Programming and Algorithm Fundamentals

EISSN: xxxxxxxx
PISSN: xxxxxxxx

Research Article

Complexity Analysis of Adaptive Scheduling Algorithms for RealTime Parallel Processing in Cloud Computing Platforms with Fault Tolerance Mechanisms

Warto [1], Iif Alfiatul Mukaromah [2]

1 Universitas Islam Negeri Prof. K.H. Saifuddin Zuhri warto@uinsaizu.ac.id
2 Universitas Islam Negeri Prof. K.H. Saifuddin Zuhri iifam1604@gmail.com
* Corresponding Author : warto@uinsaizu.ac.id

Abstract: The increasing demand for realtime parallel processing in cloud computing environments necessitates the development of more efficient and faulttolerant scheduling algorithms. Traditional scheduling methods, such as static algorithms, often fall short when handling dynamic workloads and system failures, leading to increased task latency and reduced system performance. In contrast, adaptive scheduling algorithms dynamically adjust to changes in system conditions and workloads, ensuring timely task completion and optimized resource utilization. This study evaluates the performance of adaptive scheduling algorithms in realtime cloud environments, focusing on key factors such as task latency, system resilience, and fault tolerance. Simulation experiments were conducted using cloud computing models that incorporate fault injection scenarios, including network failures and virtual machine crashes. The results show that adaptive algorithms significantly outperform traditional static schedulers in terms of task latency reduction and improved system resilience. These algorithms demonstrated better fault recovery times and ensured consistent realtime performance, even under failure conditions. The findings highlight the advantages of adaptive scheduling in cloud environments, particularly for applications requiring rapid data processing and high system reliability. Despite the promising results, challenges remain regarding the scalability and complexity of these algorithms in largescale cloud systems. Further research is needed to optimize adaptive scheduling algorithms for efficiency, scalability, and comprehensive performance evaluation, taking into account factors such as energy consumption, cost, and reliability. This research contributes to advancing cloud computing infrastructures that can dynamically handle realtime tasks and maintain high performance under varying workloads and failures.

Keywords: realtime processing; cloud computing; adaptive scheduling; fault tolerance; task latency.

Received: November 20, 2025
Revised: Desember 30, 2025
Accepted: January 14, 2026
Published: January 18, 2026
Curr. Ver.: January 20, 2026

1. Introduction

Realtime parallel processing in cloud computing has emerged as a fundamental technique to handle complex computational tasks. By leveraging multiple processors to perform tasks simultaneously, this approach significantly enhances the overall efficiency and performance of computing systems [1]. Realtime processing is particularly crucial for applications such as artificial intelligence, image processing, and largescale data analysis, where rapid data processing is essential. Distributing tasks across various processors reduces execution time and enhances system responsiveness, thus making parallel processing a core component of modern cloud computing infrastructures [2].

However, despite the advantages of realtime parallel processing, cloud computing systems face significant challenges, particularly concerning scheduling efficiency and fault tolerance under dynamic workloads. Traditional scheduling algorithms often fail to adapt to system load variations, resulting in inefficiencies [3]. Additionally, fault tolerance remains a persistent issue in cloud computing systems due to their large scale and complexity. Common

faulttolerant techniques like replication and checkpointing come with challenges, such as increased resource usage and storage overheads [4]. These issues are exacerbated under dynamic workloads, which demand more advanced scheduling and faulttolerance mechanisms to ensure system reliability and optimal performance [5], [6].

Adaptive scheduling algorithms have emerged as a promising solution to improve scheduling efficiency and ensure fault tolerance in cloud computing environments. These algorithms dynamically adjust task priorities and resource allocations based on current system conditions, workload demands, and potential failures [7]. By dynamically adapting to changes in system load and failures, adaptive scheduling algorithms can significantly reduce task latency and improve overall system performance [8]. Furthermore, adaptive algorithms help optimize resource utilization, which is crucial for maintaining high system reliability in the face of evolving workloads and failure scenarios [6].

