Smart Internet of Things

www.siot.reapress.com

Smart. Internet. Things. Vol. 1, No. 1 (2024) 83–92.

Paper Type: Original Article

# Load Balancing Strategies for Scalable and Resilient Cloud Systems Aper Title

**Akshyayanand Pani[1,*], Aman Kumar[1], Sourav Nayak[1]**

[1] School of Computer Engineering, KIIT (Deemed to Be) University, Bhubaneswar-751024, Odisha, India; ak.pani2003@gmail.com, aman.k2003@gmail.com, souravn2003@gmail.com.

**Citation:**

| | |
|---|---|
| *Received: 28 February 2024*<br>*Revised: 11 April 2024*<br>*Accepted: 03 June 2024* | Pani, Akshyayanand., Kumar, A., & Nayak, S. (2024). Load balancing strategies for scalable and resilient cloud systems aper title. *Smart Internet of Things, 1* (1), 83-92. |

## Abstract

Task scheduling in cloud environments poses NP-hard optimization challenges due to diverse user requests and infrastructure configurations. Load imbalance, whether underloaded or overloaded, leads to system failures impacting electricity consumption, execution time, and machine reliability. Effective load balancing is crucial to mitigate these issues. This involves distributing tasks, dependent or independent, across Virtual Machines (VMs) to achieve load equilibrium. Various types of loads, such as memory, CPU, and network, contribute to the complexity. Researchers have proposed diverse load-balancing approaches aimed at optimizing different performance metrics. This paper presents a taxonomy of load-balancing algorithms in cloud computing, along with a discussion on thirteen performance parameters of 10 scheduling algorithms.

**Keywords:** Cloud computing, Scalability, Virtualization, Energy consumption, Load balancing, Elasticity, Taxonomy.

## 1 | Introduction

Cloud computing is an internet-based network technology that is rapidly revolutionizing communication technology through the provision of online computing resources on demand that is billed on a Pay-As-You-Go (PAYG) framework to customers with a broad spectrum of requirements and backgrounds. It includes both hardware and software applications, as well as software development platforms and testing tools. It has three service models while the former comes under Infrastructure as a Service (IaaS) cloud with four deployment options, namely, private, public, hybrid and community; the latter two come under Software as a Service (SaaS) cloud and Platform as a Service (PaaS) cloud respectively. These deployment models offer different levels of control, flexibility, and cost-effectiveness depending on the organization's needs and resources. Private clouds provide the highest level of control and security but may require more investment,

Corresponding Author: ak.pani2003@gmail.com

while public clouds offer greater scalability and cost savings. Hybrid and community clouds strike a balance between these extremes, allowing organizations to leverage the benefits of both private and public clouds. Cloud computing, adopted by numerous leading IT firms like Amazon, Apple, Google, HP, IBM, Microsoft, Oracle, among others, represents a transition towards utility computing.

The Cloud computing model is efficient if its resources are utilized with minimal idle time; and such an efficient utilization can be achieved only by employing and maintaining proper management of cloud resources. This is one of the most important characteristics of the cloud system, which is in the virtual form. It is a crucial role for the Cloud Service Provider (CSP) to provide services to users on a rented basis, which is a complex task given the available virtual resources. Therefore, among researchers this topic has gained a peculiar interest and attention to research. This load balancing has a positive impact on the system performance. The CSP strikes off a unique and delicate balance between financial rewards and user satisfaction through load balancing in a scalable manner. The load balancing procedure also addresses the Service Level Agreements (SLAs), which are the contractual agreements between the CSP and the cloud users.

The load balancing in clouds may be among physical hosts or VMs. In the context of cloud computing, the allocation of different tasks to VMs is called the load and computing of this load is task scheduling which is an NP-hard optimization problem. This balancing mechanism allocates the dynamic workload evenly among all the nodes (hosts or VMs) (*Fig. 1*). There are two types of load-balancing algorithms: static and dynamic. The static-based balancing algorithms are appropriate for stable environments with homogeneous systems. Dynamic-based balancing algorithms are more adaptable and effective in both homogeneous and heterogeneous environments. However, the application of static load balancing procedures has less system overhead as compared to the dynamic load balancing procedures. If a Virtual Machine (VM), also known as a VM, encounters exhaustion of resources as a result of an elevated amount of incoming task requests, then no resources remain available to handle the new demands.

