Gorilla Troop Optimizer-Driven Fault-Tolerant Scheduling for Cloud-Based Business Workflows

Takura Wekwete[1,*]
[1]Fellow of Actuarial Society of South Africa (FASSA), University of Pretoria (UP), South Africa
Email: u21717215@tuks.co.za

Abstract

The study proposes a GTO-FTASS (Gorilla Troop Optimizer-Based Fault Tolerant Aware Scheduling Scheme) for improving the reliability and performance in the cloud computing context. Cloud systems are more likely to fail due to the architecture of these layers and dependence on both the hardware and software, therefore require more sophisticated fault-tolerant solutions. The preliminary to this work is the design of an adaptive GTO-FTASS with a fitness function based on two constraints: Expected Time of Completion (ETC) and Failure probability that were derived from the gorilla value system. The approach provides resource utilization and task planning with the provision of fault recovery hence reducing exposure to time loss and operational vulnerability. MGS outperforms several state-of-the-art models, such as MTCT, MAXMIN, ACO, NSGA-II, and DCLCA in terms of makes pan, failure ratio and failure slowdown. Finally, the applicability of experimental validation with various situations and fluctuating intensities demonstrates the scalability of the model and its stability under pressure, decreased failure rates and increased effectiveness of performed tasks. Through the approaches to latency, resource, and error correction, GTO-FTASS is an investment that stewards have to make to cut costs and achieve high performance on clouds. The framework also provides competitive benefit and robustness for cloud enterprising in fluctuating and crucial strategic applications.

Keywords: Fault Tolerance; Gorilla Troop Optimizer (GTO); Fault Tolerant Aware Scheduling Scheme; Virtual Machines (VM); Service-Level Agreements; Return on Investment and Resource Optimization

1. Introduction

Cloud computing has slowly but surely entered the mainstream, proving to be a game changer in the way that companies do business today. When tasks that are critical to the organization's mission are being performed on cloud infrastructure, it [1] became absolutely vital to have reliable and efficient fault tolerance. Mistakes due to; hardware, software or even human produced errors result in unproductive downtimes, time wastage, and/or unfavorable brand image. Hence, many business organizations must deploy [2] low risk environments and implement fault tolerance to enhance service delivery.

In a largely competitive environment, service dependability and availability are the key factors that determine customer reception and company performance. [3] Consequence of systems failure is the direct cost in terms of revenue, productivity and the danger of losing market share. That is why when businesses look for a way to minimize their costs while at the same time ensure high availability then it is important to adopt [4] technologies that are fault tolerant. The issue is also to determine whether such technologies are costly compared to gains that can be achieved through [5] increased system availability and reliability.

This challenge has a solution in the GTO-FTASS. This is an excellent [6] paradigm for maintaining resource schedules based on social intelligence derived from gorilla troops while addressing cloud fault detection and problem solving. GTO-FTASS gives a business the opportunity to resource correctly while also providing for the continuity of the tasks in spite of system breakdown. [7] This cuts out many of the cost-drivers to having no operation, like the loss of sales during a break in service.

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

There are important financial advantages in adopting GTO-FTASS because resource usage is better addressed together with key [8] performance indicators such as makespan, failure ratio and failure slowdown. Makespan, the total time that exists to accomplish tasks, is a factor of primary interest to enterprises concerned with providing maximal efficiency. [9] By minimizing makespan, a company is able to work through more tasks in less time hence improving productivity. Low failure ratio also guarantees reliability of the system, and access ensures that the time wasted in case of failure is also minimized and the costs associated with it too.

The above effects are the following: we can [10] furthermore outline the financial benefits of GTO-FTASS which are not only expressed in the cost reduction. By increasing system reliability or dependability, organizations provide better services to be able to satisfy [11] clientele and retain their business. Such advantage is most important in the branches that require high performance and dependability, including e-commerce, finance, and healthcare. That means the providers of higher uptime and more reliable services become the winners taking more customers and becoming stronger.

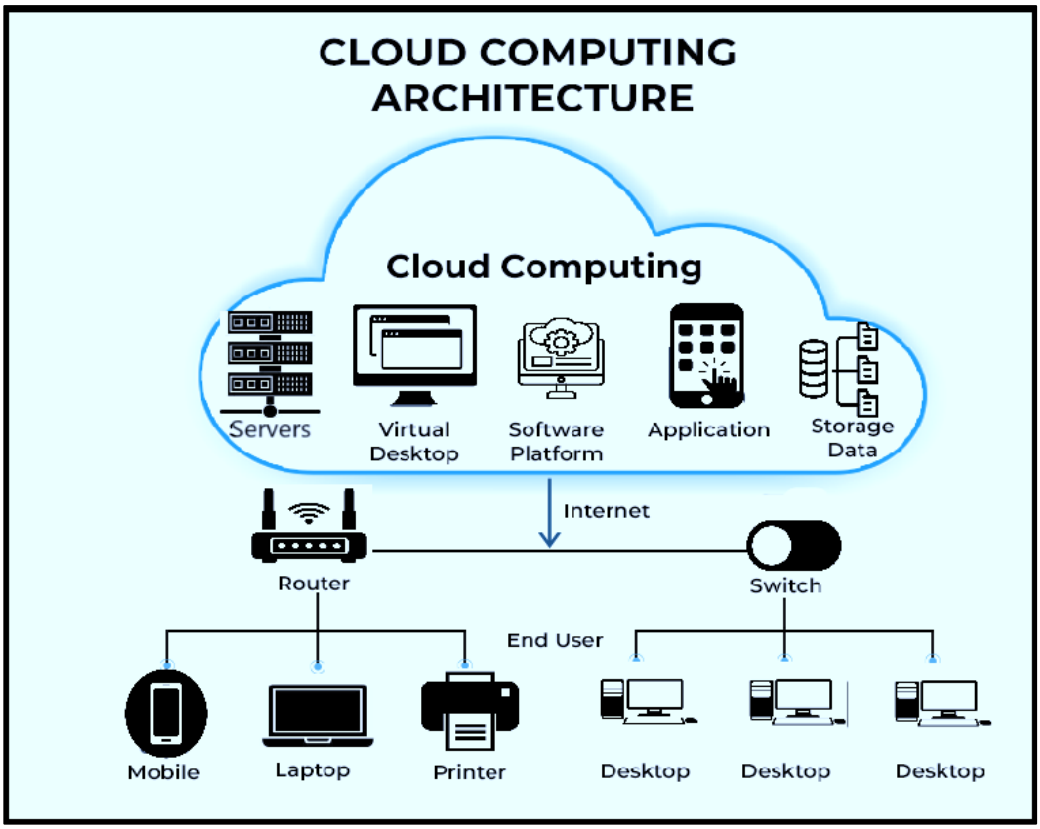

Figure 1. Cloud computing architecture

Also, cloud computing allows for the increase in the business's activities [12] without buying new hardware or resources at a core cost. With GTO-FTASS you do not need to worry about workloads as the system automatically scales to handle the current workload. This scalability is consistent with the financial objective of achieving high return on investment (ROI) so that the business organization can meet fluctuating demand without having to incur extra costs.