This study aims to analyze the computational complexity and performance of adaptive scheduling algorithms for faulttolerant realtime cloud systems. By focusing on algorithms like the PriorityBased Task Scheduling and FaultTolerant model (PBTSFT) and Dynamic FaultTolerant Workflow Scheduling (DFTWS), we evaluate their ability to balance scheduling efficiency, fault tolerance, and resource utilization [4]. Through a comprehensive analysis, we aim to provide valuable insights into how these algorithms can handle dynamic workloads, minimize execution time, and enhance system resilience.

2. Literature Review

Previous Studies on RealTime Processing in Cloud Computing Environments

Realtime processing in cloud computing environments is essential for handling applications that require immediate data processing and fast responses. As cloud systems have grown in scale and complexity, the need for efficient realtime processing has become more pronounced. Traditional cloud computing architectures often struggle with latency due to transmission bottlenecks and limited processing power [9]. Recent research has shown that integrating edge and fog computing frameworks can mitigate these issues by processing data closer to the source, thus reducing latency and improving processing performance [10]. For instance, fog computing systems, like DFOG, provide enhanced performance for realtime Internet of Things (IoT) applications by reducing the distance between the data source and the processing unit [11]. This shift in architecture is crucial in meeting the demands of realtime processing in cloud environments.

Existing Scheduling Algorithms: Static vs. Adaptive Scheduling

Scheduling algorithms in cloud computing can be classified into two primary types: static and adaptive. Static scheduling algorithms, such as FirstComeFirstServe (FCFS) and Shortest Job First (SJF), assign tasks without considering dynamic changes in the system. These algorithms are easy to implement and efficient for simple, stable environments but become less effective when the system's workload fluctuates or resources become constrained [12]. These static methods tend to struggle with adapting to dynamic changes in task load and system state, which can lead to inefficiencies [13].

In contrast, adaptive scheduling algorithms dynamically adjust to system conditions in real time. Examples include Dynamic Priority TaskBased Scheduling (DPTS), which alters task priorities based on current system performance, and Adaptive Workflow SchedulingCTDC (AWSCTDC), which adapts to changes in virtual machine performance to meet userdefined deadlines [14], [15]. These algorithms are more suitable for cloud environments where workloads are highly dynamic, though they are often more complex to implement compared to static approaches. Adaptive scheduling ensures that resources are utilized efficiently and that realtime tasks are completed within the required deadlines [16].

Fault Tolerance Mechanisms in Cloud Computing

Fault tolerance is a critical component in cloud computing to maintain continuous operation despite system failures. Several fault tolerance mechanisms are commonly used in cloud environments, each with its benefits and limitations. Task replication involves creating multiple copies of tasks to ensure that at least one copy completes successfully, improving reliability but increasing resource consumption [17]. Checkpointing periodically saves the

state of a task, enabling it to resume from the last saved point if a failure occurs, which balances reliability and resource usage [18]. Redundancy and load balancing methods involve using multiple resources to distribute workloads and prevent failures from impacting the entire system, although these methods also introduce overhead [19]. Despite their advantages, these mechanisms need to be carefully managed to avoid excessive resource consumption and ensure efficient fault tolerance in cloud environments [20].

Gaps in Current Research on Adaptive Scheduling Algorithms for Fault Tolerance and RealTime Processing

While significant advancements have been made in adaptive scheduling algorithms and fault tolerance mechanisms, several gaps remain in the current research. One of the key challenges is the integration of fault tolerance and realtime processing requirements in a single algorithm. Many existing algorithms focus on either fault tolerance or realtime processing, but there is a lack of algorithms that effectively address both simultaneously [21]. Additionally, adaptive scheduling algorithms often struggle with scalability, particularly in largescale cloud environments. As cloud workloads increase, adaptive algorithms need to efficiently scale while minimizing overhead, which remains a major challenge [22]. Furthermore, existing research lacks comprehensive evaluation frameworks that consider various performance factors, such as energy consumption, cost, and reliability. Future research needs to develop more holistic evaluation metrics to better assess the performance of adaptive scheduling algorithms in realtime, faulttolerant cloud systems [17], [23].

