. In place of considering

input parameters as definitely high or low, fuzzy systems decide them on a continuum by employing linguistic variables like "lightly loaded," "moderately loaded," or "heavily loaded." Ultimately, the fuzzy logic system goes over those rules and then decides the optimal task-to-node assignment. For example, if we find some node whose CPU usage is "medium" and at the same time, its memory usage is "low," a scheduler might conclude that the node is free and thus can be given more tasks. Such an approach makes fog infrastructure scheduling much more intelligent in the situation when so many resources have to be assessed simultaneously, although the condition of those resources is not very certain and they are in a hurry. They have demonstrated remarkable success in cases such as tracking of circumstances in nature, health-related IoT, and the quick reaction of emergencies, because of the speed requirements and the lack of certainty in the operating conditions. Resources which are innately constrained in computation, coupled with fog nodes that do not rely on the heavy computational models, fit best with the ways these systems deal with imprecision of information perfectly.

5.3. Energy-Aware Load Balancing

Energy efficiency is becoming a focal point in fog computing, notably as fog nodes are often installed in settings with limited power such as remote locations, sensor networks, and urban IoT configurations. Energy-aware load balancing targets the efficient allocation of tasks that considers not just resource capacity but also the energy consumption characteristics of the nodes. The purpose of these algorithms is to achieve the lowest possible energy usage in the fog network and at the same time guarantee the best performance levels. They do so by assigning tasks preferentially to nodes that either are of low energy consumption or that are powered by renewable energy sources. Various methods adhere to the inclusion of the thermals and battery conditions of the nodes in the decision procedure so that the nodes get the correct level of energy usage and no energy is wasted. Besides this, energy-efficient balancing can result in savings that go beyond just energy. Besides extending the life of the appliances, it lowers running costs, and also makes a contribution to sustainability targets, which are indispensable for big IoT implementation such as smart cities, agricultural monitoring, and disaster management systems [26]. In a study reported recently, in addition to that, such strategies are combined with work capacity prediction prototypes so that tasks can be scheduled in advance during low-energy consumption times, or only a part of fog nodes can be activated during off-peak hours, thus still meeting QoS requirements and simultaneously saving energy. In the global shift towards environmentally friendly computing paradigms, energy-aware load balancing remains a crucial pathway for future fog computing developments [23], [26].

6. Load Balancers in Grid Computing

Grid computing is one of the models for distributed computing which aggregates the geographically distributed and heterogeneous resources—ranging from the processing units and the storage to the software services, to work as a single system for solving complicated scientific, academic, and industrial problems. These resources are really quite different and they may come from various administrative domains and could vary very much in availability, capability, and reliability. Consequently, effective load balancing in grid computing is indispensable to ensure optimal utilization, reduce job turnaround time, and maintain service reliability. A decentralized and intelligent scheduling approach, different from the centralized cloud perspectives that governs grid environments, is needed in grid systems. Such a mechanism would enable these systems to keep track and dynamically adjust during the changing conditions of load, node availability, and the intake of application requirements. Two of the most popular load balancing techniques that have been proven as effective in grid computing are those which employ Genetic Algorithm, Ant Colony Optimization, and Min-min Scheduling as the base. In addition to dealing with the diversity and spread of the grid resources, the algorithms also provide scalability, fault tolerance, as well as more efficient performance for all nodes.

6.1. Genetic Algorithm- Based Load Balancing

Genetic Algorithm (GA) is a class of stochastic search procedure which are inspired by natural selection and meiosis. GAs are well suited for tackling NP-complete problems efficiently if these problems can be translated to the form of scheduling

and resource allocation in grid infrastructure characteristics. The major principle of GA-based load balancing is to iteratively generate new populations of candidate solutions (chromosomes) in the hope of finding the minimum or maximum of an objective, usually corresponding to time, load variance, or cost if the algorithm is implemented as a minimization or maximization problem respectively. Here, each chromosome represents a possible configuration of the tasks assigned to the computational nodes. The algorithm then continues with the following genetic operations:

