DESIGN OF HYBRID SCALABLE FAULT TOLERANT SCHEDULING ALGORITHM IN CLOUD COMPUTING EVENTS

Osuolale, A. F., and Ibrahim, R. S.

DOI10.51459/jostir.2025.2.1.0305

Department of Computer Science, School of Computing, The Federal University of Technology, Akure, Nigeria

Correspondence
aofestus@futa.edu.ng

History
Received: 13-01-2025
Accepted: 05-02-2026
Published: April, 2026

https://www.futa.edu.ng

JOSTIR
JOURNAL OF SCIENCE, TECHNOLOGY AND INNOVATION RESEARCH
https://jostir.futa.edu.ng

ABSTRACT

As cloud computing becomes increasingly integral to modern IT infrastructures, the need for efficient and reliable scheduling algorithms is paramount. This paper presents a scalable fault-tolerant scheduling algorithm designed specifically for cloud computing environments. Our a:roach addresses the inherent challenges of dynamic resource allocation and the unpredictability of workload events. By leveraging a combination of distributed system principles and advanced error detection mechanisms, the algorithm ensures optimal task distribution while maintaining service continuity in the event of node failures. We implement a multi-layered architecture that incorporates real-time monitoring and adaptive resource management to dynamically adjust to varying workloads. The algorithm employs a priority-based scheduling model that optimizes resource utilization and minimizes latency, ensuring efficient execution of tasks across heterogeneous cloud resources. Experimental results demonstrate significant improvements in both performance metrics and fault recovery times compared to existing scheduling algorithms. This research contributes to the field by providing a robust framework that not only enhances scheduling efficiency but also improves system resilience, thereby supporting the growing demands of cloud-based applications and services.

Keywords: Scheduling Algorithms, Cloud Computing, Operating System (OS), Fault Tolerant Mechanism, node, latency

1. | Introduction

The idea of cloud computing (CC) has grown into a powerful paradigm for providing users with on demand access to computing resources anywhere in the world. The optimum utilization of cloud resources and continuous service delivery require effective scheduling algorithms. In this research, a brand-new fault tolerant, scalable scheduling algorithm technique is proposed for cloud computing events. The strategy aims to increase resource utilization, decrease reaction times, and lessen the impact of errors on cloud operations. The suggested method makes use of distributed techniques to handle scalability and fault tolerance challenges, which ultimately improve the reliability and performance of cloud services at peak times. As cloud computing (CC) becomes more and more widespread, the landscape of computing and data processing has altered. Cloud service providers must properly manage their resources to meet the diverse needs of their customers, especially when workloads suddenly increase. In contrast to the extensive research on cloud computing scheduling methods that has been carried out in the research community, (Joundy, *et al.,* 2015), This research

provides a novel scheduling strategy for cloud computing events and a survey of fault tolerant in cloud computing (Kumari and Kaur, (2021) that addresses scalability and fault tolerance issues.

The main issues with current techniques are fault tolerance,

1.1 | Resource allocation

Allocate resources over a shared cloud infrastructure, scheduling algorithms are utilized in cloud computing. In a cloud setting, events or activities can be planned, according to (Puri and Venkatesh, 2023).

i. Load balancing.

ii. Adaptive Scheduling: Dynamically modify workloads during cloud events.

These solutions typically don't have the scalability or agility needed to handle unforeseen event-driven increases in demand, though. The methodology used in this study uses a distributed design to offer fault tolerance and seamless scalability i.e. Handling Failures Computing nodes regularly exchange information about their state with the master node. The master distributes work to other nodes in order to maintain fault tolerance and uninterrupted service delivery in the case of a node failure. The program continuously assesses system performance across cloud events and adjusts scheduling settings to take into account changes in workload. This flexibility ensures that resources are used wisely and that responses are quick.