The VM is considered to have entered an overloaded condition under such circumstances. At the moment, tasks will either starve to death or come to a halt with little prospect of completion. As such, it becomes necessary to relocate the jobs to a separate resource on a different VM. Three fundamental processes form the basis of the workload migration process: workload migration, which transfers additional jobs to available resources; resource discovery, which locates another appropriate resource; and load balancing, which assesses the existing strain on machine resources. Three distinct units, referred to as task migration units, resource discovery units, and load balancer units, respectively, carry out these tasks.

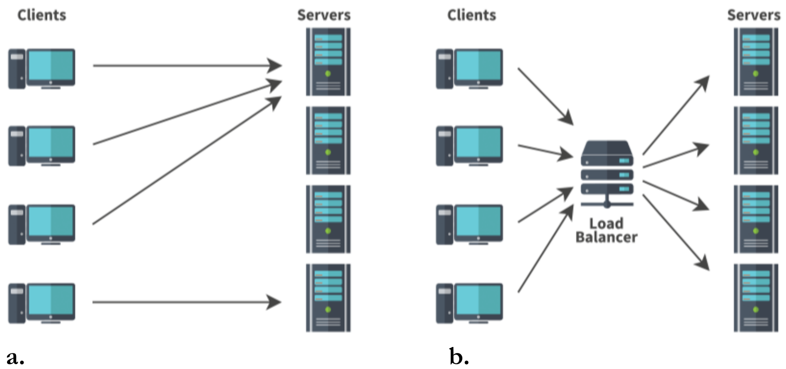


**Fig. 1. With load balancing, servers will not be overloaded or sit idle; a. Whitout load balancing, b. With load balancing.**

In a cloud computing environment, load balancing is an intricate process of dynamically redistributing the workload in real time. This ensures that no computer is idle, overloaded, or underutilized, thereby maximizing Throughput (TP). The primary goal of this mechanism is to optimize various parameters like Response Time (RT), execution time, and system stability, improving cloud performance and providing flexibility a key feature that sets cloud environments apart from traditional systems. *Fig. 2* presents the taxonomy of load-balancing algorithms.