- **Selection:** Taking the fittest individuals (task-resource mappings) based on a certain fitness function, which can be made up of parameters like makespan (total time to execute all tasks), load balance index, or network delay.
- **Crossover:** Merging pairs of chromosomes in order to create new offspring that have the traits of both parents.
- **Mutation:** Giving offspring some random changes to keep genetic diversity and not get stuck at a local optimum.
- A popular fitness function is given by the equation:
- Fitness = 1 / Makespan

Here:

a) Makespan is the longest time of completion among all the resources.
b) We want to get the smallest makespan, which is equivalent to the highest fitness.

Genetic Algorithm (GA)-based scheduling has been proven to produce better outcomes than conventional heuristics. It is especially useful in large-scale grid systems where the search space for optimal solutions is enormous. This technique is especially suitable in coping with non-deterministic behavior, multi-objective constraints, and dynamic workloads [27], [28].

6.2. Ant Colony Optimization (ACO)

ACO is a population-based metaheuristic that aims to mimic social insects' behavior in finding shortest paths to food sources. The ants perform such tasks as searching and foraging out of exploration and exploitation of previously found routes. This behavior complicates the understanding of the task of grid load balancing, where tasks that arrive dynamically could be assigned simulatenously to the different nodes as we traverse along the various paths in the solution space, with each path to a certain redirecting of the tasks to the nodes.

The concept is that every ant takes a tiny part of the total effort sampling a task, choosing the current best decision, constructing a solution, and committing it incrementally.

- **Pheromone Trails:** Chemical signals left by ants showing them the direction to follow to find the optimal food source.
- **Heuristic Information:** Such as the size of the task, the CPU speed, the length of the queue, and the bandwidth.

Solutions that are good become more attractive to other ants over time, resulting in a kind of runway effect where new ants keep following the recent and best solution. ACO allows the nodes in the system to act independently and coordinate with each other as they work on the problem, which is beneficial in a grid computing environment where nodes might be difficult to reach, fail, or change their state without any warning. ACO has found effective use in scientific workflows, climate modeling, and genome analysis, where it has provided more equal load distribution, higher fault tolerance, and lower overall execution time than deterministic algorithms [29], [30]. Figure 7 shows Ant Colony Optimization

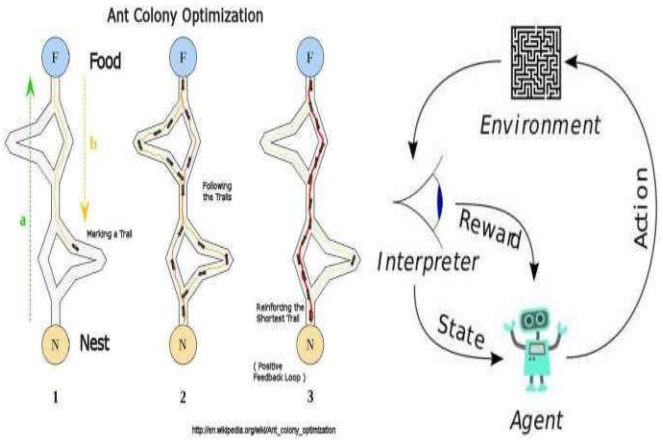

Figure 7 Ant Colony Optimization

6.3. Min-Min Scheduling Algorithm

The Min-Min algorithm is a deterministic scheduling heuristic that is characterized by its simplicity and speed. It operates by finding a task-resource pair such that the task with the minimum earliest completion time is scheduled first. The process can be formulated as follows:

- For every task that is not yet scheduled, find the earliest completion time (ECT) on every resource.

- Choose the task with the minimum of overall ECT.
- Give the task to the resource which corresponds to that and indicate the task as done.
- Do it again till all the tasks have been given.

