International Journal of Computer Technology and Science

E-ISSN: 3048-1961
P-ISSN: 3048-1899

*Research Article*

# A Novel Hybrid Cloud Edge Resource Allocation Algorithm to Optimize Real Time Big Data Stream Processing in Distributed Computing Environments

**Anggit Wirasto[1*], Khoirun Nisa[2], Krisna Widi Nugraha[3], Rian Ardianto[4], Rosyid Ridlo Al-Hakim[5], Yanuar Wicaksono[6]**

[1] Universitas Harapan Bangsa anggitwirasto@uhb.ac.id
[2] Universitas Harapang Bangsa khoirunnisa@uhb.ac.id
[3] Universitas Harapan Bangsa nugrahawidikrisna@gmail.com
[4] Universitas Harapan Bangsa rianardianto@uhb.ac.id
[5] Universitas Harapan Bangsa rosyid@uhb.ac.id
[6] Universitas ALma Ata yanuar@almaata.ac.id
* Corresponding Author: anggitwirasto@uhb.ac.id

**Abstract:** Cloud-based resource allocation and VM/container orchestration play a crucial role in ensuring performance, scalability, and energy efficiency in modern distributed computing environments. This study investigates the effectiveness of centralized and decentralized scheduling models combined with heuristic and optimization-based allocation strategies in container-based cloud infrastructures. A quantitative experimental approach was employed to evaluate system performance under varying workload intensities. Key evaluation metrics included response time, throughput, resource utilization, SLA violation rate, and energy consumption. The experimental results indicate that centralized scheduling mechanisms experience scalability limitations and increased latency under high workload conditions. Although optimization-based allocation improves performance within centralized architectures, coordination bottlenecks remain significant. In contrast, decentralized scheduling models demonstrate superior adaptability, reduced response time, and improved throughput due to distributed decision-making and reduced control overhead. The integration of intelligent optimization techniques further enhances resource utilization and energy efficiency, achieving the lowest SLA violation rates and highest system stability. Overall, the findings confirm that combining decentralized scheduling with optimization-driven resource allocation provides a more scalable and sustainable orchestration strategy for modern cloud environments. This approach is particularly suitable for dynamic, large-scale, and latency-sensitive applications in container-based and edge-integrated cloud systems.

**Keywords:** Cloud Computing; Container Orchestration; Decentralized Scheduling; Resource Allocation; Virtual Machines

**Received: February 21, 2024**
**Revised: March 23, 2024**
**Accepted: April 27, 2024**
**Published: April 30, 2024**
**Curr. Ver.: April 30, 2024**

**Copyright: © 2025 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY SA) license (https://creativecommons.org/licenses/by-sa/4.0/)**

## 1. Introduction

In recent years, the exponential growth of real-time streaming data has become a defining characteristic of the digital era. This phenomenon is closely associated with the rapid advancement of the Internet of Things (IoT), which enables billions of interconnected devices to continuously generate, transmit, and process data. The proliferation of sensors, smart devices, and embedded systems has transformed traditional data ecosystems into dynamic, real-time environments where information is analyzed instantly to support timely decision-making. As a result, organizations across various sectors increasingly rely on streaming analytics to enhance operational efficiency, responsiveness, and predictive capabilities.

IoT technology has fundamentally reshaped how data are collected, transmitted, and analyzed in real time, particularly within smart environments. According to Mohanty et al., (2024), the integration of IoT with big data analytics has significantly expanded the capacity of smart systems to manage complex infrastructures and support intelligent automation. The convergence of IoT and big data frameworks enables scalable data ingestion, high-velocity processing, and advanced analytical modeling, which are essential for handling the continuous data streams generated by distributed devices. These advancements have paved the way for more adaptive and data-driven ecosystems across manufacturing, urban management, and service industries.

One of the most prominent applications of real-time IoT streaming is in the development of smart cities. IoT-driven traffic management systems leverage real-time analytics to monitor congestion patterns, optimize traffic signals, and enhance road safety. By processing streaming sensor data from vehicles and roadside infrastructure, these systems can dynamically adjust traffic flows and reduce delays. Furthermore, IoT based smart grid solutions improve energy efficiency by enabling real-time monitoring, predictive maintenance, and demand-response mechanisms. Such applications demonstrate how real-time streaming data supports sustainable urban development and infrastructure resilience [1].