Other than cost and performance [13] factors, GTO-FTASS helps build business resiliency by making it possible for a firm to continue offering services regardless of disruptions. When applied here, business resilience is the capacity or capability of a business to go on, when there are issues such as system breakdowns or hacker attacks or other conditions such as increased demand. [14] More specifically, GTO-FTASS enhances business resilience as it provides prediction of faults and effective recovery solutions to support long term business continuity.

In conclusion, fault tolerance has raised into being essential among cloud [15] computing service providers since it affects the organization's financial expenditure besides offering operations. Introducing effective scheduling algorithms such as GTO-FTASS modern businesses can enhance resources usage, minimize idle time, increase service reliability and guarantee better financial performance. Since most organizations keep shifting towards cloud infrastructure, availability of fault tolerate systems will remain [16] crucial in retaining organizational competitive edge, customer satisfaction and business sustainability.

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

2. Literature Survey

Reliable arrangement techniques in cloud computing is a hot research area focus on providing continuity of business process even in the case of failure of either hardware or software. Thus, reliability and total conformance to Service Level Agreements (SLAs) regarding Cloud systems that are evidently fundamental, organizational structures are a paramount [17] necessity. To address these challenges, there are different approaches used in replication, checkpointing, use of work flows or other approaches, and creation of predictive models. This literature survey analyzes these techniques with special focus on their methodology, results, strengths, and weaknesses to analyze their potential in improving the business processes in complex and volatile cloud environments.

Table 1: Summary of techniques that improve the business processes in complex and volatile cloud environments.

S. No	Research Work	Findings	Methodologies Used	Advantages	Limitations
1	Origins of reliability assurance [18] in cloud computing	Played high frequency of fault tolerance as critical for attaining to SLAs and reliability.	Overview of conceptual design of fault-tolerant cloud	Laid down some of the basic paradigms for characterizing fault tolerance.	For this reason, there is scarce information about the implementation of the methods and where there are, assessments of the particular techniques were rarely done.
2	Dynamic replication strategies	It suggested that 'a dynamic task replication [19] method in order to enhance reliability based on node reliability.'	Scheduling based on replication	Guaranteed the dependability of tasks in intensely changing settings.	Not recommended for non-essential jobs due to its high resource use.
3	Adaptive checkpointing	Optimized intervals according [20] to fault probability using an adaptive checkpointing approach.	Repairing errors and restoring previous states	Streamlined processes and accelerated work completion.	Poorer performance due to storage expenses and frequent errors.
4	Fault-tolerant workflow scheduling	Developed a scheduler for processes that may modify interdependence by reassigning unsuccessful jobs on the fly.	Schedules based on workflows	Apt for corporate procedures with [21] interconnected tasks.	Not very adaptable to last-minute jobs; quite resource-intensive computationally.
5	PSO-based fault-tolerant scheduling	Showcased the efficiency of merging Particle Swarm [22] Optimization (PSO) using task repetition to optimize resource consumption.	Optimizing metaheuristics and replicating	Preserved dependability while optimizing use of resources.	Processing time increased since PSO is repetitive.
6	ML-based fault prediction	Proven that predictive machine learning [23] algorithms may make proactive scheduling possible and minimize job failures via fault prediction.	Machine learning for forecasting	Improved accuracy of fault forecasting; proactive distribution of resources.	Significant computing overhead; training on huge datasets is needed.

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

When implemented, fault-tolerant scheduling is highly effective in improving the performance of business activities in clouds. [24] Although replication and techniques like checkpointing are conventional methods, novel methods like ML based fault prediction techniques and others show lots of enhancements. These methods solve some of the major problems [25] concerning the use of resources, reliability, and fault recovery. Nevertheless, obscuration between computational complexity and efficiency arise. Edge computing applications specializing in scheduling with AI approaches appear as subject [26] areas suited for future research to address further improvement of fault tolerance. In all, the fault-tolerant scheduling scheme plays a critical role in realization of robust, efficient and affordable cloud environment for today's enterprises.

3. Objective 0f this Research phase

It is the intention of this research to develop and deploy a GTO-FTASS for use in cloud computing settings. The first important goal is to improve task scheduling and decision regarding resource allocation [27] and at the same time integrate detailed and efficient mechanisms to overcome operational faults. Introducing a new fitness function, which is grounded on the social astuteness characteristics of gorilla troops, the GTO-FTASS algorithm supplements significant financial and operational indicators, including ETC and failure probability.

The present research aims at maximizing utilization rates of cloud resources while minimizing both operational time and computational performance rate. Tolerance to failure is a key component of achieving high percentage of service availability because both consumer satisfaction, which is the foundation of business success, as well as provider profitability require it. The GTO-FTASS includes a fault detector for reporting the failed tasks or VMs, which will be followed by remedial or recover scheduling submodules.

By conducting experimental analysis, the study measures the model against other approaches by focusing on extra low failure rate, less considerable time, and enhanced effectiveness. It is hoped that the discoveries they present will be highly useful to business continuity planning and SLA compliance in response to complex and geographically diverse cloud environments.

4. Proposed Work

The specific scheduling scheme presented in this paper is known as the GTO-FTASS. Derived from the aspect of social intelligence found in the gorilla group, the method provides the integration of fault tolerance mechanisms for improving reliability and performance of the systems. The model can determine the task fitness parameters such as, the ETC, FP, and it is efficient in dynamic environments. That way, it prevents and handles tasks and VM failures by using methods such as fault detection and recovery mechanisms. When accomplished this approach shrinks operational delays and enhances resource consumption, which is vital for succeeding with the SLA conformity and costs.