2. | Material and Methods

2.1 | Concept and architecture of Scheduling Algorithm

2.1.2 | Analysis of Operating System Scheduling Criteria

The minimal completion time (CT), burst time (BT), arrival time (AT), and turnaround time (TAR) must all be achieved using an effective work scheduling technique. To ensure that jobs are processed within the anticipated time frame and in compliance with the Service Level Agreement (SLA), this process necessitates the use of schedule-based algorithms to process the operation and execution time for all tasks. Scheduling algorithms are the method for allocating resources to activities in order to reduce waiting and execution time. (Aladwani, 2020). In order to efficiently manage jobs and allocate resources over a shared cloud infrastructure, scheduling algorithms are utilized in cloud computing. In a cloud setting, events or activities can be planned, according to (Puri and Venkatesh, 2023).

Scalability and fault tolerance issues are addressed in the scheduling a:roach for cloud computing events. Resource allocation, Load balancing, Adaptive Scheduling, and dynamically modifying workloads during cloud events are the primary problems with present a:roaches. Assessment of Experiments. The scheduling method in Figure 2, is evaluated using extensive simulations and real-world trials. The algorithm is tested alongside current scheduling techniques in a range of workload and failure scenarios. It is possible to tackle the problem of job scheduling in the cloud by meeting a variety of scheduling objectives, such as cost, completion time, availability, service level agreement, and energy consumption. The system also needs to be able to handle all user requests with high performance and a guarantee of quality of service (QoS). (Aladwani, 2020). CPU scheduling decisions criteria may take place when a process as depicted in Figure 1.

i. Switches from running to waiting state

ii. Switches from running to ready state

iii. Switches from waiting to ready

iv. Terminates

First, we calculate the completion time for all the cloudlets or jobs to determine the cloudlet with longest processing time and assigned it to the corresponding resource with fastest processor. (Khurma, 2018).

Then we ma:ed other cloudlets according to their arrival time.

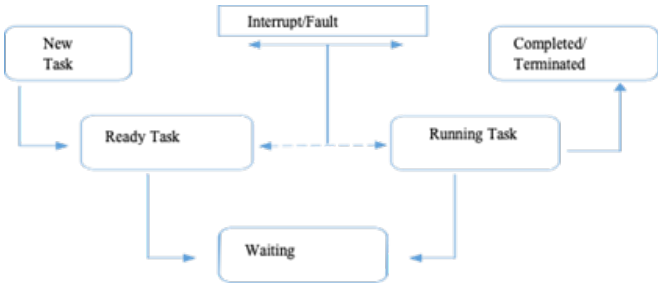

Figure 1 | Scheduling Task/Jobs Process Criteria

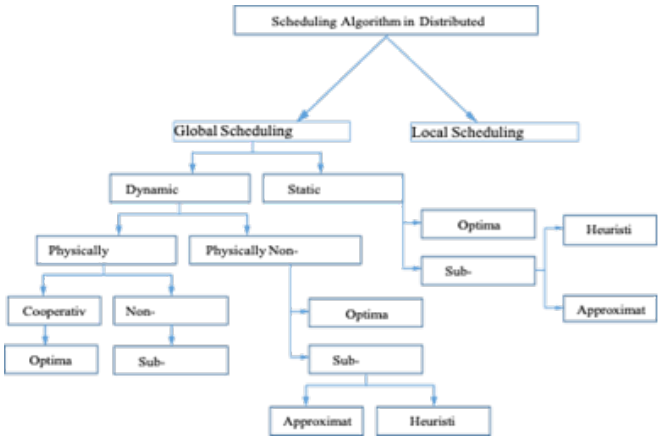

Figure 2 | Taxonomy of scheduling algorithm

Current Status Data Analysis and Findings

Consider the following existing Fault-Tolerance techniques:

i. Retrying: The unsuccessful job is attempted again on the same resource.

ii. Alternative reSource | A different resource is used to try again the failed task.

iii. Check-pointing: If a task fails, processing resumes from the last checkpoint that was saved rather than from the beginning.

iv. Replication: Various copies of the same task are handled concurrently on various resources.