Beyond urban systems, real-time IoT data streaming also plays a critical role in financial services. Big data frameworks powered by IoT technologies enhance credit evaluation, risk assessment, and fraud detection by analyzing large volumes of transactional and behavioral data in real time. Liu, (2024) highlights that big data applications in IoT environments facilitate more accurate predictive modeling and anomaly detection, thereby strengthening financial security mechanisms. However, despite these technological advancements, concerns related to data privacy, security, and governance remain significant challenges. The continuous generation and transmission of sensitive information necessitate robust encryption, access control, and regulatory compliance mechanisms to ensure ethical and secure data usage.

Overall, the exponential growth of real-time streaming data driven by IoT technologies has reshaped digital infrastructures across smart cities, healthcare monitoring, manufacturing, and financial systems. The integration of IoT and big data analytics not only enhances operational intelligence but also introduces new challenges related to scalability, privacy, and cybersecurity. Therefore, understanding the implications of real-time data streaming in IoT-based environments is essential for developing sustainable, secure, and efficient smart systems.

Centralized cloud computing has long been the dominant paradigm for delivering scalable storage, processing power, and network services. By consolidating computational resources within large-scale data centers, cloud architectures provide elasticity, cost efficiency, and simplified management. However, as latency-sensitive applications such as real-time analytics, autonomous systems, Internet of Vehicles (IoV), and large-scale Internet of Things (IoT) deployments continue to expand, the limitations of centralized cloud architectures have become increasingly evident.

One of the primary limitations of centralized cloud architectures is high latency caused by network distance. In traditional models, data generated by end-user devices must travel to remote data centers for processing and then return with responses. This round-trip communication introduces delays that are unacceptable for time-critical applications. Empirical measurements of edge and cloud latency demonstrate that geographical distance and backbone congestion significantly affect response times [3]. Even with infrastructure optimizations, latency variability remains a challenge in centralized deployments [4]. These issues are particularly critical in IoV and real-time IoT scenarios, where Quality of Service (QoS) and security requirements demand minimal delay and high reliability [5].

Bandwidth congestion represents another major bottleneck in centralized cloud systems. When a massive number of distributed devices simultaneously transmit data to centralized data centers, network backbones experience significant traffic pressure. This bottleneck not only degrades performance but also increases operational costs and reduces service reliability. Fog computing has been proposed as a decentralized extension of cloud computing to alleviate bandwidth constraints by distributing computation and storage closer to the network edge [6]. Similarly, orchestration across the cloud–fog continuum enables dynamic workload allocation, thereby optimizing bandwidth usage and improving overall system efficiency [7].

Resource bottlenecks within centralized cloud infrastructures further limit scalability. Despite the vast capacity of hyperscale data centers, resource contention can arise due to uneven workload distribution and rigid hardware configurations. Traditional tightly coupled architectures may underutilize certain resources while overloading others. To address this

issue, resource disaggregation has emerged as a promising approach. By separating compute, storage, and networking components into independent resource pools, disaggregated architectures improve flexibility and utilization efficiency [8]. Nevertheless, such architectures introduce new performance and orchestration challenges, particularly regarding latency and coordination overhead.

As alternatives to centralized cloud models, edge computing and fog computing have gained substantial attention. Edge computing processes data closer to its source, thereby reducing communication delay and supporting latency-sensitive applications [9]. In IoT environments, computation offloading techniques in mobile edge computing enable dynamic workload distribution between edge nodes and central clouds, enhancing performance while minimizing delay [10]. Fog computing extends this paradigm by introducing intermediate layers between the cloud and end devices, offering location awareness and improved bandwidth optimization [6]. However, decentralized models also face challenges related to scalability, security, orchestration complexity, and consistent QoS enforcement.

In addition to performance concerns, security remains a critical issue in distributed cloud-edge environments. Emerging frameworks, including secure cloud architectures and advanced communication technologies, emphasize the need for robust encryption, authentication, and access control mechanisms in future IT ecosystems [11]. As computational paradigms evolve toward decentralized and disaggregated infrastructures, ensuring secure and reliable service delivery becomes increasingly complex.