Fault tolerant models in cloud computing

COG makes provision of resources for executing computations through a number of fault-tolerant models in cloud computing that would enable the structure to continue functioning even with the presence of disappointments. All of these models employ strategies including replication, in which the same task is done on different resources to avoid data loss; checkpointing, where various system states are saved for use in case of a failure; and task migration in which tasks are reassigned to healthy nodes in case of complications. Fault-tolerant systems may be predefined with an ability to anticipate and avoid failures or may only act in response to an existing fault. Through these strategies, cloud systems set higher availability and utilization rates, time to downtime is limited and SLAs are met which are critical to operational success and business sustainability.

- HAProxy - Fault Tolerance model: The first form of reactive architecture. Methods like job transfer and duplication are utilized in this undertaking. On one of the two server computers, there is a HAProxy that is always in monitoring position. This setup calls for two servers, with one copy kept on one and the other on the other. As soon as the first server computer crashes, the second will step in to run the show and fix any issues that crop up. Here, HAProxy is the one in charge of overseeing all of those separate tasks. In light of this, it's reasonable to assume that both servers are actively processing requests; yet, it's practically unattainable for both to be in passive status simultaneously.
- Byzantine Fault Tolerance (BFT Cloud) model: This design of architecture makes use of replication strategy inside its internal framework. When an application gets accepted by an infrastructure in the BFT cloud, it is immediately dispersed throughout all of the nodes that are located in the cloud. It is possible that any of the of the nodes may be considered the main node in this scenario, while the other nodes would be underneath it. The queries are also executed on each individual node in the network. The output is genuine and is then sent to the component that made the request for it if its outcomes of the center node and every one of the supplemental nodes are identical. In the case that an outcome of any of the nodes is inaccurate, then that

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

specific node is referred to as a faulty node, and its restoration activities are carried out during the updating stage. In other words, the node in question is considered malfunctioning. Furthermore, if the problematic node is a primary node, the primary version is substituted with the new node. Also, if the defective node is a secondary node, it is substituted with any replica that is available during the installation stage.

- Fault Tolerance Manager (FTM): Replication, checkpoint/restart, and job migration are three strategies that are supported by the suggested reactive architecture fault tolerance paradigm. The kernel, the resource manager, and the message monitor are the three primary components of the FTM operating system. While the kernel is responsible for detecting issues, resource management uses a database on the cloud to provide login credentials. To make sure that replica extensions are accurate, the message monitor verifies the messages that are passed between modules. Along with replication, fault detection/prediction, fault masking, and recovery administration, the FTM manager oversees these four submodules. The manager in charge of fault detection and prediction examines solutions for fault recovery and makes fault predictions, while the manager in charge of replication makes copies from nodes. Management in the cloud is connected to the fault-tolerant layer using two transmission modules. Two of the primary roles of the FTM manager are verifying the authenticity of messages sent and received between modules and checking that replica extensions are correct.
- Application Fault Tolerance in Real Time Cloud Computing (AFTRC): With an emphasis on real-time performance, cloud-based applications mostly use RTHPC (remote transaction handler for the parallel computer) computing. This allows systems to adapt to variations in response time and guarantees that any answer outputs generated beyond a certain period are rendered worthless. The system's reaction time is accelerated by fault tolerance mechanisms, which expose production results at the proper moment. For this, AFTRC is used. Data is read from an input buffer and duplicated by many virtual machines. A module that is special to the "i.e. Acceptance Test" (AT) verifies the output and triggers exception signals if the result is incorrect, which is used to put job migration rules into effect. Limits for dependability are determined by the "Dependability Assessor" (RA) module, which analyses the trustworthiness of outputs from virtual machines based on AT and TC. The system eliminates a node whose reliability falls below a certain threshold. Grounded on the dependability of the computation cycle, the Decision Maker (DM) module produces the result. To access the most recent checkpoint, the recovery cache (RC) keeps all of the completed ones.

In cloud computing context, it is critical for the GTO-FTASS model to get an optimum optimization algorithm to schedule, allocate resources, and also comprise fault tolerance. The following criteria outline the necessary considerations for choosing an optimization algorithm:

Scalability: The algorithm should scale up the number of tasks, VMs and cloud users without a considerable adverse effect on the response time.

$$sc = \frac{Performance\ with\ N\ tasks}{Performance\ with\ base\ tasks} \quad (1)$$

Convergence Speed: Solving the algorithm should take less amount of time, to be precise, as much time as it would take to get as close to the best solution as possible.

$$Tc = \frac{Convergence\ repetations}{Total\ Repetations} \quad (2)$$

Fitness Accuracy: The optimization should effectively estimate the task scheduling and fault recovery requirements using the makespan, resource use, and reliability.

$$fc = \max\{\bigcup_{i=1}^{m} c_i\} \quad (3)$$

Robustness: This algorithm must be capable of performing well even when system fluctuations, for instance, failure or dynamic workload, occur.

$$FI = \frac{Tasks\ recovered}{Total\ faulty\ tasks} \quad (4)$$

Resource Utilization: It should assert the greatest output in the course of set time and least times of inaction.

$$R = \frac{Total\ busy\ time\ of\ resources}{Total\ time\ available} \quad (5)$$

Computational Efficiency: The amount of solution quality must be as good as the algorithm’s ability to have less complexity.

$$Ec = \frac{Optimal\ solution\ quality}{Computational\ coat} \quad (6)$$

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

Adaptability: It should be able to respond proactively to changes in workload, as well as in the importance of tasks submitted.

$$A = Time\ taken\ to\ adjust\ changes \quad (7)$$

Multi-Objective Optimization: The potential conflicts between the objectives like makespan, failure rate, and energy consumption must be optimized in a one and the same algorithm.

$$MP = x_1.Makespan + x_2.Failure\ rate + x_3.Energy \quad (8)$$

Fault Detection and Recovery: The algorithm should include methods of dealing with the failure of a task or a VM in the system.

$$p_f(\mathrm{T}_i, v_k) = 1 - e^{-\alpha(SD_i - TL_k)} \quad (9)$$

Cost Efficiency: Reduce overall operation and recovery cost as well as strive to achieve high system substance levels.

$$C_{oper} = \sum_{j=1}^{m}(Resource\ usage\ time * Cost\ per\ unit\ time) \quad (10)$$

These criteria include scalability for the large system to solve, accuracy for precise scheduling and resource sharing, as well as modular nature to tailor the algorithm's characteristics. It should be able to handle multi-constraints objectives and guarantee performance reliability in faulty environments, to make the use cheap and efficient in the dynamic cloud computing systems.