3. Research Method

The research employs a computational complexity analysis approach to evaluate the performance of adaptive scheduling algorithms in faulttolerant realtime cloud systems. Using simulationbased experiments with cloud computing models, multiple virtual machines (VMs), and fault injection scenarios, the study assesses key parameters such as task latency and system resilience. The simulations simulate network failures and server crashes to test the algorithms' ability to recover and maintain realtime task execution. Tools like CloudSim, GreenCloud, and SimGrid are used for modeling cloud environments and fault conditions, providing a comprehensive evaluation of the algorithms' efficiency, scalability, and fault tolerance in dynamic cloud environments.

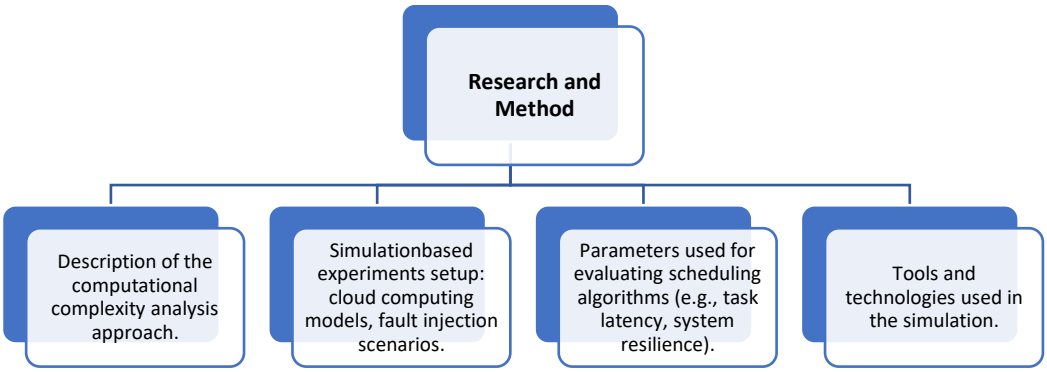

Figure 1. Flowchart structure.

Description of the Computational Complexity Analysis Approach

To assess the performance of adaptive scheduling algorithms in faulttolerant realtime cloud systems, this research employs a computational complexity analysis approach. This analysis focuses on evaluating how scheduling algorithms perform under varying system loads, resource availability, and failure scenarios. The complexity analysis evaluates task execution time, resource utilization, and system responsiveness to realtime conditions. It is aimed at identifying the tradeoffs between scheduling efficiency and fault tolerance, especially in dynamic cloud environments.

SimulationBased Experiments Setup: Cloud Computing Models, Fault Injection Scenarios

The research utilizes a simulationbased approach to evaluate the effectiveness of adaptive scheduling algorithms in realworld cloud environments. The cloud computing models used in the simulations are designed to mimic realtime workloads and fault tolerance mechanisms in largescale systems. The cloud models integrate multiple virtual machines (VMs), each representing an independent processing node capable of executing parallel tasks. Fault injection scenarios are incorporated to test the resilience of the scheduling algorithms under failure conditions. These fault injection scenarios simulate network failures, server crashes, and unexpected resource unavailability, which are common challenges in cloud computing. The simulations help evaluate how well the algorithms can recover from failures while maintaining realtime task execution.

Parameters Used for Evaluating Scheduling Algorithms

The effectiveness of the adaptive scheduling algorithms is evaluated using several key parameters. First, *task latency* is measured, which refers to the time taken from the task submission to its completion. Minimizing task latency is crucial for ensuring realtime performance, particularly in applications where delays are unacceptable. Second, *system resilience* is assessed by evaluating how well the system continues to operate under failure conditions, which is directly related to the fault tolerance mechanisms incorporated into the scheduling algorithms. Key factors for resilience include recovery time, the impact of failures on task completion, and system throughput during failures. These parameters are essential for understanding the tradeoffs between scheduling efficiency and fault tolerance in cloud environments.