The algorithm takes advantage of the greedy strategy of picking up small jobs first and thus it ensures that those resources which work faster are not blocked unnecessarily by big jobs. This is often beneficial since it yields higher throughput and better system utilization in environments where the major part of workload is composed of small to medium jobs. The Min-Min method caused the problem of starvation of the big tasks in some situations, but it is preferred by many people since it is easy to implement. It has minimal overhead and is suitable for time-sensitive grid applications such as multimedia rendering, medical diagnostics, and distributed data analysis [31], [32] because of its straightforwardness and low overhead.

7. Load Balancers in Quantum Computing

Quantum computing prominently showcases a new facet of computational science, where information is processed by qubits, quantum bits that utilize the principles of superposition, entanglement, and quantum tunneling. The latter features, make quantum computers exponentially more powerful compared to classical machines in particular areas like cryptography, optimization, and running simulations of complex systems. However, on the other side of the coin, resource management and task scheduling in quantum computing are demanding jobs, thus load balancing, a key factor, has to cope with not only quantum-specific limitations but also the synergy with classical systems. To load balancing in normal computing, algorithms are employed that can be either deterministic or probabilistic is traditional computing, whereas in quantum computing, native quantum scheduling models, which can utilize the non-classical behaviors of machines, are available to find solutions for resource allocation and work distribution problems in a more efficient manner. Quantum computing is on the rise cloud- based quantum processors (e.g., IBM Q, D-Wave, and Google Sycamore) will explode, practical and smart load balancing will be crucial in keeping the integrity of the performance, lowering communication latency, and optimizing quantum resource utilization.

7.1. Quantum Entanglement- Based Scheduling

Quantum entanglement is a really abnormal phenomenon when the states of not one or two but multiple qubits become so interdependent that if you change one it will affect the others immediately even if they are in different places. When it comes to distributed quantum computing, this feature is the one that is used to keep the task assignments and the synchronizing of the resources in the same state, as if the communication overhead were not there. In Quantum Entanglement-Based Scheduling, the task qubits get entangled with resource qubits. When a qubit is used for a task, the entangled partner qubit auto-magically reflects the same assignment, thus, the scheduling decision is communicated in an instant all over the quantum system without any delay. This method can be especially advantageous in quantum cloud systems where entangled qubits are being distributed among quantum nodes located at different places. Such entanglement-driven synchronization eliminates classical bottlenecks like:

- Delays due to message passing.
- Task queuing inconsistencies.
- Resource idling due to communication latency.

Though still mostly theoretical and hindered by the limitations of practical implementation (i.e., decoherence, error correction), this kind of load balancing may become crucial with quantum interconnects' advancement [33], [34].

7.2. Grover-Based Search Load Balancer

Grover's algorithm is a quantum search algorithm that is most famous for its ability to find an element in an unsorted database of N elements in about O(N), O(√N), O(N) time, thus providing a quadratic speedup over classical linear search methods. If we consider the load balancing scenario, the problem of assigning the best resource to a certain job can be seen as a search problem, where the scheduler needs to find an optimal match from a set of possible task-resource pairings. Grover's algorithm is intended to:

- Choose a node that satisfies given latency or capability requirements.
- Make scheduling decisions that are based on the cost function evaluated through quantum.

- Find the resource that has the least load.

This method comprises quantum state initialization, oracle evaluation (which marks the desirable resource), and amplitude amplification to enhance the probability of picking the best node. Only a few steps are involved when a quantum-based scheduler interdisciplinarity mergers with classical processing units since that enables the fast reduction of the best candidates for task execution [35], [36]. The load balancing system that is based on Grover is also scalable over cloud platforms. In fact, those who can provide quantum as a service (QaaS) will be able to avail themselves of this new technology to optimize the load at distributed quantum cores in real-time. Figure 8 shows Quantum Annealing