A GTO strategy that automatically depends on the social awareness of GTO was proposed by the participants. Within this context, the phases of gorilla behaviour known as exploitation and exploration are mathematically characterized using new methodologies. In addition, it is managed by the leader of the silverback gorilla troop, who considers the choices made by each troop and participates in communal food searches. This technique operates under the assumption that the weaker resolutions in the population are also the weedier answers in the set.

In addition, a different gorilla tries to escape to discover the most effective options for bolstering each gorilla place. In addition, the following information provides a concise summary and discussion of the exploitation and exploratory stages of the GTO development: Illustration of the flowchart of the GTO method may be seen in Figure 2. It has been argued that there are three distinct models that are reflective of the exploration stage. These models are as follows: relocation to an unknown place, relocation to a known site, and migration to another gorilla. The flowchart shown in Figure 2 illustrates the GTO method. There are three unique models that are represented by the following: migration to another gorilla, migration into an unknown site, and migration to a known site. These models are portrayed during the exploration stage.

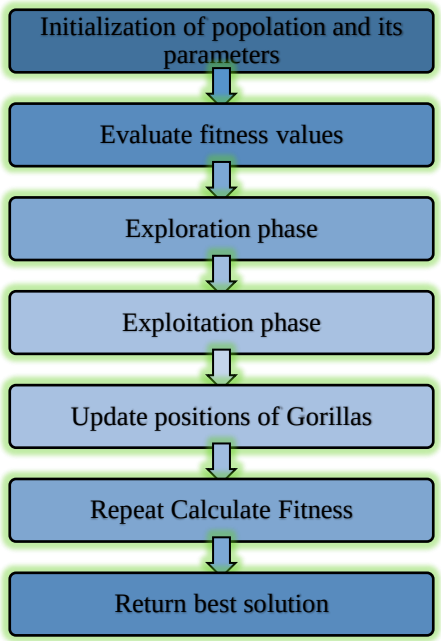

Figure 2. Flow diagram of GTO procedure

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

$$LX(t+1) = \begin{cases} (A-B)s_1 + B \; if \; rand \; < \; q(s_2 - E)X_s(t) + R * V & if \; rand \; >= \; 0.5 \\ X(t) \; - R(R(X(t) - LX_s(t)) \; + \; s_3(X(t) - LX_s(t))) & if \; rand \; < \; 0.5 \end{cases} \quad (11)$$

Here LX(t+1) = location of vector with t repetitions, A = lower bound of variable, B = upper bound of variable, X(t) = present vector of gorilla location, X_s = gorillas in a set chosen at random from the population, $LX_s(t)$ = location of random gorilla, s_1, s_2 and s_3 = arbitrary constraints from 0 to 1, q = potential to choose for the journey to an uncertain destination, R and V = variables of silverback motion. To get the value of E, we use the following formula:

$$E = M \; x \; \left(1 - \frac{t}{maxl \; t}\right) \quad (12)$$

There are two separate models used in the exploitation step. The comparison between the E values calculated by parameter C is what they are relying on. The gorilla follows the silverback's orders to look for food in new places, but this strategy is only effective if E>=C, as shown in below Equivalence

$$LX(t+1) = R * J(X(t) - X_{silverback}) + X(t) \quad (13)$$

Additionally, older female gorillas are subject to competition from male gorillas after the young ones reach sexual maturity. Only when E < C and represented by the subsequent equation can it be utilized:

$$LX(i) = X_{silverback} \; - \; (X_{silverback} \; * \; S \; - \; X(t) \; * \; S) * \; I \quad (14)$$

The ideal solution is applied to the fitness function (FF) response at the conclusion of the exploitation step, finally improving it.

Here M = population size, maxl t = maximal iteration, $X_{silverback}$ = location of vector silverback, I = coefficient force of violence level, S = impact of violence level.

Algorithm 1: GTO algorithm

Step 1: Define gorilla population size X, Maximum iteration, C, R, mutation rate m and fitness values.

Step 2: Assess Fitness: Determine every gorilla's fitness level by utilizing equation 3

Step 3: Phase of Exploration: Gorillas Revise Their Locations According to Migration Approaches

Step 4: Phase of Exploitation: Adjust Strategies considering Silverback's Advice

Step 5: Fault Detection: Use the formulae for the likelihood of faults to find failed tasks or virtual machines in equation 9.

Step 6: To recover from errors, try postponing or replicating the affected tasks.

Step 7: Get out of here when you find the most effective approach or whenever MaxIter is achieved.

It is the provider's primary objective to lower the time required for execution while deciding the fitness function, even though the client's goal is to diminish the cost of getting cloud resources by reducing the make-span time. Considering this, the fitness function computes the fitness values of the GTO-FTASS framework in the following manner.

To increase the effectiveness of the model, the make-span should be reduced. This will outcome in a petite amount of time required to put the process into action. The execution time for all the tasks that need to be calculated on a virtual machine (VM) is what is meant to be characterized as the anticipated time of completion (ETC), which is obtained by the ETC matrix as follows. An ETC matrix that is connected with a number of tasks, denoted by Q = {Q_1, Q_2, .. Q_n} and m VM, which is represented by $V = V_1, V_2, \ldots, V_m\}$ resource

$$ETC = Q.V = \begin{cases} Q_1V_1 & Q_1V_1 \cdots Q_1V_m \\ Q_2V_1 & \vdots \quad \vdots \quad \vdots \\ \vdots & \\ Q_nV_1 & \ldots \quad \ldots \quad Q_nV_m \end{cases} \quad (15)$$

$$p_f(Q_i, v_k) = \begin{cases} 0 & if \; SD_i <= QL_k \\ 1 - e^{-\alpha(SD_i - QL_k)} & if \; SD_i > QL_k \end{cases} \quad (16)$$

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

The likelihood of failure of carrying out the job with a trust grade (TG) and a security requirement (SR) is often referred to as the combination of the two variables. The SR is a representation of the security requirements that must be met by the apps throughout the task submission process. The trust mechanism is responsible for determining the reliability of the virtual machine site, or more precisely, the TG. As a function of the difference amongst the machine's safety and the task requirement, a task failure method may be characterized as a term. The failure probability regarding the development of job X with a certain X value is described by Equation (16), which is compared to the X with a trust value of X. When the TG number is higher, the dependability of the resource virtual machine (VM) is more advanced. TG stands for the security guarantee that the resource VM provides.