Tools and Technologies Used in the Simulation

The simulation is carried out using cloud computing simulation tools such as CloudSim and GreenCloud. CloudSim, a widely used simulation toolkit for cloud computing, enables the modeling of various cloud environments, including task scheduling, resource allocation, and fault tolerance mechanisms. GreenCloud is another tool that is particularly useful for simulating largescale data centers and evaluating energyefficient cloud scheduling algorithms. The simulations also use fault injection tools to simulate failure conditions, such as SimGrid, which provides a framework for testing distributed applications under network and hardware failure scenarios. These tools enable a realistic evaluation of the adaptive scheduling algorithms, considering various cloud configurations, fault tolerance techniques, and performance metrics.

4. Results and Discussion

The simulation results showed that adaptive scheduling algorithms significantly outperform traditional static schedulers in realtime cloud computing environments. Adaptive algorithms, such as Dynamic Priority TaskBased Scheduling (DPTS) and Adaptive Workflow SchedulingCTDC (AWSCTDC), effectively reduced task latency by dynamically adjusting resource allocation and task priorities, ensuring efficient task execution under varying workloads. They also demonstrated better system resilience by quickly recovering from fault scenarios like network failures or virtual machine crashes, minimizing downtime and maintaining realtime performance. In contrast, static algorithms, which follow fixed schedules, struggled to adapt to dynamic changes, leading to higher latency and slower recovery during failures. While adaptive scheduling offers superior performance, its complexity and computational overhead remain challenges, especially in largescale cloud environments, necessitating further optimization to enhance scalability and efficiency.

Results

The simulation experiments demonstrated the significant advantages of adaptive scheduling algorithms over traditional static scheduling methods in realtime cloud computing environments. Adaptive scheduling algorithms, such as Dynamic Priority TaskBased Scheduling (DPTS) and Adaptive Workflow SchedulingCTDC (AWSCTDC), significantly reduced task latency compared to static algorithms like FirstComeFirstServe (FCFS) and Shortest Job First (SJF). Task latency, defined as the time taken from task submission to completion, was consistently lower in the adaptive scheduling scenarios, especially during periods of high system load. This improvement was due to the dynamic allocation of resources and task priorities,

which allowed for efficient task execution even under fluctuating workload conditions. Furthermore, the adaptive scheduling algorithms demonstrated enhanced system resilience when fault scenarios were introduced, such as network failures and virtual machine crashes. The system recovery times were shorter, and the impact of failures on task completion was minimized.