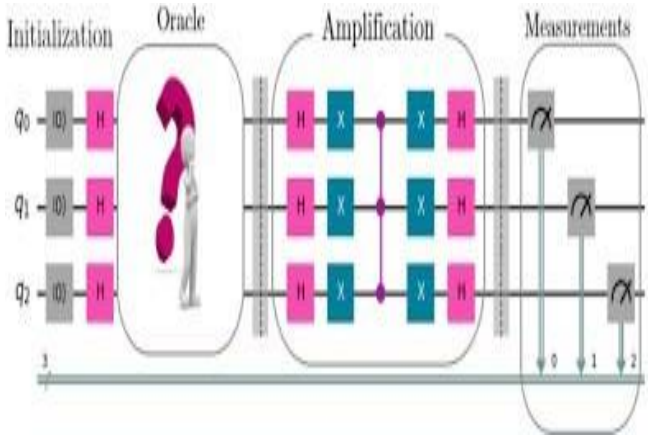

Figure 8 Quantum Annealing

7.3. Quantum Annealing for Load Optimization

Quantum annealing is a metaheuristic approach for optimization problems that use quantum fluctuations rather than classical thermal noise. The main idea is that the problem (here, load balancing) is converted into a quadratic unconstrained binary optimization (QUBO) or an Ising model, where every configuration of task-resource assignment is an energy level. The goal is to get a configuration (i.e., task distribution) that reduces the total energy of the system the most or, in other words, ensures load balance. Quantum annealers like the D-Wave systems can make use of quantum tunneling to venture through the solution space, thus, they can smoothly go beyond local minima and reach the global optimal solution faster. The approach is a good fit for:

- Highly dimensional constrains that define complex load balancing problems.
- Sharing of resources between the quantum and classical parts of a hybrid network.
- Situations, wherein it is necessary to optimize in real- time under changing workloads.

Examples of objective functions that might be employed by quantum annealing for load balancing can be:

a) Execution time variances at different nodes.
b) Power consumption.
c) Maximum permissible delays.
d) Ensuring no workload gets left behind when sharing resources.

Quantum annealing is gaining traction in the data center scheduling, IoT traffic control, and quantum-aware HPC clusters [37], [38]. Studies go on in the sense of enlarging the working space of annealers so as to be able to allocate the load in real time as the quantum hardware becomes mature. Figure 8 shows Quantum Annealing

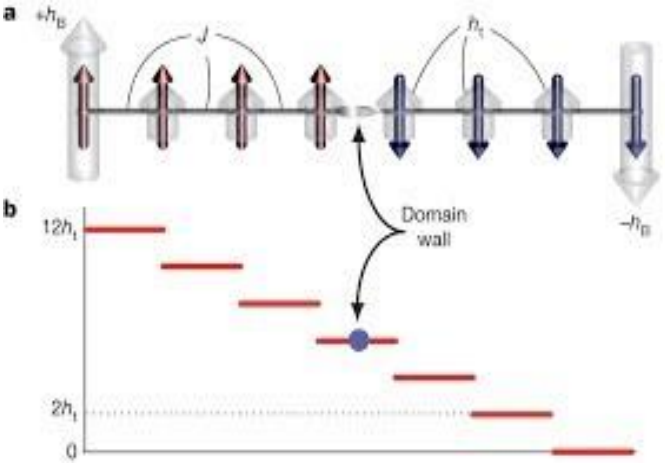

Figure 8 Quantum Annealing

Conclusion

Load balancing has been completely reinvented after several years that it has gone from simple static models to much more context-aware, intelligent and quantum-enhanced models. This survey has delved into load balancing methods in five different, though interconnected, computing paradigms - cloud, edge, fog, grid, and quantum computing. Each computing paradigm has its own specific architectonic limitations, operational targets, and performance issues that require custom-tailored balancing techniques suitable for their distinctive characters. The three Round Robin-based load balancing methods of Round Robin, Least