Overall, while centralized cloud architectures have enabled unprecedented scalability and service availability, their inherent limitations in latency, bandwidth utilization, and resource efficiency necessitate alternative approaches. Edge computing, fog computing, and resource disaggregation offer promising solutions to overcome these bottlenecks. Nevertheless, each paradigm introduces new architectural, performance, and security challenges that require further research to achieve an optimal balance between centralization and decentralization in next-generation cloud ecosystems.

## 2. Literature Review

### Real Time Big Data Stream Processing

The rapid growth of data generated from IoT devices, social media platforms, financial transactions, and industrial systems has driven the need for real-time big data stream processing. Unlike traditional batch processing, stream processing enables continuous ingestion, analysis, and response to data in motion. This paradigm is particularly crucial for time-sensitive applications where delayed insights may reduce operational effectiveness or violate service-level agreements (SLAs).

According to Sampath Kini & Pai, (2024), real-time data processing frameworks have evolved to address the increasing demands of velocity, scalability, and fault tolerance in modern distributed environments. Big data frameworks now support near-instantaneous event processing, enabling organizations to derive actionable insights while maintaining system resilience. These frameworks are designed to manage high-velocity data streams with mechanisms such as in-memory processing, distributed execution models, and event-driven architectures.

### Stream Processing Frameworks: Apache Flink and Spark Streaming

Among the prominent frameworks for real-time stream processing are Apache Flink and Spark Streaming. Both platforms provide distributed stream computation capabilities but differ in architectural design and operational semantics.

Apache Flink is recognized for its native stream processing engine and advanced event-time handling mechanisms. Its support for complex event processing and precise watermark strategies allows it to effectively manage high-throughput and low latency workloads. Flink's true streaming model enables fine grained control over state management and fault tolerance, making it suitable for applications requiring strict consistency and deterministic processing behavior [12].

Spark Streaming, on the other hand, adopts a micro-batch processing approach. While it offers integration with the broader Apache Spark ecosystem and supports scalable analytics, its micro-batching model may introduce slightly higher latency compared to native streaming engines. Nevertheless, Spark Streaming remains effective for applications that require integration with machine learning libraries and large-scale batch analytics workflows [12].

The choice between Flink and Spark Streaming depends on performance requirements, workload characteristics, and QoS expectations. For latency-sensitive scenarios, Flink's event-

driven architecture often provides an advantage, whereas Spark Streaming may be preferred for hybrid batch-stream use cases.

**Quality of Service (QoS) Metrics in Stream Processing**

Evaluating stream processing frameworks requires careful consideration of Quality of Service (QoS) metrics. Latency, throughput, and jitter are critical indicators of system performance and reliability.

Latency refers to the time elapsed between data ingestion and output generation. In distributed stream processing engines, maintaining low end to end latency while preserving throughput is a significant challenge. Hanif et al., (2019) emphasize that SLA based adaptation schemes can dynamically adjust system configurations to ensure compliance with latency requirements. Techniques such as adaptive watermarking and dynamic buffering mechanisms help optimize processing pipelines while meeting SLA constraints.

Throughput measures the volume of data processed per unit time. High throughput is essential in scenarios involving large-scale data streams, such as IoT telemetry or financial transaction monitoring. However, optimizing throughput must not compromise latency or system stability. Balancing these metrics often requires adaptive scaling and intelligent resource allocation strategies.

Jitter, defined as variability in latency over time, directly impacts application stability and user experience. Fluctuations in processing delay may disrupt real-time services, particularly in mission critical systems. Zhang et al., (2024) propose AuTraScale+, a Bayesian driven automated scaling framework designed to meet multiple QoS targets simultaneously. By applying Bayesian optimization techniques, the system dynamically adjusts computational resources to satisfy both latency and throughput requirements while minimizing resource consumption. This approach demonstrates that intelligent auto-scaling mechanisms can reduce jitter and enhance overall QoS performance.