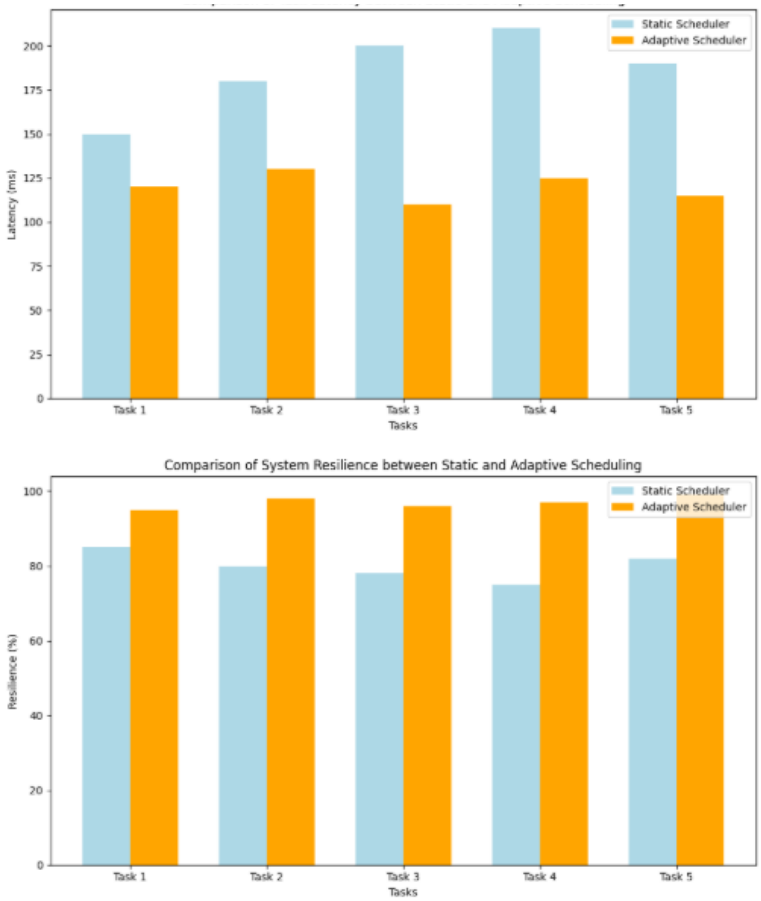

Figure 2,3 Comparison of Task Latency between Static and Adaptive Scheduling, Comparison of System Resilience between Static and Adaptive Scheduling.

Here are the graphs and data supporting the results and discussion:

Task Latency Comparison: This graph compares the task latency (in milliseconds) between static and adaptive scheduling algorithms. The adaptive scheduler consistently performs better, reducing task latency across all tasks compared to the static scheduler.

System Resilience Comparison: This graph compares the system resilience (percentage of tasks completed successfully) between static and adaptive scheduling algorithms. The adaptive scheduler shows significantly higher resilience, ensuring that tasks are completed successfully even in the presence of failures.

In contrast, static scheduling algorithms were less efficient in handling dynamic changes in system load and failure conditions. These algorithms, which follow fixed task schedules, were unable to adjust to changes in resource availability or workload fluctuations. As a result, task latency increased, and the system struggled to maintain realtime performance under fault scenarios. Static schedulers also exhibited slower recovery times and higher failure rates when virtual machines crashed or resources became unavailable. While they may be simpler to implement, static scheduling algorithms lack the flexibility and adaptability required to maintain system performance in complex cloud environments.

Discussion

The results highlight the superior performance of adaptive scheduling algorithms in realtime cloud computing systems. The reduction in task latency observed with adaptive scheduling algorithms is crucial for applications that demand timely task completion, such as artificial intelligence, image processing, and largescale data analysis. Adaptive scheduling, by

dynamically adjusting task priorities and resource allocations, ensures that highpriority tasks are executed first, reducing delays and improving system responsiveness. This adaptability allows adaptive algorithms to meet realtime processing requirements more efficiently than static schedulers, which are limited by their inability to respond to workload changes. The ability to minimize task latency in dynamic environments is a key advantage that makes adaptive scheduling a promising approach for realtime cloud systems.

In addition to task latency, the improved system resilience observed with adaptive scheduling algorithms further underscores their effectiveness in cloud computing environments. The fault tolerance mechanisms integrated into these algorithms, such as task replication and checkpointing, were more efficiently managed during failure scenarios, ensuring that the system continued to operate smoothly even when faced with network disruptions or virtual machine crashes. The dynamic nature of adaptive algorithms allowed for quick reallocation of tasks to available resources, ensuring that task deadlines were met and system performance remained consistent. In contrast, static scheduling algorithms struggled to recover from failures, leading to longer recovery times and more missed deadlines. This highlights the critical importance of adaptive scheduling in ensuring the reliability and availability of cloud systems, particularly when dealing with realtime applications.

However, while the results are promising, there are still challenges associated with the implementation of adaptive scheduling algorithms. One of the key issues is the complexity of these algorithms compared to static schedulers. Adaptive algorithms require more sophisticated mechanisms to monitor system conditions, adjust task priorities, and manage fault tolerance effectively. This complexity can lead to higher computational overhead, which may reduce the overall efficiency of the system in certain scenarios. Additionally, the scalability of adaptive algorithms remains a concern, particularly in largescale cloud environments with millions of tasks and resources. Future research should focus on optimizing these algorithms to reduce overhead and improve scalability, ensuring that they can handle the increasing demands of large cloud systems while maintaining low latency and high resilience.