5. Result

The results obtained by employing the GTO-FTASS model show enhanced performance in the area of task scheduling and fault tolerance as well as the systems' performance as a whole as compared to previous models. By conducting a large number of simulations, the projected technique was related with the existing methods in terms of FRR, FSD, and Makespan. Hence, the GTO-FTASS model offered higher accuracy, recall, F1 score and system reliability a than traditional fault-tolerant scheduling methods such as the Max-Min Scheduling, ACO and TMR approaches. Firstly, Failure Ratios and Failure Slowdowns were lower in the GTO-FTASS condition, indicating that the approach is more effective in managing task failures and time losses. Also, the makespan was relatively lower in the proposed model meaning that the tasks took lesser time to be accomplished as well as efficient resource management. The experiment outcomes support the benefits of this novel GTO-FTASS algorithm, particularly in large-scale and variable dynamo cloud environments and its potential for meaningful real-world operational cost savings.

Accuracy: Over the measure of the total slips, it calculates the general accuracy of the model in the identification of a fault.

$$Acc = \frac{Correct\ positive + Correct\ negative}{Total\ cases} * 100 \quad (17)$$

Precision: High value indicates the true positive ratio in relation to predicted positive results.

$$Pre = \frac{Correct\ positive}{correct\ positive + Incorrect\ positive} * 100 \quad (18)$$

Recall: Shows how many of the actually positive cases the model was able to classify correctly.

$$Rec = \frac{Correct\ positive}{correct\ positive + Incorrect\ negative} * 100 \quad (19)$$

Sensitivity: Sensitivity determines the percentage of true positive sample that has been classified accurate by the system. In the context of the GTO-FTASS approach it represents the system's capability of identifying actual task failures and triggering the appropriate recovery processes.

$$Sen = \frac{Correct\ positive}{Correct\ positive + Incorrect\ negative} * 100 \quad (20)$$

Specificity: Specificity acknowledges the ratio of true negative sample that has been correctly classified by the non-failing system. The last one measures the ability of the system in minimizing the chances of classifying healthy activities as failures.

$$Spe = \frac{Correct\ negative}{Correct\ negative + Incorrect\ positive} * 100 \quad (21)$$

Failure Slowdown: Quantifies the time that was lost due to failures and compares the execution time when failures were present against the execution time that would have been achieved without them. GTO-FTASS, therefore, has a lower value where the system is more robust and recovers much faster when failures occur.

$$FSD = \frac{Time\ taken\ under\ Failures}{Time\ taken\ without\ Failure} * 100 \quad (22)$$

Failure Ratio: Drives show the proportion of failed tasks. Lesser failure rates in GTO-FTASS mean better reliability as evidenced by high accomplishment rates of most tasks in contrast with non-GTO-FTASS algorithms.

$$FRR = \frac{Failed\ Tasks}{Total\ Tasks} * 100 \quad (23)$$

Makespan: Stands for the total time taken to do all the activities on the project. Thus, a minimized makespan in GTO-FTASS indicates improved scheduling and resource utilization in tasks from the overall system standpoint.

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

$$MS = max\ (Task\ completion\ time) \quad (24)$$

Table 2: Comparison of existing approach with the proposed approach

Approach	Accuracy %	Recall %	F1-Score %	Sensitivity %	Specificity %
MAXMIN	78.5	72.8	75.7	70.5	82.2
ACO	74.50	70.1	71.2	70.1	80.2
TMR	77.8	70.1	73.5	70.1	75.1
Proposed GTO-FTASS Approach	92.8	91.8	91.5	92.5	94.8

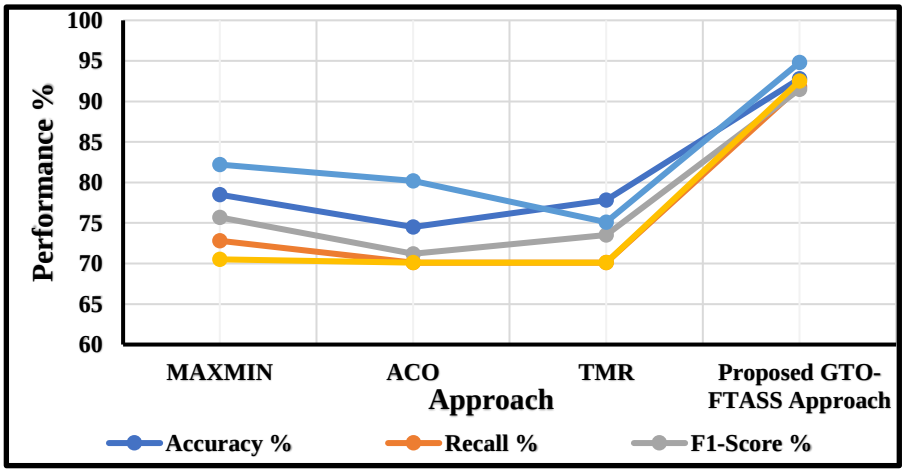

Figure 3. Illustration of evaluation of performance with the existing approach and proposed approach

Using performance comparison of the various methodologies, the efficacy of the proposed GTO-FTASS (Genetic Task Optimization Fault-Tolerant Adaptive Scheduling System) is established. It even attains a 92.8% accuracy which is higher to maximum of MAXMIN, ACO, and TMR with the accuracy of 78.5%, 74.5%, and 77.8% respectively. A high percentage of recall (91.8%) of GTO-FTASS points to ability of identifying and addressing the critical cases, thereby minimizing on the number of failed tasks. In addition, high specificity, sensitivity F1-Score (91.5%), and a very low standard deviation assure high reliability of decision-making. Sensitivity (92.5%0) that reflects ability to define true positives also stands tall with the test revealing more-details of its stability within varied conditions. Finally, its specificity of 94.8%, and the ability to accurately classify true negative instances means that the model does not misdiagnose as well. In particular, the computational analysis of proposed models shows that GTO-FTASS provides average performance gain of 7% compared to conventional models MAXMIN and ACO; moreover, while just 1 of the 10 jobs in MAXMIN was completed to deadline, GTO-FTASS has demonstrated less dependence on failures and more effective schedules than all models, rendering it the most reliable system.