Each a:roach has advantages and disadvantages of its own. The retrying a:roach is currently being used because it is the easiest. Both the retrying and the alternate resource strategies are time consuming because, when a process fails, they both involve starting over from scratch, which takes more time. The replication process

Network error: Cloud computing has a number of intrinsic issues because of its highly distributed structure, resource variability, and passive scale of operation. This means that a range of issues could arise in the cloud environment, leading to malfunctions and poor performance. The primary categories of flaws are shown below.

Physical defects: Also known as physical faults, they are defects that mostly impact hardware elements including CPUs, memory, storage, and power outages.

Process errors: Inadequate processing capacity, software bugs, a shortage of resources, etc. can all lead to process errors.

Service expiry fault: When a resource's service time runs out while it's being used by an application that leased it, there is a service failure

In Figure 3 and Figure 4, proactive and reactive strategies are the two main methods used to achieve fault tolerance. These methods might be divided into other categories according to certain guidelines and operating procedures. The goal of proactive fault tolerance is to swap out malfunctioning parts for working ones. Software rejuvenation and pre-emptive migration are two strategies that are based on this concept. Software rejuvenation restarts the system in a clean state, whereas preemptive migration uses a feedback loop control mechanism to periodically examine the program's state. Another method that manages application instance failures when numerous instances of the same application are operating on different virtual machines is called self-healing. Conversely, reactive fault tolerance seeks to lessen the effect of a failure on the application. Exception handling, resubmission, replication, and checkpoint are a few of the various techniques within this technique. Task-level fault tolerance can be effectively achieved using checkpointing, which enables the failed work to be restarted using the most recent checkpoint state. The purpose

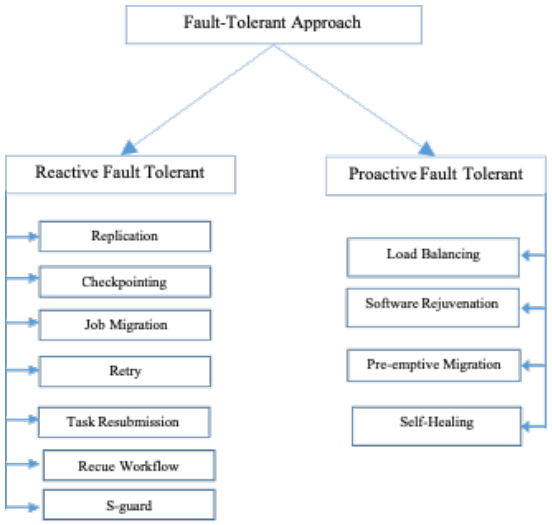

Figure 3 | Fault Tolerant A:roach

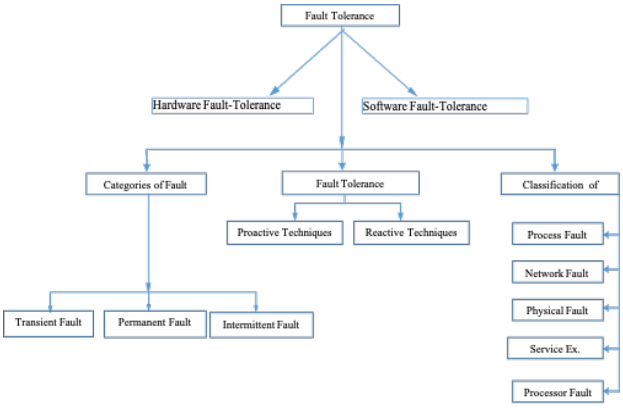

Figure 4 | Fault Tolerance Architecture

of replication is to provide system-level redundancy by maintaining cloned copies of tasks on other machines. Maintaining several copies of an update might be challenging when employing replication. It is necessary to resubmit the unsuccessful task to either the same computer or a different one.