Connections, and Weighted Round Robin are the simplest mechanisms that form the foundation of the balancing of resources in cloud computing where the key characteristics are scalability, elasticity, and resource abstraction. However, with the increasing heterogeneity of cloud workloads and the need for low latency, the traditional approaches tend to take a back seat, and there is a requirement to integrate various monitoring and profiling techniques for real-time responsiveness and system throughput to be maintained. In the case of edge computing, the goal is to reduce the processing delay of data that is passing through the peripherals of the network, and therefore, it sets the parameters of the load balancing task as highly local and context-driven. The dynamic algorithms that are implemented in simulators like EdgeCloudSim, schedulers that are resource-aware, and methods based on proximity, such as those in MEC networks, are examples of the way that load balancing at the edge should be able to take into consideration the user that is moving around, the limited computing power, and the changing network conditions. Fog is seen as both a layer of resources from the cloud and the edge. The fog layer is the nature of the low-latency and energy- conscious load balancers. ALB technique, the scheduling method based on fuzzy logic, and energy-efficient algorithms are the greatest contributors to the application of energy-saving methods along with latency in the context of distributed and resource-limited fog nodes. These methods are especially significant for IoT to support its scalability and sustainability since it has been thought that real-time decision- making and power-consumption are the most essential factors in this ecosystem. Metaheuristic-based methods like Genetic Algorithms (GA), Ant Colony Optimization (ACO), and heuristic basics such as Min-Min scheduling have been reported in various research studies to be effective solutions to handle load balancing in the grid computing paradigm. It is essential to understand that these algorithms are capable of distributing the workload in a combinatorial manner and simultaneously taking various factors into account, including the execution time, availability of resources, and the communication overhead. Such new quantum computer architectures radically redefine the dimensions of computation and with that, the principles of work redistribution also change. Transfer of information through quantum entangled states allows for an almost instantaneous setting of task assignments, as one of the fastest quantum algorithms, Grover's algorithm, offers a rapid search of an optimal schedule. By utilizing phenomena like quantum tunneling, quantum annealing can open a fresh route to high- dimensional load optimization problems. The future may open to quantum capabilities where load balancing is no longer constrained by traditional computation boundaries, but rather given a new dimension by the quantum.

The main point of the research is that it is not possible to find a universal solution for load balancing. The working conditions of different paradigms vary significantly and all of the contextual characteristics must be taken into account if work distribution efficiency is to be achieved. The nearest roadmap for the future of work distribution leads to hybrid, context- aware architectures that include the best of various approaches (rules, machine learning, and quantum-inspired optimization) in unified and autonomous units. The systems will require making inductive decisions relying on entirely static characteristics, as well as dynamic and predictive analytics, with behavioral patterns, energy limitations, mobility, and data locality being some of the variables analyzed. In addition to that, AI and RL are set to become game changers in the creation of independently operating load balancers that can carry out perpetual self-improvement. The combining of AI and quantum computing has a potential to bring about quantum- accelerated learning models that can decide the scheduling in real-time. Collaborative load balancing across nodes could become a reality, with the help of federated and decentralized learning models, while the privacy of data is still maintained. In conclusion, the domain of load balancing is changing from a focus on systems to a multidisciplinary area dealing with distributed systems, AI, quantum physics, and energy-efficient computing. As computer paradigms keep changing, the creation of smart, reliable, and scalable load balancing systems will play a crucial role in allowing resilient, adaptable, and green future computing infrastructures.

Acknowledgements

The authors extend their heartfelt gratitude to Dr. Pradeep S for his invaluable guidance, continuous support, and mentorship throughout the course of

this research. With over 15 years of experience in academia and research, his insights and expertise significantly enriched the quality and direction of this work. The authors also appreciate the support and encouragement received from peers and collaborators during the development of this paper.

References

[1]. Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. Journal of Internet Services and Applications, 1(1), 7–18. https://doi.org/10.1007/s13174-010-0007-6

[2]. Beigi-Mohammadi, N., Ghobaei-Arani, M., & Souri, A. (2022). A comprehensive survey on fog computing: State-of-the-art and research challenges. Journal of Systems Architecture, 122, 102372. https:// doi.org/ 10.1016/j.sysarc.2021.102372

[3]. Buyya, R., Broberg, J., & Goscinski, A. (Eds.). (2011). Cloud computing: Principles and paradigms. Wiley.