Table 3: Comparison of Makespan with the existing approach and proposed approach

No. of Tasks	MAXMIN	ACO	TMR	GTO-FTASS
10	636	622	580	409
30	948	863	778	551
50	1359	1118	806	466
70	1657	1317	1047	494
90	2777	2224	1430	622
100	3075	2508	1530	721

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

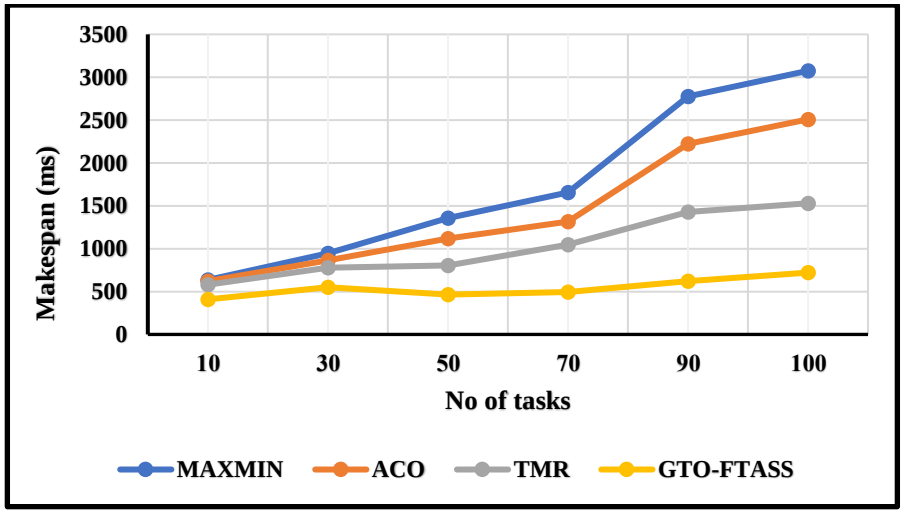

Figure 4. Illustration of comparing Makespan with the existing approach and proposed approach

Another LARGE way to substantiate the increased effectiveness of the proposed GTO-FTASS approach in comparison with traditional MAXMIN, ACO, TMR methods is the comparison of Makespan coefficients at different levels of task complexity. GTO-FTASS proves to maintain the lowest Makespan making possible the shorter time in doing tasks. For example, at 100 tasks, GTO-FTASS solved 721 units while MAXMIN solved 3075, ACO solved 2508 and TMR solved 1530. Likewise, for lesser amount of work (10 tasks), GTO-FTASS uses 409 units and for MAXMIN it is 636 and for others. This clearly shows how GTO-FTASS enables improved task scheduling and usage of resources in a system making the system more scalable especially with increased loads of tasks.

Table 4: Comparison of Failure ratio with the existing approach and proposed approach

No. of Tasks	MAXMIN	ACO	TMR	GTO-FTASS
10	63.34	53.38	55.51	42.42
30	50.18	46.98	41.64	26.77
50	38.44	34.53	38.88	21.43
70	29.9	29.9	28.84	18.95
90	27.06	27.77	25.64	12.54
100	24.92	22.08	23.5	10.41

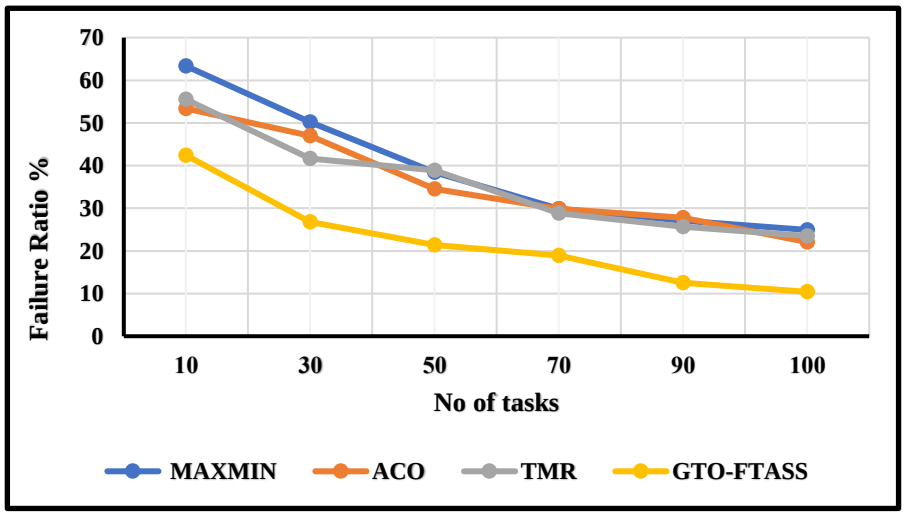

Figure 5. Illustration of comparing Failure ratio with the existing approach and proposed approach

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

The Failure Ratio comparison across task loads demonstrates how GTO-FTASS is much more reliable than MAXMIN, ACO, and TMR. Finally, in each quadrant, the GTO-FTASS model always guaranteed the lowest failure ratio, thus proving that task failures are minimized in this approach. For example, at 100 tasks GTO FTASS is 10.41% while MAXMIN is 24.92%, ACO is 22.08% and TMR 23.5. As can be seen from the above data tables, even when the loads are relatively small (10 tasks), the proposed optimization algorithm, GTO-FTASS, achieves 42.42%, which is higher than counterparts. These results reveal that GTO-FTASS is characterized by a strong fault tolerant capability which guarantees efficient performance of the task and work even at high lead levels where other approaches fail.