2.2 | Errors and Deep Learning Precaution

In addition to the techniques currently employed to guarantee fault tolerance in cloud systems is adopted to enable users run their devices with resistant fault, fault tolerance algorithms in cloud systems have been detailed for a number of distributed systems, including mobile computing systems, Figure 5. Consequently, a thorough understanding of the problem and its challenges is given. It looks into how the underlying network topologies of Data Centers (DC) affect the fault tolerance mechanisms used by cloud computing systems. The assessment covers the most widely used network topologies for cloud systems in data centers as well as the fault tolerance techniques that may be applied to them. It looks into how the underlying network topologies of Data Centers affect the fault tolerance mechanisms used by cloud computing systems. The assessment covers the most widely used network topologies for cloud systems in data centers as well as the fault tolerance techniques that may be applied to them.

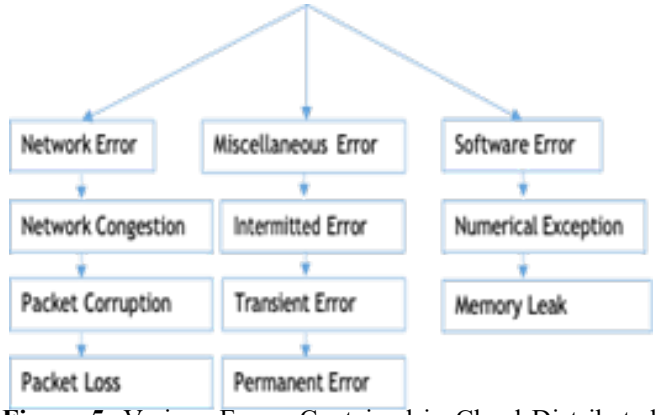

Figure 5: Various Errors Contained in Cloud Distributed System

Table 1 | Performance metrics and description

S/N	PERFORMANCE METRICS	DESCRIPTION
1	Throughput	The quantity of inquiries that a service provider can handle in a given amount of time.
2	System Scalability	The capacity of a system to function well in different size conditions.
3	System Resilience	The consistency of the system's operation over time, particularly at peak demands.
4	System Availability	The likelihood that a service provider will accept a request for service.
5	System Elasticity	A system's capacity to adjust to variations in its load
6	Response time (delay)	The interval of time between a service request and its fulfillment

7	Throughput	The quantity of inquiries that a service provider can handle in a given amount of time.

Table 1 illustrates the range of performance measures that can be employed to assess distinct aspects of cloud services.

2.3 | Practical A:lication and Prospects of Comprehensive Hybrid Scheduling Algorithm

In this paper, three cost-driven algorithms were proposed to enable service providers to optimize their profit: *Base Algorithm:* Reducing the quantity of SLA violations in order to maximize profit the algorithm aims to maximize profit by minimizing cost through the reuse of virtual machines (VMs) and servers with maximum available space. Additionally, it *maximizes profit by minimizing cost* through the reuse of *VMs and servers with lowest available space.*

With the help of this design, Figures 6 and 7, a reliable and fault-tolerant scheduling system is produced. It incorporates efficient resource allocation, both dynamic and static scheduling strategies, and fault tolerance techniques like replication and checkpointing. Through the integration of a feedback loop, the system consistently enhances and adjusts to fluctuating circumstances, guaranteeing dependable and effective functioning inside a cloud computing setting. Through virtualization, Figure 8 and Figure 9, the cloud computing paradigm offers consumers flexible, scalable services that are tailored to their needs on-demand. This is made possible by the motivations and contributions of numerous writers related to the scalability and fault-tolerant ability of the scheduling algorithm.