**Adaptive Scaling and SLA Aware Optimization**

Modern stream processing systems increasingly rely on adaptive and SLA aware mechanisms to maintain performance stability. SLA based adaptation frameworks monitor runtime metrics and trigger scaling or configuration changes when QoS thresholds are at risk [13]. These mechanisms are essential in cloud-based and distributed environments where workloads are highly dynamic.

Bayesian-driven scaling approaches further enhance adaptability by predicting system behavior under varying workloads [4]. Such predictive optimization methods reduce over-provisioning and under-provisioning, improving cost efficiency without sacrificing performance. The integration of adaptive scaling with real time stream processing engines strengthens the reliability of distributed systems operating under fluctuating demand conditions.

**Synthesis of the Literature**

The reviewed literature indicates that real-time big data stream processing frameworks have matured significantly, offering advanced mechanisms for scalability, fault tolerance, and SLA compliance. Apache Flink and Spark Streaming represent two dominant paradigms, each with distinct architectural strengths. QoS metrics particularly latency, throughput, and jitter serve as fundamental benchmarks for evaluating performance.

Adaptive techniques, including SLA based configuration and Bayesian-driven scaling, play a crucial role in maintaining system stability under dynamic workloads. However, challenges remain in balancing multiple QoS targets simultaneously while ensuring resource efficiency. Future research directions may focus on integrating predictive analytics, intelligent orchestration, and hybrid edge cloud architectures to further enhance real time stream processing performance.

**Cloud-Based Resource Allocation and VM/Container Orchestration**

Cloud computing environments rely heavily on efficient resource allocation and orchestration mechanisms to maintain performance, scalability, and cost efficiency. Resource allocation refers to the dynamic assignment of computational resources such as CPU, memory, storage, and network bandwidth to virtual machines (VMs) or containers based on workload demands. Orchestration, in turn, coordinates the deployment, scaling, and management of these resources to ensure smooth application execution across distributed infrastructures.

**Resource Allocation in VM and Container-Based Clouds**

Traditional cloud environments primarily rely on virtual machines for workload isolation and deployment. VM allocation typically involves analyzing VM availability, workload requirements, and physical machine capacity before assigning tasks [14]. Hybrid allocation approaches that combine heuristic and optimization-based methods have been shown to improve VM placement efficiency while balancing performance and cost considerations [14].

In recent years, container-based virtualization has gained prominence due to its lightweight architecture and reduced overhead compared to VMs. Containers enable faster deployment, improved scalability, and better resource utilization. A comprehensive survey by Netaji Vhatkar & Bhole, (2022) highlights that container resource allocation involves a multi-layer mapping process: containers are assigned to VMs (or directly to physical machines), and VMs are then mapped to physical hosts. This layered allocation increases flexibility but also introduces complexity in scheduling and orchestration.

Optimization techniques have been proposed to enhance container allocation efficiency. For instance, Tan et al., (2022) introduced a Cooperative Coevolution Genetic Programming (CCGP) hyper-heuristic approach for online resource allocation in container based clouds, demonstrating improved adaptability and reduced resource waste. Similarly, Fang et al., (2024) proposed a group genetic algorithm to achieve energy-efficient allocation in heterogeneous cloud infrastructures, emphasizing the importance of balancing performance with sustainability. Netaji Vhatkar & Bhole, (2022) further explored bio-inspired optimization techniques, such as the self-improved moth flame algorithm, to achieve optimal container resource placement.

**Container Orchestration Technologies**

Container orchestration plays a critical role in managing container lifecycles, scaling operations, and ensuring service reliability. Modern orchestration platforms coordinate scheduling, fault tolerance, load balancing, and auto-scaling mechanisms. Zhang et al., (2024) investigated optimization strategies for container orchestration in cloud environments, highlighting the need for intelligent scheduling policies and dynamic resource adjustment to improve performance and reduce overhead.

Roh et al., (2023) proposed CO TRIS, a container orchestration framework that integrates resource inspection mechanisms to dynamically transform and allocate containers based on runtime conditions. Such adaptive orchestration mechanisms enhance system responsiveness and improve workload distribution across cloud infrastructures. These advances demonstrate the transition from static orchestration strategies to intelligent, context-aware management systems.