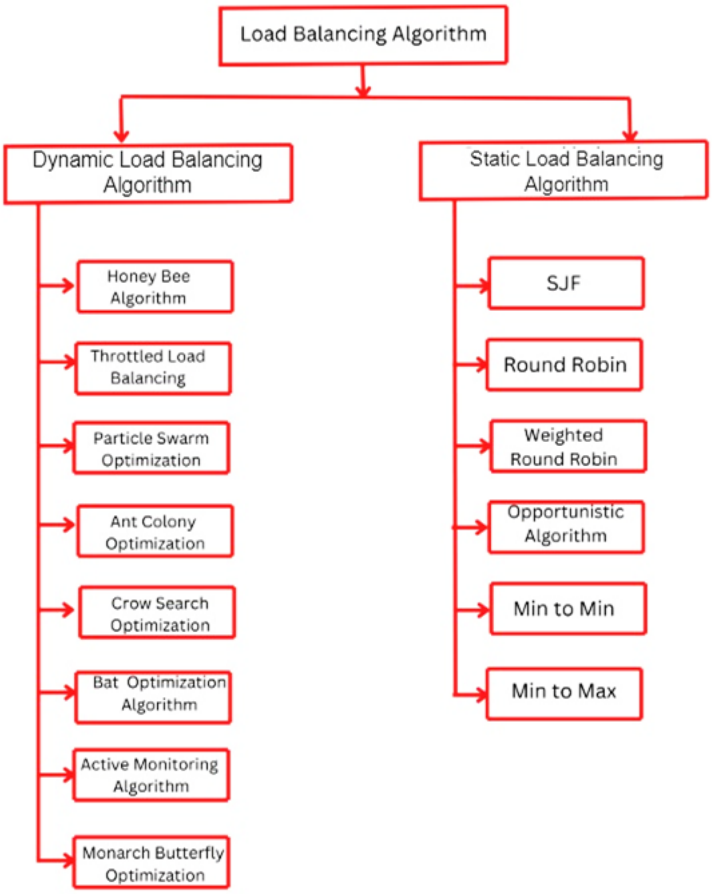


**Fig. 2. Taxonomy of load balancing algorithms.**

Scalability is proving to be the cornerstone of this flexibility. It enables cloud environments to adjust resource allocation based on fluctuating demands gracefully. Imagine an application that experiences a sudden spike in traffic; in the traditional world, such spikes could cripple a single server, leading to slow RTs or even outages, but cloud computing systems have the capacity to dynamically and in real-time scale resources both horizontally (by adding more resources) and vertically (by adding more power to existing resources) in response to shifting workload needs. Additional resources, if any, are seamlessly provided to distribute the workload and ensure a smooth user experience effectively. Conversely, during periods of low activity, resources can be reduced, resulting in cost optimization and avoidance of unnecessary Energy Consumption (EC). Scalability guarantees that services and applications can efficiently handle varying demands. The demand for computing resources can change greatly in the dynamic corporate environment of today depending on many factors such as user activity, seasonal trends, and market circumstances.

Organizations can react quickly to uncertainties by assigning more resources or reducing them as per the demand in real-time, enabling them to capture the pulse. With this degree of elasticity, companies can continue to provide services without lacking in any customer experience and performance without losing profits. In turn, load balancing, a component discussed above, ensures that this adaptation happens efficiently and smoothly, preventing any performance bottlenecks and maintaining a consistent user experience. Together, they create a dynamic and resilient cloud environment capable of effortlessly handling unexpected surges, allowing CSPs to maintain competitive price innovation in an agile fashion.

We provide a review of current load balancing methods in this study, which have been specifically developed to work in cloud systems while focusing on scalability. We have presented and described a cloud system architecture to provide context in the cloud system. A taxonomy is presented and elaborated for the classification of load-balancing algorithms in the cloud. Various performance parameters are explained as well as compared among different research on load balancing in a cloud. The main aim of the paper is to examine the existing research work and new developments along with the advantages and shortcomings and analyze the role of load balancing in achieving scalability in cloud applications. A comparison is also made among different existing load-balancing techniques and the challenges associated with cloud load-balancing. The survey also outlines factors responsible for load unbalancing problems and suggests methods that can be used in future work. The contributions of this paper are summarized as follows:

I. Explore the factors that cause load unbalancing problems in cloud computing.

II. Provide a systematic overview of the existing approaches in the load balancing process and the way in which these approaches have been used in cloud technology.

III. Provide a detailed classification of different load-balancing methods and their characteristics.

IV. Analyze how scalability can be achieved by load-balancing algorithms in a cloud environment.

The remaining paper is structured as follows. Section 1 features a brief description about the load balancing model and system model for cloud computing. Section 2 proposes taxonomy-based classification. The results are evaluated in Section 3, while Section 4 discusses the open issues in cloud load balancing. And finally, Section 5 concludes our work and points out some future directions.

# 2 | Literature Review

When designing a system model, various factors such as cost, complexity, speed, system portability, and security necessitate careful consideration. Cloud computing architecture distinguishes itself from traditional distributed system architecture through its highly scalable nature, abstract entity status, provision of distinct service levels to consumers, and governance by economies of scale, all facilitated by dynamic demand services via virtualization.

A cloud hosting facility consists of a limited number of diverse physical hosts, each differentiated by characteristics like the amount of memory, bandwidth, processing element lists, host identification numbers, and processing throughput represented by Millions of Instructions Per Second (MIPS). Every host supports numerous VMs with comparable characteristics. For resource mapping, tasks from different users are transmitted to the serial scheduler or central load balancer. Every VM performs one particular task at a time, with the load balancer allocating tasks based on constraints such as resource accessibility and time limits. Once a task is finished, the resources that have been employed are released, allowing the creation of additional VMs to process further requests.

The Expected Time to Compute (ETC) matrix is used to model tasks in a diverse computing environment. The formula ($L_n = P_m$) gives the value of ($ETC_{nm}$), where ($L_n$) represents the length of the ($n^{th}$) task in terms of Million Instructions (MIs), and ($P_m$) denotes the processing speed of the ($m_{th}$) VM in terms of MIPS. Each VM operates in two states: active and idle. When a VM is idle, its EC is 60% of its EC when it is active. In cloud systems, two crucial performance metrics are identified: Makespan (MS) and EC. Due to varying execution times across different VMs in the cloud system, Makespan refers to the maximum time taken by any VM to execute all input tasks within the system. Minimizing Makespan is essential for achieving optimal load balancing. The execution time of $m^{th}$ VM ( $ET_m$) is based on the decision variable $X_{mn}$, where,

$$X_{nm} = \begin{Bmatrix} 1 \text{ if } T_n \text{ is assigned to } VM_m \\ 0 \text{ if } T_n \text{ is not assigned to } VM_m \end{Bmatrix}. \quad (1)$$