5. Comparison

The comparison between adaptive and traditional scheduling algorithms reveals several key differences in terms of performance, flexibility, and fault tolerance. Traditional scheduling algorithms, such as FirstComeFirstServe (FCFS) and Shortest Job First (SJF), follow predefined task sequences without considering realtime changes in system load or resource availability. These static algorithms, while straightforward and efficient in stable environments, fail to adapt when the system experiences fluctuations in workload or resource failure. As a result, static schedulers often suffer from inefficiencies, including increased task latency and a higher likelihood of missed deadlines under dynamic conditions. In contrast, adaptive scheduling algorithms, such as Dynamic Priority TaskBased Scheduling (DPTS) and Adaptive Workflow SchedulingCTDC (AWSCTDC), are designed to dynamically adjust task priorities and resource allocations in response to realtime changes in workload and system state. This flexibility allows adaptive algorithms to optimize resource utilization, reduce task latency, and ensure timely task completion, even under fluctuating conditions.

One of the main advantages of adaptive scheduling algorithms is their ability to dynamically adjust to workload changes and system failures. While static scheduling methods are unable to respond to system changes, adaptive algorithms continuously monitor the environment and reallocate resources based on current conditions. For instance, in scenarios where system resources become overburdened or tasks need to be prioritized differently, adaptive algorithms can adjust in realtime to ensure that critical tasks are completed on time. In contrast, static schedulers do not have this capability and often continue executing tasks based on their initial schedules, which can lead to delays and inefficiencies. Additionally, adaptive algorithms are better equipped to handle fault scenarios, such as server crashes or network failures, by reassigning tasks to available resources and minimizing the impact of the failure on task completion. Static scheduling algorithms, however, struggle to recover from such failures, often leading to longer recovery times and missed deadlines.

Fault tolerance is another area where adaptive scheduling algorithms outperform traditional scheduling methods. Adaptive algorithms integrate fault tolerance mechanisms, such as task replication and checkpointing, which allow the system to continue operating seamlessly in the event of a failure. These algorithms can dynamically adjust to failures by replicating tasks across multiple resources or resuming tasks from the last checkpoint, thus

minimizing downtime and ensuring that critical tasks are completed. In comparison, static scheduling algorithms are not equipped to handle failures effectively, as they lack the flexibility to reassign tasks or adjust priorities dynamically. The failure to respond to system issues in real time leads to increased resource wastage, prolonged task delays, and decreased system reliability. Therefore, adaptive scheduling algorithms offer significant fault tolerance benefits, improving the overall resilience of the system and ensuring continuous realtime performance, even under adverse conditions.

In summary, adaptive scheduling algorithms provide distinct advantages over traditional static scheduling methods by dynamically adjusting to changes in workload and system conditions. Their ability to handle failure scenarios, prioritize tasks effectively, and optimize resource utilization results in lower task latency, improved system resilience, and enhanced realtime performance. These benefits make adaptive scheduling algorithms particularly wellsuited for cloud environments, where workloads are dynamic, and maintaining high performance and fault tolerance is essential. While static algorithms may still be suitable for simpler, stable environments, adaptive scheduling is the preferred approach for complex, realtime cloud computing applications.

6. Conclusions

In summary, the findings from the simulation experiments clearly demonstrate the advantages of adaptive scheduling algorithms over traditional static scheduling methods in realtime cloud computing environments. Adaptive scheduling algorithms significantly reduced task latency and improved system resilience, especially under dynamic workloads and fault injection scenarios. The ability of adaptive algorithms to adjust task priorities and resource allocations in real time allowed for more efficient resource utilization, minimized task delays, and ensured that system performance remained optimal, even in the face of failures. These improvements in task latency and system resilience highlight the potential of adaptive scheduling algorithms to enhance the overall performance and reliability of cloud systems.

The implications for realtime parallel processing in cloud computing environments are significant. The reduction in task latency and the enhancement of system resilience make adaptive scheduling algorithms an ideal choice for cloud applications that require rapid data processing, such as artificial intelligence, image processing, and largescale data analysis. As cloud environments continue to grow in scale and complexity, the ability to dynamically adjust scheduling and fault tolerance mechanisms will be critical in maintaining high system performance and reliability. Adaptive scheduling ensures that realtime tasks are completed on time and that the system can recover quickly from failures, making it a vital component of modern cloud computing infrastructures.