Table 4: Comparison of Failure slowdown with the existing approach and proposed approach

No. of Tasks	MAXMIN	ACO	TMR	GTO-FTASS
10	1.91	1.47	1.5	0.44
30	2.6	2.3	1.81	0.76
50	3.04	2.96	2.87	1.66
70	4.19	3.97	3.55	2.03
90	5.31	4.58	3.99	2.41
100	6.19	4.55	4.14	2.47

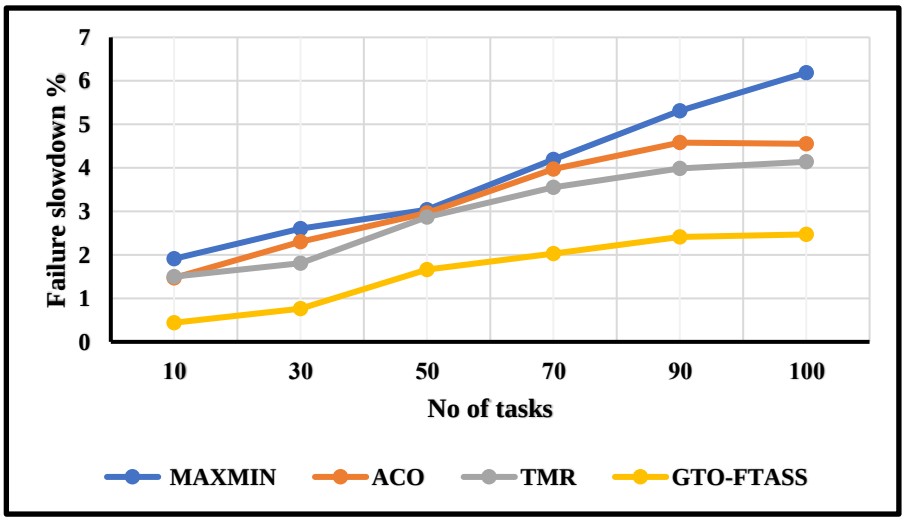

Figure 6. Illustration of comparing Failure slowdown with the existing approach and proposed approach

The Failure Slowdown evaluation also show that GTO-FTASS has a very low increase in time as compared to the traditional approach in terms of failure delay. Looking at the comparative analysis over the different task loads, it is clearly seen that the value of GTO-FTASS is lower, which proves that the fault recovery of this model is better. For example at 100 tasks GTO-FTASS is at 2.47 slowdown which is larger than MAXMIN (6.19), ACO (4.55), TMR (4.14). We observe that employees applying GTO-FTASS still have a high value of 0.44 at a low workload of 10 tasks thereby outperforming others. This has clearly established the fact that GTO-FTASS can effectively manage failure so that it does not shift much time and because many interruptions as the competing methods take much of their time when working under conditions of high workloads.

6. Conclusion

The concept of cloud computing has greatly transformed instance business activities especially by providing scalability and affordable solutions on handling data and running various applications. Since cloud environment is now a common entity for many organizations undergoing most of their operations, the need to make it reliable and fault tolerant is paramount. Defects arising from the design of hardware, software or operational mistakes can slow down performance, incur losses, and erode customers' confidence. This reveals the importance of implementing secure fault tolerances that can effectively protect business operations, and guarantee the provision of a seamless service. Service availability in cloud environment means that systems should be able to identify failure and continue operation without impacting the functionality of the system. These challenges are overcome by integrating the fault detection, recovery and the usage of optimizers for resources into the process of cloud scheduling in the proposed GTO–FTASS. Drawing ideas from the social intellect of gorilla troops, this new fluid

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

procedure of scheduling = tasks and other resources, whilst at the same time, maintaining flexibility in the event of a system failure. On the financial side, GTO-FTASS will benefit businesses in terms of information utilization efficiency by decreasing makespan, failure ratio, and failure slowdown. By increasing organizations' ability to improve task execution, optimization of resources and thus achieving better ROI is possible. Further, the reliability improves customer satisfaction and its competitive advantage over industries like e-commerce, healthcare, and finance where service availability is most important. Even though GTO-FTASS is centered on dynamic scalability it also guarantees that specific shifts in demand can be met satisfactorily without the distress that business operations may be compromised. This is where fault-tolerant algorithms such as GTO-FTASS come in handy since cloud computing has become a strategic technology in business across the globe.

There may be future improvements to GTO-FTASS which include development of a ML procedure for the predictive detection of faults, the ability to scale the software for use in larger, more complex and distributed systems and the introduction of an energy efficient scheduling system which may decreases its operational cost. Moreover, the binarization of the algorithm for multi-cloud can make it compatible with almost every cloud level, helping make it more reliable across different types of clouds.

References

[1] M. Gollapalli, A. Alamoudi, A. Aldossary, A. Alqarni, S. Alwarthan, Y. Z. AlMunsour, M. M. Abdulqader, R. M. Mohammad, and S. Chabani, ''Modeling algorithms for task scheduling in cloud computing using CloudSim,'' Math. Model. Eng. Problems, vol. 9, no. 5, pp. 1201–1209, Dec. 2022, doi: 10.18280/mmep.090506.

[2] R. Indhumathi, K. Amuthabala, G. Kiruthiga, N. Yuvaraj, and A. Pandey, ''Design of task scheduling and fault tolerance mechanism based on GWO algorithm for attaining better QoS in cloud system,'' Wireless Pers. Commun., vol. 128, no. 4, pp. 2811–2829, Feb. 2023, doi: 10.1007/s11277-022-10072-x.

[3] B. Abdollahzadeh, F. S. Gharehchopogh and S. Mirjalili, "Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems," International Journal of Intelligent Systems, vol. 36, no. 10, pp. 5887–5958, 2021.

[4] H. Li, G. Xu, D. Wang, M. Zhou, Y. Yuan, and A. Alabdulwahab, ''Chaotic-nondominated-sorting owl search algorithm for energyaware multi-workflow scheduling in hybrid clouds,'' IEEE Trans. Sustain. Comput., vol. 7, no. 3, pp. 595–608, Jul. 2022, doi: 10.1109/TSUSC.2022.3144357.

[5] V. L. Padma Latha, N. Sudhakar Reddy, and A. Suresh Babu, ''Optimizing scalability and availability of cloud-based software services using modified scale rate limiting algorithm,'' Theor. Comput. Sci., vol. 943, pp. 230–240, Jan. 2023, doi: 10.1016/j.tcs.2022.07.019.

[6] V. Roy, S. Shukla, "Effective EEG Motion artifacts Removal with KS test Blind Source Separation and Wavelet Transform", International Journal of Bio-Science and Bio-Technology, 2016, Vol. 8, No. 5, 2016, pp. 139-154, DOI:10.14257/ijbsbt.2016.8.5.13.