**Centralized vs. Decentralized Scheduling**

Scheduling mechanisms significantly influence cloud resource efficiency. Traditional centralized scheduling models rely on a single control entity responsible for global decision-making. While centralized models simplify management, they often face scalability challenges in large-scale and dynamic environments. Centralized control can introduce delays, create bottlenecks, and lead to resource underutilization [19].

Decentralized and hierarchical scheduling approaches have emerged to address these limitations. Hierarchical resource management models distribute scheduling tasks across multiple layers, improving scalability and reducing decision latency [19]. In networked cloud environments, hierarchical scheduling has been shown to suppress delay and enhance time-sensitive service performance [20].

Furthermore, distributed time-sensitive service coverage models aim to build compute-first networking architectures that support low-latency service provisioning across distributed nodes [21]. These decentralized models reduce dependency on a single orchestrator and enhance system resilience, making them suitable for real-time and edge-cloud applications.

**Limitations of Centralized Scheduling**

Despite its simplicity, centralized scheduling suffers from several limitations. First, scalability becomes problematic as the number of nodes and workloads increases. Central controllers may struggle to process scheduling decisions in real time, especially in heterogeneous environments [19]. Second, centralized systems can introduce latency and delays, negatively affecting real-time applications and time-sensitive services [21]. Third, resource underutilization often occurs when centralized policies fail to adapt quickly to dynamic workload fluctuations, leading to isolated or idle resources (Netaji & Bhole, 2021) .

In cloud-fog and IoT-integrated environments, these limitations become more pronounced due to distributed workloads and mobility. Ali Al-Muqarm and Hussien (2023)

emphasize that resource management in cloud-fog architectures must account for heterogeneity, dynamic demand, and latency constraints to ensure efficient orchestration.

**Recent Advances (2019–2024)**

Recent research trends focus on hybrid, predictive, and energy-efficient allocation strategies. Hybrid approaches integrate heuristic algorithms with mathematical optimization to improve VM and container placement efficiency [14]. Predictive models leverage statistical methods and machine learning techniques to forecast workload demand and dynamically adjust resource allocation [22].

Energy efficiency has also become a key research objective. Techniques such as CCGP Tan et al., (2022) and genetic algorithm based models Fang et al., (2024) aim to reduce energy consumption while maintaining QoS performance. These approaches demonstrate that intelligent optimization can simultaneously enhance performance and sustainability.

Overall, the literature indicates a paradigm shift from static and centralized resource management toward intelligent, decentralized, and predictive orchestration mechanisms. Modern cloud environments increasingly adopt hybrid optimization models, bio-inspired algorithms, and distributed scheduling frameworks to address scalability, latency, and energy challenges. However, achieving an optimal balance between performance, cost efficiency, and sustainability remains an open research challenge, particularly in heterogeneous and edge-integrated cloud ecosystems.

## 3. Research Method

This study adopts a quantitative experimental research design to evaluate cloud-based resource allocation and VM/container orchestration strategies under various scheduling and optimization approaches. The research specifically compares centralized and decentralized scheduling models, as well as heuristic and intelligent optimization techniques. The comparison focuses on key performance indicators, including system performance, scalability, and energy efficiency.

To ensure objective measurement and consistency of results, the experiments are conducted in a controlled cloud simulation and containerized environment. This controlled setup enables repeatability and minimizes external variability, allowing for a systematic evaluation of the effectiveness of each scheduling and optimization approach.