[4]. Abbas, N., Zhang, Y., Taherkordi, A., & Skeie, T. (2018). Mobile edge computing: A survey. IEEE Internet of Things Journal, 5(1), 450–465. https:// doi.org/ 10.1109/ JIOT.2017.2750180

[5]. Mitzenmacher, M. (2001). The power of two choices in randomized load balancing. IEEE Transactions on Parallel and Distributed Systems, 12(10), 1094–1104. https:// doi.org/10.1109/71.963420

[6]. Zhang, Z., Chen, L., Chen, J., & Hu, X. (2013). A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. The Journal of Supercomputing, 63(2), 538–560. https://doi.org/10.1007/s11227-012-0817-2

[7]. Jennings, B., & Stadler, R. (2015). Resource management in clouds: Survey and research challenges. Journal of Network and Systems Management, 23(3), 567–619. https:// doi.org/10.1007/s10922-014-9307-7

[8]. Aazam, M., & Huh, E.-N. (2014). Fog computing and smart gateway based communication for cloud of things. In 2014 IEEE International Conference on Future Internet of Things and Cloud (pp. 464–470). IEEE. https:// doi.org/ 10.1109/ FiCloud.2014.83

[9]. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., & Zomaya, A. Y. (2020). Edge intelligence: The confluence of edge computing and artificial intelligence. IEEE Internet of Things Journal, 7(8), 7457–7469. https://doi.org/10.1109/JIOT.2020.2984887

[10]. Ghosh, S., Banerjee, S., & Mandal, S. (2017). An energy-efficient data locality-aware load balancing strategy for Hadoop. Concurrency and Computation: Practice and Experience, 29(17), e4144. https:// doi.org/ 10.1002/cpe.4144

[11]. Das, P., Chakrabarti, S., Debnath, B., Mandal, S., & Ghosh, S. (2022). A survey on classical and quantum resource management in hybrid quantum-classical computing systems. ACM Computing Surveys, 55(1), 1–36. https:// doi.org/ 10.1145/3490236

[12]. Hwang, J., Choi, S., & Kim, S. (2016). Load balancing in cloud computing: A state of the art survey. Journal of Network and Computer Applications, 73, 48–65. https://doi.org/10.1016/j.jnca.2016.08.007

[13]. Khiyaita, S. A., Zbakh, M., Moussaoui, O., & El Omri, A. (2012). Load balancing cloud computing: State of art. In National Days of Network Security and Systems (JNS2) (pp. 106–109). IEEE.

[14]. Kokilavani, T., & Amalarethinam, D. I. G. (2011). Load balanced min-min algorithm for static meta-task scheduling in grid computing. International Journal of Computer Applications, 20(2), 43–49. https://doi.org/10.5120/2432-3256

[15]. Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems, 28(5), 755–768. https:/ /doi.org/ 10.1016/ j. future. 2011.04.017

[16]. Wu, L., Li, X., Zhang, L., & Liang, J. (2015). A hierarchical weighted round robin algorithm for load balancing in cloud computing. In 2015 International Conference on Smart City and Systems Engineering (pp. 159–162). IEEE. https://doi.org/10.1109/ICSCSE.2015.52

[17]. Mishra, M., & Sahoo, A. (2018). On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. IEEE

Transactions on Cloud Computing, 6(1), 41–54. https:// doi.org/ 10.1109/ TCC. 2015.2400462

[18]. Verma, A., Ahuja, P., & Neogi, A. (2008). pMapper: Power and migration cost aware application placement in virtualized systems. In Middleware 2008 (pp. 243–264). Springer. https://doi.org/10.1007/978-3-540-89856-6_13

[19]. Sonmez, A., Ozgovde, A., & Ersoy, C. (2018). EdgeCloudSim: An environment for performance evaluation of edge computing systems. Transactions on Emerging Telecommunications Technologies, 29(11), e3493. https://doi.org/10.1002/ett.3493

[20]. Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials, 19(4), 2322–2358. https:// doi.org/10.1109/COMST.2017.2745201