And the $ET_m, 1 \leq m \leq N$ is calculated as

$$ET_m = \sum_{n=1}^{k} x_{nm} \times ETC_{nm}. \quad (2)$$

The Makespan is the maximum of $ET_{m \text{ i.e}}$.

$$MS = Max. [ET_m]_{m=1}^{k}. \quad (3)$$

Suppose the VM $\mathrm{VM_m}$ consumes $\gamma_m$ Joules/MI in operative state and $\delta_m$ Joules/MI in a non-operative state. The $\mathrm{VM_m}$ remains in a non-operative state for $\mathrm{MS} - \mathrm{ET_m}$ seconds. And the total energy consumed in the operative and non-operative state is in *Eq. (4)* as follows:

$$\mathrm{EC} = \sum_{m=1}^{k}[[\mathrm{ET_m} \times \gamma_m + (\mathrm{MS} - \mathrm{ET_m}) \times \delta_m] \times \mathrm{P_m}]. \quad (4)$$

I. Throughput (TP): the amount of user requests (tasks) that a VM can process in a given amount of time is called throughput. The value of throughput establishes the functionality of the system. A high throughput suggests a well-performing system. Throughput and Makespan are inversely related to each other.

II. Reliability (R): reliability is the ability to operate in accordance with system specifications consistently. To increase the reliability of the task, it is moved to any other resources (VMs) in the event of a failure during execution. The reliable system increases the system's stability.

III. Accuracy (A): accuracy represents the degree of precision in task execution outcomes, indicating how closely a measurement aligns with the actual value of the task being assessed. The IT industries now place a higher priority on system accuracy in response to client demand. A decrease in accuracy impacts the system's completion time to some extent.

IV. Makespan (MS): this is the entire amount of time needed to finish every task that has been sent into the system. The system's Makespan is the longest period of time that the operating over the data centre as a host. When there are tasks that must be completed in a certain order of priority, the CSP must make concessions in order to maintain system stability evaluation. Proper load balancing of the system is achieved by the ideal MS.

V. Scalability (S): it is a property of a model or system that indicates how well the system can function in unforeseen situations. It denotes the level at which a balanced system can continue to function whether the quantity, size, or effort is increased. Resources in a scalable cloud system will be scaled back and forth on a regular basis.

VI. Fault Tolerance (FT): the fault-tolerant method allows the system to continue operating even in the event that one or more system components malfunction. It also removes barriers caused by logical mistakes. The fault-tolerance level can be computed using the number of failure points, either one or multiple points of failure. To solve some issues with the cloud system, the service providers require additional resources or VMs. Although there are some extra expenses involved in this process, the consumer can receive a system that is error-free.

VII. Associated Overhead (AO): this is the overhead resulting from the algorithms' operation. The method of balancing the system's load leads to certain overhead expenditures. Minimum overhead happens when the system's load is appropriately balanced.

VIII. Migration Time (MT): the real amount of time needed to move a VM or job from one resource to another. The job transfer could happen between VMs. VM on one host or several hosts. Similarly, VM migration will occur between hosts inside a data centre or between data centres. A task is moved when it takes resources from numerous VMs or when there is an interruption in the task's execution. Likewise, if a VM crashes while running, it is moved to a different host. A higher number of VM migrations causes longer MTs, which impairs the system's Makespan and load balancing.

IX. Response Time (RT): this is the amount of time the system needs to respond to a request. Stated differently, it is the total of the waiting, gearbox and service times. Consequently, the relationship between RT and system performance is inverse. A superior Makespan value is obtained with the ideal RT.

X. Associated Cost (AC): distributing the workload across a system comes with some overhead costs. For example, the services offered by EC2 can reduce the entire cost by up to 72% while the resource is fully utilized [1]. When the system's load is properly balanced, the overhead is minimized. The process of distributing the load results in overhead expenses. However, when the system's load is well-balanced, the overhead costs are kept to a minimum.