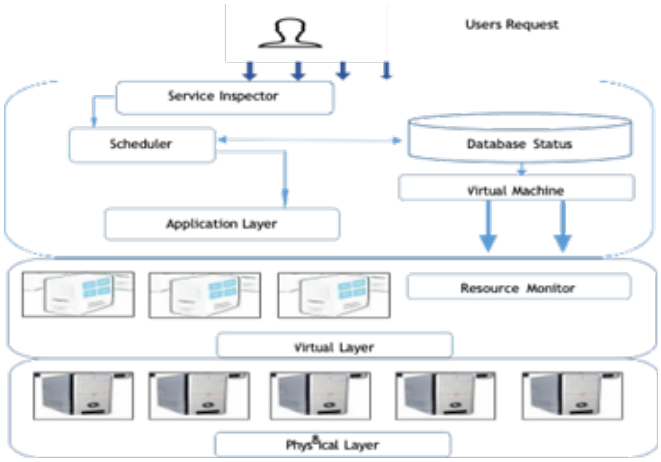

Figure 6 | The level architecture of cloud computing system

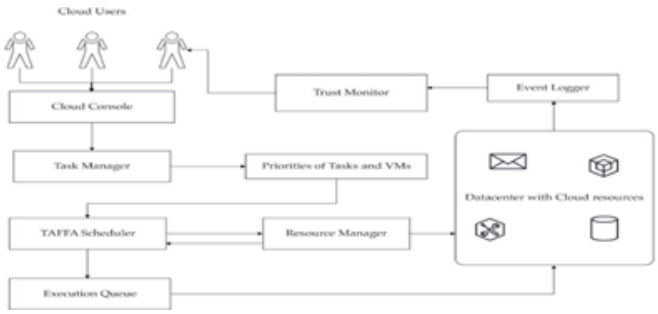

Figure 7 | System Architecture of Hybrid Scheduling Algorithms in Cloud Computing Environment

2.4 | Practical A:lication and Prospects Hybrid Scalable Fault Tolerant Technology

Current application status of comprehensive Scalable fault tolerant scheduling algorithm

In order to repeat the job by ensuring adherence to the procedures as a cure for the realtime scheduling algorithm mechanism, the replication feature was removed from the model. As demonstrated in Fault-tolerant Mechanism by Osuolale, (2020) and Liu *et al.,* (2018), where the authors suggested the use of checkpointing and replication technique as the standalone solution a:roach towards addressing fault tolerance issues in real time cloud environments, this was done in order to evaluate the performance of the developed model against an existing system. After putting the independent checkpoint-

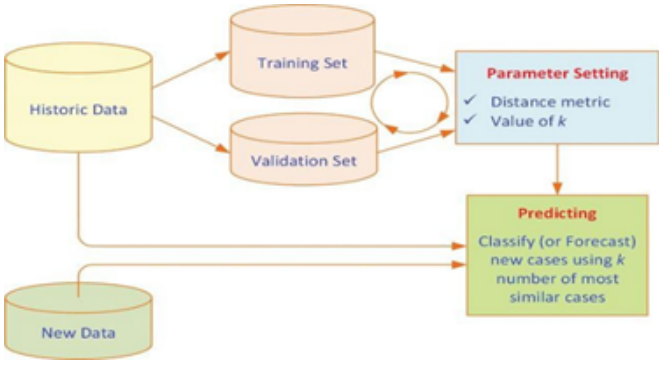

Figure 8 | General Predictive Analysis

ing, replication, and resource allocation mechanism that remained in the

developed model through the same tests as the current model, the outcomes of multiple simulations are displayed.

3 | Results and Discussion

3.1 | Performance Measure Mathematical Concept

Figure 10 depicts a 2 by 2 Matrix whose output is based on two classes.

(TP = True Positive; FP = False Positive; FN = False Negative; TN = True Negative.)