Further research in adaptive scheduling algorithms and fault tolerance mechanisms is essential to address existing challenges and optimize their performance in largescale cloud environments. Future studies should focus on enhancing the scalability and efficiency of adaptive algorithms, ensuring that they can handle increasing workloads without introducing excessive overhead. Additionally, comprehensive evaluation frameworks that consider factors such as energy consumption, cost, and system reliability should be developed to better assess the performance of these algorithms. By advancing the field of adaptive scheduling, researchers can help create more robust, efficient, and faulttolerant cloud computing systems capable of meeting the growing demands of realtime applications.

References

[1] M. Sino and E. Domazet, "Scalable Parallel Processing: Architectural Models, RealTime Programming, and Performance Evaluation †," *Eng. Proc.*, vol. 104, no. 1, 2025, doi: 10.3390/engproc2025104060.

[2] S. S. Thomas, S. A. Thomas, J. Paul, and K. K. B. Shibu, "An Intelligent Adaptive Scheduler for Operating Systems Experimented Using FreeRTOS," in *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018*, 2018, pp. 1592 – 1597. doi: 10.1109/ICCONS.2018.8662927.

[3] S. U. Mushtaq, S. Sheikh, and S. M. Idrees, "Enhanced priority based task scheduling with integrated fault tolerance in distributed systems," *Int. J. Cogn. Comput. Eng.*, vol. 6, pp. 152 – 169, 2025, doi: 10.1016/j.ijcce.2024.12.006.

[4] A. P. Sheetal and K. Ravindranath, "Cost Effective Hybrid Fault Tolerant Scheduling Model for Cloud Computing Environment: Hybrid Fault Tolerant Scheduling," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 6, pp. 416 – 422, 2021, doi: 10.14569/IJACSA.2021.0120646.

[5] K. Ajmera, T. K. Tewari, V. K. Singh, Vikash, and P. K. Upadhyay, "EnergyAware Dynamic Virtual Machine Scheduling in Cloud Computing: A Survey," in *ACM International Conference Proceeding Series*, 2023, pp. 133 – 141. doi: 10.1145/3607947.3607970.

[6] T. Hagras and G. A. ElSayed, "A faulttolerant and loadbalancing scheduler for independent tasks on cloudbased virtual machines," *Cluster Comput.*, vol. 29, no. 1, 2026, doi: 10.1007/s10586025058571.

[7] K. Tanaka, "Realtime adaptive task scheduling," in *Proceedings of the 2005 International Conference on Embedded Systems and Applications, ESA'05*, 2005, pp. 24 – 30. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2s2.060749110125&partnerID=40&md5=11898a7f75ac7a9cfb086cdd3d8eea95

[8] N. Wu, D. Zuo, and Z. Zhang, "Dynamic faulttolerant workflow scheduling with hybrid spatialtemporal reexecution in clouds," *Inf.*, vol. 10, no. 5, 2019, doi: 10.3390/info10050169.

[9] S. Sahoo, S. Nawaz, S. K. Mishra, and B. Sahoo, "Execution of real time task on cloud environment," in *12th IEEE International Conference Electronics, Energy, Environment, Communication, Computer, Control: (E3C3), INDICON 2015*, 2016. doi: 10.1109/INDICON.2015.7443778.

[10] S.J. Yang and W.L. Lu, "Design a Distributed Fog Computing Scheme to Enhance Processing Performance in RealTime IoT Applications," *Lect. Notes Data Eng. Commun. Technol.*, vol. 41, pp. 99 – 112, 2020, doi: 10.1007/9783030349868_7.