XI. Energy Consumption (EC): the total energy used by all the ICT devices connected to a cloud system is known as the EC of the system. To calculate this, three types of devices are considered: personal devices (desktops, laptops, smartphones, etc.), networking equipment (routers, switches, hubs, etc.), and local servers (application servers). There are four main strategies to save energy: using energy-efficient hardware, employing energy-aware scheduling methods, minimizing power in server clusters, and reducing power in wired and wireless networks. By implementing these strategies, the overall EC of the cloud system can be optimized.

XII. Resource utilization (RU): it calculates the proportion of unused or idle resource fragments. It is defined as the ratio of efficient utilization of cloud infrastructure (i.e. VMs). The higher this value, the better the use of cloud resources.

XIII. Service Level Agreement (SLA): a service provider and customer enter into a contract known as an SLA, which specifies the level of service that will be provided, along with performance indicators and consequences for not fulfilling them.

*Table 1* outlines the specific approaches used in different simulation environments for various load-balancing algorithms. These algorithms are designed to distribute workloads across multiple resources, ensuring efficient utilization and optimized performance. The methodology employed in each simulation environment can vary, reflecting the unique requirements.

**Table 1. Various load balancing algorithms in different environments.**

| SL No. | Work | Approaches | Simulator | Environment |
|---|---|---|---|---|
| 1 | [2] | Proposed a Load-balancing algorithm with a hybrid combination of GA and PSO algorithms. | CloudSim | Homogeneous |
| 2 | [3] | Proposed a Metaheuristic Hybrid HEFT-PSO-GA (HEPGA) algorithm which enhances Resource Utilisation when specific optimization goals need to be addressed. | CloudSim | Heterogenous |
| 3 | [4] | Proposed a hybrid metaheuristic manta ray Modified Multi-objective Harris Hawk Optimization Algorithm (MMHHO) which solves load balancing problem in cloud by shortening the waiting time of a task. | CloudSim | Homogeneous |
| 4 | [5] | Proposed a Metaheuristic Bio inspired Lion Load Balancing Algorithm for cloud environment which improved identification of fitness of nodes. | CloudSim | Heterogenous |
| 5 | [6] | Proposed a dynamic Load Balancing algorithm Multi-Criteria Decision-Making (MCDM) for IIOT systems. | CloudSim | Homogeneous |
| 6 | [7] | Proposed a scheduling mechanism which automatically takes the decision based on the upcoming tasks onto the cloud console to handle the dynamic nature of tasks in cloud environments. | CloudSim | Homogenous |
| 7 | [8] | Proposed a search optimization algorithm which can be used for handling task scheduling uncertainty problems. | CloudSim | Homogeneous |
| 8 | [9] | Proposed a multi-objective PSO algorithm with Pareto dominance to achieve high quality of service, throughput, scalability, low RT, and optimal bilateral transposed conv filtering. | Python, Spyder IDE | Homogenous |
| 9 | [10] | Proposed an adaptive weight based multi-dimensional learning strategy for load balancing in cloud environments. | Cloudsim | Homogenous |
| 10 | [11] | Proposed a dual-phase Metaheuristic Clustering Sparrow Search Algorithm-Differential Evolution (CSSA-DE) which decreases the RT and enhances RU by balancing workloads and migrating tasks. | Cloudsim | Heterogeneous |

# 3|Proposed Study

Load balancing within cloud computing, especially in systems with multiple objectives, presents a recognized challenge known to be NP-complete. These objectives often include energy conservation, minimizing MS, and maximizing throughput, among others. Various heuristic techniques, or sub-optimal algorithms, have been proposed by researchers to address this complex problem and achieve a sub-optimal solution for load balancing in cloud environments. Load balancers play a crucial role in achieving system load equilibrium, and a comprehensive overview of different load balancers is provided in *Table 2* below.