[21]. Mach, P., & Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys & Tutorials, 19(3), 1628–1656. https:// doi.org/ 10.1109/ COMST.2017.2682318

[22]. Baccarelli, E., Naranjo, P. G. V., Scarpiniti, M., Shojafar, M., & Abawajy, J. H. (2017). Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study. IEEE Access, 5, 9882–9910. https:// doi.org/ 10.1109/ACCESS.2017.2705630

[23]. Varghese, B., & Buyya, R. (2018). Next generation cloud computing: New trends and research directions. Future Generation Computer Systems, 79, 849–861. https://doi.org/10.1016/j.future.2017.09.020

[24]. Chiang, M., & Zhang, T. (2016). Fog and IoT: An overview of research opportunities. IEEE Internet of Things Journal, 3(6), 854–864. https:// doi.org/ 10.1109/ JIOT. 2016.2584538

[25]. Deng, R., Lu, R., Lai, C., Luan, T. H., & Liang, H. (2016). Optimal workload allocation in fog–cloud computing toward balanced delay and power consumption. IEEE Internet of Things Journal, 3(6), 1171–1181. https:// doi.org/ 10.1109/ JIOT. 2016.2565516

[26]. Sarkar, S., & Misra, S. (2016). Theoretical modelling of fog computing: A green computing paradigm to support IoT applications. IET Networks, 5(2), 23–29. https://doi.org/10.1049/iet-net.2015.0034

[27]. Abraham, A., Buyya, R., & Nath, B. (2000). Nature's heuristics for scheduling jobs on computational grids. In Proceedings of the 8th IEEE International Conference on Advanced Computing and Communications (ADCOM) (pp. 45–52). IEEE. https://doi.org/10.1109/ADCOM.2000.917757

[28]. Tiwari, A., & Singh, R. K. (2015). A genetic algorithm based efficient load balancing technique for grid computing. International Journal of Computer Applications, 122(3), 1–5. https://doi.org/10.5120/21758-5006

[29]. Dorigo, M., & Stützle, T. (2004). Ant colony optimization. MIT Press.

[30]. Dutta, A., & Sharma, A. K. (2020). Ant colony optimization based task scheduling and load balancing in cloud computing. Procedia Computer Science, 167, 2360–2369. https:// doi.org/ 10.1016/ j.procs. 2020.03.288

[31]. Casanova, H., Legrand, A., & Quinson, M. (2008). SimGrid: A generic framework for large-scale distributed experiments. In 2008 10th IEEE International Conference on Computer Modeling and Simulation (pp. 126–131). IEEE. https:// doi.org/ 10.1109/ UKSIM.2008.27

[32]. Krishna, D. G. A., & Rao, A. S. (2009). Comparative study of min-min, max-min and sufferage scheduling algorithms in grid computing. International Journal of Computer Science and Network Security, 9(5), 106–112.

[33]. Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press.

[34]. Pirandola, S., Eisert, J., Weedbrook, C., Furusawa, A., & Braunstein, S. L. (2015). Advances in quantum teleportation. Nature Photonics, 9(10), 641–652. https:// doi.org/ 10.1038/nphoton.2015.154

[35]. Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In Proceedings of the 28th Annual ACM Symposium on Theory of Computing (pp. 212–219). ACM. https:// doi.org/ 10.1145/ 237814.237866

[36]. Tullsen, D. M., & Frank, M. P. (2022). Quantum-assisted load balancing for high-performance computing systems. IEEE Transactions on Quantum Engineering, 3, 1–9. https:// doi.org/ 10.1109/ TQE. 2022. 3152470

[37]. Benedetti, M., Realpe-Gomez, J., Perdomo-Ortiz, A., & Perdomo, O. (2019). A generative modeling approach for benchmarking and training shallow quantum circuits. npj Quantum Information, 5(1), 45. https://doi.org/10.1038/s41534-019-0157-8

[38]. Lucas, A. (2014). Ising formulations of many NP problems. Frontiers in Physics, 2, 5. https://doi.org/10.3389/fphy.2014.00005