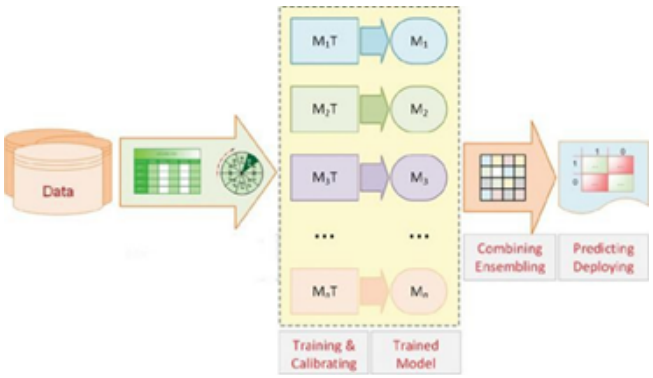

Figure 9 | Ensemble Prediction Modelling

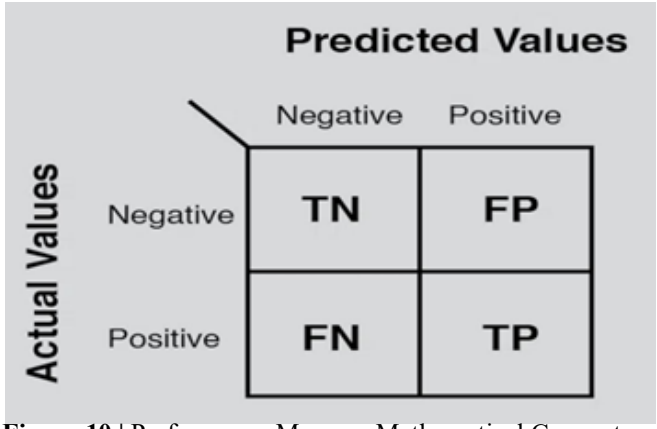

Figure 10 | Performance Measure Mathematical Concept

Accuracy = TP+TN/ TP+TN+FP+FN

F1-Score = 2∗ Precision ∗ Recall/Precision + Recall

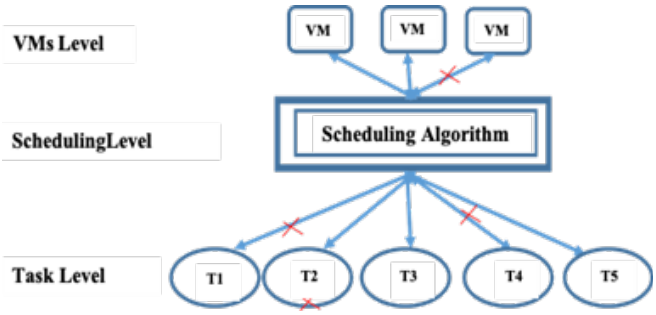

Figure 11 | Performance measure in Networking Concept

3.2 | Performance Measure In Networking Concept

By using an alternative resource allocation mechanism at every level, the hybrid fault- tolerant scheduling algorithm design, Figure 11, resolves all algorithm, virtual machine (VM), and task level failures. It also satisfies the Software Level Agreement (SLA) requirement, allowing users to request that input tasks be executed with accurate output results in the event of a network failure at any level. Implementing techniques that steer clear of the prerequisites for deadlock usually entails preventing deadlock in a system. The following are some typical strategies to avoid deadlock.

1. ***Avoidance:*** Graph of Resource Allocation (RAG): Resources and processes can be represented with a graph. Don't allocate a resource if doing so causes a cycle to form in the graph.
The Banker's Algorithm This algorithm determines if granting a request won't put the user in danger. The request can be granted if doing so maintains the system's security.

2. ***Identification and Recuperation:*** Use methods to identify cycles in resource allocation graphs and keep an eye out for deadlocks in the system. After detection, take corrective action, including sto:ing or reversing procedures.

3. ***Avoiding Mutual Exclusion***: Make sure that at least one resource is shareable, or non-blocking. Wait and Hold: Prevent processes from retaining resources while they wait for others by requiring

them to request all resources before execution begins.

Absence of preemption If resources are being held by processes while they wait for other resources, permit their forcible removal.

Wait in a circle: To avoid cyclical wait situations, enforce a rigorous ordering on resource acquisition. Resources must be requested by processes in a specific order.

4. ***Timeouts***: Put timeouts in place for resource requests. A process releases all resources it presently has and attempts again if it is unable to obtain a resource within the allotted period.

5. ***Strategies for Allocating Resources***: Employ a resource allocation technique that reduces the likelihood of stalemate, such as assigning resources according to priority or utilizing equitable queuing techniques. By putting these tactics into practice, a system's risk of stalemate can be greatly decreased. The particular needs and limitations of the system in question frequently influence the strategy selection.