**Table 2. Overview of different load balancers.**

| Load Balancers | Descriptions |
|---|---|
| Hardware load Balancer (HLD) | Web traffic is distributed among several network servers by the hardware device known as a Host Load Distributor (HLD), which controls each server in a network. An alternative to HLD is LBaaS. It is suitable for heterogeneous environments and provides worldwide server load balancing. |
| Network Load Balancer (NLB) | In the OSI model, NLB is located at Layer 4, or the network layer. It is ideal for TCP traffic load balancing. For each subnet, it offers a static IP address that applications can use as the balancer's front-end IP. To prevent overloading, it is used to divide network traffic among several VMs in a cluster. |
| Application Load Balancer (ALB) | ALB operates at OSI model layer 7. For high-level HTTP and HTTPS traffic load balancing, it is required. By guaranteeing that the most recent SSL/TLS cyphers and protocols are always used, ALB interprets and enhances the application's security. |
| Classic Load Balancer (CLB) | Amazon Elastic Load Balancing, often known as CLB, provides basic load balancing across several Elastic Cloud Compute (EC2) instances. It functions at both the connection and request levels. CLB is designed for the EC2-Classic network applications. |
| Elastic Load Balancer (ELB) | Another name for it is an AWS load balancer. It splits up incoming work among several Amazon EC2 instances. ALBs, NLBs, and CLBs are the three types of load balancers that are available [12]. |
| HAProxy Load Balancer (HAPLB) | Two interfaces are included in its configuration: one for users and one for the server LAN. Additionally, the HAPLB functions in OSI model Layer 4 and Layer 7. It is typically utilized in load balancers, such as ALOHA or reverse proxy. Infrastructures that are dependable and scalable are offered by the ALOHA Load-Balancer. Using HAProxy, the ALOHA Load-Balancer created a number of open-source load-balancing programs. |

In this research paper, we've extensively covered 10 recent cloud-based Load Balancing algorithms from different perspectives, evaluated them on the basis of 13 different bases, and arranged the result in a simple YES/NO format, which reflects does the algorithm show that particular characteristic or not as mentioned by the authors of that particular paper. This survey allowed us to form an opinion that there might be a large supply of algorithms in the market available. Still, most of them are a derivative of one or the other with innovative tweaks from each other. The load-balancing algorithms and their corresponding consideration of different performance metrics are presented in *Table 3*. These metrics indicate the performance of different load-balancing algorithms.

**Table 3. Load balancing with their corresponding performance metrics.**

| Work | Environment | Simulator | TP | R | A | MS | S | FT | AO | MT | RT | AC | EC | RU | SLA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [2] | Homogenous | Cloudsim | YES | | | YES | | | | | | YES | | | |
| [3] | Heterogenous | Cloudsim | | | | YES | | | | | | | | YES | |
| [4] | Homogenous | Cloudsim | YES | YES | YES | YES | | | | YES | YES | | YES | | |
| [5] | Homogenous | Cloudsim | YES | | | | | YES | | YES | YES | | | | |
| [6] | Homogenous | Cloudsim | YES | | YES | YES | YES | YES | | | | | | | |
| [7] | Homogenous | Cloudsim | | | | YES | | | | | YES | | YES | | YES |
| [8] | Homogenous | Cloudsim | YES | | YES | YES | YES | | | | | YES | YES | YES | |
| [9] | Homogenous | Python, Spyder IDE | YES | YES | YES | YES | YES | YES | | YES | YES | | | YES | |
| [10] | Homogenous | Cloudsim | YES | YES | YES | YES | YES | YES | YES | YES | YES | | YES | YES | YES |
| [11] | Heterogenous | Cloudsim | YES | YES | YES | YES | | | | | | | YES | YES | YES |

## 4 | Conclusion

In this study, we have described a range of load-balancing methodologies applicable to diverse cloud computing environments, including homogeneous and heterogeneous setups. Our study provides a comprehensive system architecture featuring distinct models for hosts and VMs. We have elaborated on various performance metrics outlined in the preceding tables, which serve to assess system efficacy. Detailed insights into the calculation of EC within the system are provided. Furthermore, we have introduced a taxonomy for categorizing load-balancing algorithms in cloud environments. Future scope of work will involve assessing the efficacy of the proposed algorithms within real-world cloud deployments. Additionally, we aim to implement and compare all discussed techniques, thereby advancing our understanding of their relative strengths and weaknesses in real-world scenarios.

## Acknowledgements

We would like to extend our heartfelt gratitude to all those who have contributed to the successful completion of this research paper. We are profoundly grateful to our professor, Dr. Hitesh Mahapatra, whose guidance and support have been invaluable throughout this process. His dedication and commitment to helping his students have been crucial in ensuring the quality and clarity of our work. The encouragement and insightful feedback provided by Dr. Mahapatra have been instrumental in helping us accomplish this task. His contributions were essential in facilitating our research and ensuring its integrity. Lastly, we thank our families and friends for their unwavering support, giving us the strength to undertake this journey. Thank you all for your invaluable assistance and support.