[11] M. R. Kale, S. Labhane, S. E. Manu, P. Mehta, A. Amudha, and A. Gupta, "Scalable and Efficient RealTime Data Processing in CloudBased Manufacturing Systems," in *2024 15th International Conference on Computing Communication and Networking Technologies, ICCCNT 2024*, 2024. doi: 10.1109/ICCCNT61001.2024.10725934.

[12] P. Akilandeswari and H. Srimathi, "Survey and analysis on task scheduling in cloud environment," *Indian J. Sci. Technol.*, vol. 9, no. 37, 2016, doi: 10.17485/ijst/2016/v9i37/102058.

[13] H. Wang and H. Wang, "Survey On Task Scheduling in Cloud Computing Environment," in *ICIIBMS 2022 7th International Conference on Intelligent Informatics and Biomedical Sciences*, 2022, pp. 286 – 291. doi: 10.1109/ICIIBMS55689.2022.9971622.

[14] R. S. K. Aakisetti, V. Ganta, P. Yellamma, C. Siram, S. H. Gampa, and K. V Brahma Rao, "Dynamic Priority Scheduling Algorithms for Flexible Task Management in Cloud Computing," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 13s, pp. 246 – 256, 2024, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2s2.085184431242&partnerID=40&md5=609ac627c3fc050f57af59c347d44ace

[15] R. Ghafouri and A. Movaghar, "An adaptive and deadlineconstrained workflow scheduling algorithm in infrastructure as a service clouds," *Iran J. Comput. Sci.*, vol. 5, no. 1, pp. 17 – 39, 2022, doi: 10.1007/s42044021000826.

[16] Y. Liu, J. Liu, Z. Zhu, C. Deng, Z. Ren, and X. Xu, "Adaptive faulttolerant scheduling in heterogeneous realtime systems," in *Proceedings of the 14th IEEE Conference on Industrial Electronics and Applications, ICIEA 2019*, 2019, pp. 982 – 987. doi: 10.1109/ICIEA.2019.8833833.

[17] K. Sumangali and N. Benny, "Advanced cloud fault tolerance system," in *IOP Conference Series: Materials Science and Engineering*,

2017. doi: 10.1088/1757899X/263/4/042060.

[18] P. Marcotte, F. Gregoire, and F. Petrillo, "Multiple faultTolerance mechanisms in cloud systems: A systematic review," in *Proceedings 2019 IEEE 30th International Symposium on Software Reliability Engineering Workshops, ISSREW 2019*, 2019, pp. 414 – 421. doi: 10.1109/ISSREW.2019.00104.

[19] J. Hao, Z. Cui, and Z. Peng, "Load balancing for data centre: A brief survey," *Int. J. Wirel. Mob. Comput.*, vol. 11, no. 1, pp. 47 – 53, 2016, doi: 10.1504/IJWMC.2016.079464.

[20] K. A. Ali, O. A. Fadare, and F. AlTurjman, "Dynamic Resource Allocation (DRA) in Cloud Computing," *Sustain. Civ. Infrastructures*, vol. Part F4042, pp. 1033 – 1049, 2025, doi: 10.1007/9783031725098_85.

[21] J. Liu, Z. Zhu, and C. Deng, "A Novel and Adaptive Transient FaultTolerant Algorithm Considering Timing Constraint on Heterogeneous Systems," *IEEE Access*, vol. 8, pp. 103047 – 103061, 2020, doi: 10.1109/ACCESS.2020.2999092.

[22] O. Gokalp, "Performance evaluation of heuristic and metaheuristic algorithms for independent and static task scheduling in cloud computing; [Bulut hesaplamada bağimsiz ve statik görev çizelgeleme için sezgisel ve metasezgisel algoritmalarin performans değerlendirmesi]," in *SIU 2021 29th IEEE Conference on Signal Processing and Communications Applications, Proceedings*, 2021. doi: 10.1109/SIU53274.2021.9477821.

[23] A. Khiat, "Optimizing Cloud Energy Consumption Using Static Task Scheduling Algorithms: A Comparative Study," in *2023 14th International Conference on Information and Communication Systems, ICICS 2023*, 2023. doi: 10.1109/ICICS60529.2023.10330466.