Deadlock Prevention Concept

The concept of the hybrid fault-tolerant scheduling algorithm in this design (Figure 12), includes the main fault-tolerant mechanism techniques, such as checkpointing, replication, and replying. It also includes resource allocation to allow task execution to continue uninterrupted by the location of available resources to optimize workload and balance for accurate task execution by users of all cloud platforms.

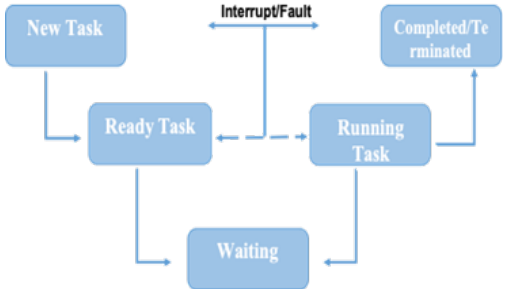

Figure 12 | Hybrid Deadlock Prevention Concept

4. | Conclusion

The key concepts of cloud computing, real-time cloud services, and hybrid scalable fault tolerance have all been thoroughly discussed in this study. The primary causes of the large costs incurred by cloud providers and clients are network breaches and delays. When waiting time problems, especially during real-time cloud events, breach their service level agreement (SLA), cloud service users usually move. Faults are the main cause of this delay. To solve this problem, existing fault-tolerant techniques including resource allocation, checkpointing, replication, and retrying were suggested in the context of real-time cloud computing. The new model underwent a series of tests after being hybridized and simulated using the CloudSim toolbox. The result shows that in the event of an error, the system may react by transferring user requests to dependable virtual machines (VMs) and resuming task execution in the final state without encountering a freeze. Because of this, systems that employ deadlock prevention techniques record less waiting times while completing tasks and avoiding mistakes.

Reference

Aladwani, T. (2020). Types of task scheduling algorithms in cloud computing environment. In R. da Rosa Righi (Ed.), Scheduling Problems. IntechOpen. https://doi.org/10.5772/intechopen.86873

Joundy, M., Elmougy, S., & Sarhan, S. (2015). Scheduling algorithms in cloud computing: A comparative study. *International Journal of Intelligent Computing and Information Science*, *15*(4).

Khurma, R. A., Aljarah, I., & Sharieh, A. (2018). The salp swarm algorithm: A comprehensive survey and its applications. In Proceedings of the International

Conference on New Trends in Computing Sciences (ICTS).

Kumari, P., & Kaur, P. (2021). A survey of fault tolerance in cloud computing. *Journal of King Saud University - Computer and Information Sciences*, *33*(10), 1159–1176. https://doi.org/10.1016/j.jksuci.2018.09.021

Liu, Y., Wang, L., Wang, X. V., Xu, X., & Zhang, L. (2018). Scheduling in cloud manufacturing: State-of-the-art and research challenges. *International Journal of Production Research*, *57*(15–16), 4854–4879. https://doi.org/10.1080/00207543.2018.1449978

Murad, S. A., Muzahid, A. J. M., Azmi, Z. R. M., Hoque, M. I., & Kowsher, M. (2022). A review on job scheduling technique in cloud computing and priority rule based intelligent framework. *Journal of King Saud University – Computer and Information Sciences*, *34*(6), 2309–2331. https://doi.org/10.1016/j.jksuci.2020.08.006

Osuolale A Festus, Adewale O. Sunday and Alese K. Boniface (2020). "Fault Mitigation in Real-Time Cloud Computing". The Journal of Computer Science and Its A:lications, 27, No 2, December, 2020. https://dx.doi.org/10.4314/jcsia.v27i2.6

Puri, A., Jose, J., & Venkatesh, T. (2023). Design, modeling, and analysis of memory allocation policies for rack-scale memory disaggregation. Research Square. https://doi.org/10.21203/rs.3.rs-2597744/v1