## Author Contribution

Conceptualization: Akshyayanand Pani; data curation: Akshyayanand Pani, Aman Kumar; formal analysis: Akshyayanand Pani, Aman Kumar; funding acquisition: no funding; investigation: Akshyayanand Pani, Aman Kumar; methodology: Akshyayanand Pani; project administration: Akshyayanand Pani; resources: Akshyayanand Pani, Aman Kumar, sourav nayak; software: Akshyayanand Pani, Aman Kumar, Sourav Nayak; supervision: Dr. Hitesh Mohapatra; validation: Akshyayanand Pani; visualization: Akshyayanand Pani; writing - original draft: Akshyayanand Pani; writing - review & editing: Akshyayanand Pani, Aman Kumar, Sourav Nayak. All authors have read and agreed to the published version of the manuscript.

## Funding

This research received no financial support.

## Data Availability

Data presented in this study were procured from publicly available research papers listed in the reference section of this article. No new data were created or analyzed.

## Conflicts of Interest

The authors declare that they have no financial or non-financial interests related to this research. Specifically, no author has received research funding from funding agencies, support from commercial sources, honoraria for presentations, holds a position on an advisory board, is an inventor on any patents related to this work, or is employed by an organization or company that could pose a conflict of interest.

## References

[1] AWS. (2024). *Amazon EC2 reserved instances*. https://aws.amazon.com/ec2/pricing/reserved-instances/

[2] Vijay, R., & Sree, T. R. (2023). Resource scheduling and load balancing algorithms in cloud computing. *Procedia computer science*, *230*, 326–336. DOI: 10.1016/j.procs.2023.12.088

[3] Mikram, H., El Kafhali, S., & Saadi, Y. (2024). HEPGA: a new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment. *Simulation modelling practice and theory*, *130*, 102864. DOI: 10.1016/j.simpat.2023.102864

[4] Haris, M., & Zubair, S. (2022). Mantaray modified multi-objective harris hawk optimization algorithm expedites optimal load balancing in cloud computing. *Journal of king saud university - computer and information sciences*, *34*(10, Part B), 9696–9709. DOI: 10.1016/j.jksuci.2021.12.003

[5] Kaviarasan, R., Balamurugan, G., & Kalaiyarasan, R. (2023). Effective load balancing approach in cloud computing using Inspired Lion Optimization Algorithm. *E-prime - advances in electrical engineering, electronics and energy*, *6*, 100326. DOI: 10.1016/j.prime.2023.100326

[6] Jamal, M. K., & Muqeem, M. (2023). An MCDM optimization based dynamic workflow scheduling used to handle priority tasks for fault tolerance in IIOT. *Measurement: sensors*, *27*, 100742. DOI: 10.1016/j.measen.2023.100742

[7] Mangalampalli, S., Karri, G. R., Kumar, M., Khalaf, O. I., Romero, C. A. T., & Sahib, G. A. (2024). DRLBTSA: deep reinforcement learning based task-scheduling algorithm in cloud computing. *Multimedia tools and applications*, *83*(3), 8359–8387. DOI: 10.1007/s11042-023-16008-2

[8] Pabitha, P., Nivitha, K., Gunavathi, C., & Panjavarnam, B. (2024). A chameleon and remora search optimization algorithm for handling task scheduling uncertainty problem in cloud computing. *Sustainable computing: informatics and systems*, *41*, 100944. DOI: 10.1016/j.suscom.2023.100944

[9] Ghafir, S., Alam, M. A., Siddiqui, F., & Naaz, S. (2024). Load balancing in cloud computing via intelligent PSO-based feedback controller. *Sustainable computing: informatics and systems*, *41*, 100948. DOI: 10.1016/j.suscom.2023.100948

[10] Janakiraman, S., & Priya, M. D. (2023). Hybrid grey wolf and improved particle swarm optimization with adaptive intertial weight-based multi-dimensional learning strategy for load balancing in cloud environments. *Sustainable computing: informatics and systems*, *38*, 100875. DOI: 10.1016/j.suscom.2023.100875

[11] Khaleel, M. I. (2023). Efficient job scheduling paradigm based on hybrid sparrow search algorithm and differential evolution optimization for heterogeneous cloud computing platforms. *Internet of things*, *22*, 100697. DOI: 10.1016/j.iot.2023.100697

[12] AWS. (2024). *Elastic load balancing (ELB)*. https://aws.amazon.com/elasticloadbalancing/