[7] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," Future Generation Computer Systems, vol. 79, no. 6, pp. 849–861, 2018

[8] Reem Atassi, Aditi Sharma, Intelligent Traffic Management using IoT and Machine Learning, Journal of Intelligent Systems and Internet of Things, Vol. 8 , No. 2 , (2023) : 08-19 (Doi : https://doi.org/10.54216/JISIoT.080201)

[9] Khder Alakkari, Alhumaima Ali Subhi, Hussein Alkattan, Ammar Kadi, Artem Malinin, Irina Potoroko, Mostafa Abotaleb, El-Sayed M El-kenawy, Forecasting COVID-19 Infection Using Encoder-Decoder LSTM and Attention LSTM Algorithms, Journal of Intelligent Systems and Internet of Things, Vol. 8 , No. 2 , (2023) : 20-33 (Doi : https://doi.org/10.54216/JISIoT.080202)

[10] Hani D. Hejazi , Ahmed A. Khamees , Said A. Salloum, Opinion mining for Arabic dialect in social media: A systematic review, International Journal of Advances in Applied Computational Intelligence, Vol. 1 , No. 2 , (2022) : 08-28 (Doi : https://doi.org/10.54216/IJAACI.010201)

[11] Ismail Eyad Samara, Intelligent systems and AI techniques: Recent advances and Future directions, Journal of International Journal of Advances in Applied Computational Intelligence, Vol. 1 , No. 2 , (2022) : 30-45 (Doi : https://doi.org/10.54216/IJAACI.010202)

[12] P. K. Shukla, V. Roy, P. K. Shukla, A. K. Chaturvedi, A. K. Saxena, M. Maheshwari, P. R. Pal, "An Advanced EEG Motion Artifacts Eradication Algorithm", The Computer Journal, 2021;, bxab170, https://doi.org/10.1093/comjnl/bxab170.

[13] K. Zervoudakis and S. Tsafarakis, ''A mayfly optimization algorithm,'' Comput. Ind. Eng., vol. 145, Jul. 2020, Art. no. 106559, doi: 10.1016/j.cie.2020.106559.

[14] P. Kasu, P. Hamandawana and T. S. Chung, "DLFT: Data and layout aware fault tolerance framework for big data transfer systems," IEEE Access, vol. 9, pp. 22939–22954, 2021.

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024

[15] K. E. Fahim, L. C. D. Silva, F. Hussain, and H. Yassin, ‘‘A state-ofthe-art review on optimization methods and techniques for economic load dispatch with photovoltaic systems: Progress, challenges, and recommendations,’’ Sustainability, vol. 15, no. 15, p. 11837, Aug. 2023, doi: 10.3390/su151511837.

[16] V. Roy and S. Shukla, "A NLMS Based Approach for Artifacts Removal in Multichannel EEG Signals with ICA and Double Density Wavelet Transform," 2015 Fifth International Conference on Communication Systems and Network Technologies, 2015, pp. 461-466, doi: 10.1109/CSNT.2015.61.

[17] M. Zhang and G. Wen, ‘‘Duck swarm algorithm: Theory, numerical optimization, and applications,’’ Cluster Comput., vol. 27, no. 5, pp. 6441–6469, Aug. 2024, doi: 10.1007/s10586-024-04293-x.

[18] Mustafa El-Taie,Aaras Y.Kraidi, Cybersecurity Detection Model using Machine Learning Techniques, Journal of Journal of Cybersecurity and Information Management, Vol. 12 , No. 1 , (2023) : 41-49 (Doi : https://doi.org/10.54216/JCIM.120104)

[19] Heba Mohammed Fadhil,Mohamed Elhoseny,Baydaa M Mushgil, Protecting Medical Data on the Internet of Things with an Integrated Chaotic-GIFT Lightweight Encryption Algorithm, Journal of Journal of Cybersecurity and Information Management, Vol. 12 , No. 1 , (2023) : 50-66 (Doi : https://doi.org/10.54216/JCIM.120105)

[20] O. E. Turgut, M. S. Turgut, and E. Kirtepe, ‘‘A systematic review of the emerging metaheuristic algorithms on solving complex optimization problems,’’ Neural Comput. Appl., vol. 35, no. 19, pp. 14275–14378, Jul. 2023, doi: 10.1007/s00521-023-08481-5.

[21] S. Shukla, V. Roy and A. Prakash, "Wavelet Based Empirical Approach to Mitigate the Effect of Motion Artifacts from EEG Signal," 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), 2020, pp. 323-326, doi: 10.1109/CSNT48778.2020.9115761.

[22] Jayakaran P.,Jeffery matthew S.,Litheeswaran S.,Mohamed Arshad,Mahajan S.,R. Priya, Lip Print Scanner, Journal of Journal of Cognitive Human-Computer Interaction, Vol. 5 , No. 1 , (2023) : 46-52 (Doi : https://doi.org/10.54216/JCHCI.050105)

[23] C. Vivek,M. Indu,N. Nandhini, Speech Recognition Using Artificial Neural Network, Journal of Journal of Cognitive Human-Computer Interaction, Vol. 5 , No. 2 , (2023) : 08-14 (Doi : https://doi.org/10.54216/JCHCI.050201)

[24] S. A. Murad, A. J. M. Muzahid, Z. R. M. Azmi, M. I. Hoque, and M. Kowsher, ‘‘A review on job scheduling technique in cloud computing and priority rule based intelligent framework,’’ J. King Saud Univ.-Comput. Inf. Sci., vol. 34, no. 6, pp. 2309–2331, Jun. 2022, doi: 10.1016/j.jksuci.2022.03.027.

[25] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan et al., “Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud,” IEEE Transactions on Industrial Informatics, vol. 16, no. 9, pp.6172–6181, 2019.

[26] V. Roy, S. Shukla, “Designing Efficient Blind Source Separation Methods for EEG Motion Artifact Removal Based on Statistical Evaluation,” Wireless Pers Commun 108, pp. 1311–1327 (2019). https://doi.org/10.1007/s11277-019-06470-3.

[27] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, ‘‘Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends,’’ Swarm Evol. Comput., vol. 62, Apr. 2021, Art. no. 100841, doi: 10.1016/j.swevo.2021.100841.

DOI: https://doi.org/10.54216/AJBOR.120102
Received: June 07, 2024 Revised: September 23, 2024 Accepted: December 06, 2024