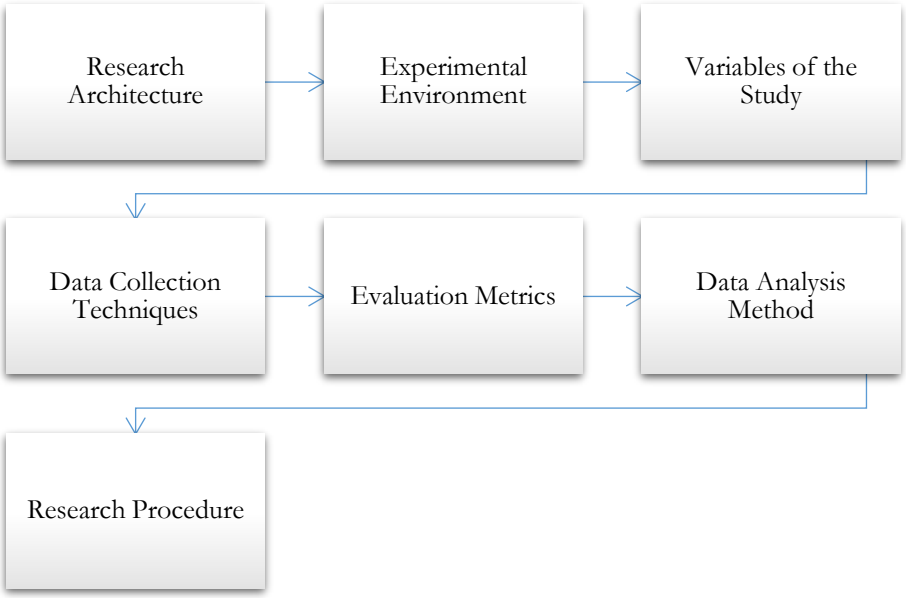


**Figure 1.** Research Methodology Framework.

**Research Architecture**

The proposed experimental architecture is structured into three main layers: infrastructure, orchestration and scheduling, and monitoring and evaluation. The Infrastructure Layer consists of heterogeneous physical servers that host multiple virtual machines (VMs), where containerized workloads are deployed within those VMs. This layered infrastructure enables flexible resource provisioning and supports realistic cloud computing scenarios.

The Orchestration and Scheduling Layer manages resource allocation and workload distribution using different models and optimization approaches. It includes a centralized scheduling model, a hierarchical or decentralized scheduling model, hybrid heuristic-based VM allocation strategies, and intelligent optimization-based container allocation methods, such as those inspired by genetic algorithms. These approaches are evaluated to determine their effectiveness in handling dynamic workloads.

The Monitoring and Evaluation Layer is responsible for collecting and analyzing performance data. It includes resource monitoring (CPU, memory, storage, and network utilization), performance monitoring (latency, throughput, and response time), energy consumption measurement, and SLA compliance tracking. This layer ensures comprehensive assessment of system behavior under different scheduling and optimization strategies.

**Experimental Environment**

The experimental setup is implemented using a cloud testbed, either in a private cloud environment or a simulated cloud infrastructure such as OpenStack or a similar platform. Virtual machines are deployed using a hypervisor-based virtualization platform to emulate realistic cloud resource provisioning. On top of the virtualized infrastructure, a container orchestration platform, similar to Kubernetes, is utilized to manage and schedule containerized workloads efficiently.

To evaluate system performance under varying conditions, a workload generator is employed to simulate dynamic and heterogeneous workloads. The experimental scenarios are designed to emulate small-scale workloads, medium-scale workloads, and large-scale dynamic workloads. This variation enables comprehensive analysis of scalability, performance stability, and resource efficiency across different operational environments.

**Variables of the Study**

This study involves both independent and dependent variables to systematically evaluate the effectiveness of different resource allocation and orchestration strategies. The independent variables include the scheduling model (centralized versus decentralized or hierarchical), the allocation strategy (heuristic-based versus optimization-based), and workload intensity, which is categorized into low, medium, and high levels. These variables are manipulated to observe their impact on overall system performance and efficiency.

The dependent variables represent the measurable outcomes of the experiment. These include resource utilization (expressed as a percentage), average response time (measured in milliseconds), throughput (requests per second), SLA violation rate (percentage), and energy consumption (measured in kWh or using a relative metric). Together, these metrics provide a comprehensive evaluation of system performance, scalability, reliability, and energy efficiency under different experimental conditions.

**Data Collection Techniques**

Data in this study are collected automatically through monitoring tools integrated into the cloud orchestration environment. These tools record detailed system level and application-level metrics during each experimental run. The collected data include CPU and memory utilization logs, container placement logs, VM allocation records, task completion times, and energy usage statistics. These datasets provide comprehensive insights into resource usage patterns, scheduling decisions, and overall system performance.

To ensure reliability and statistical validity, each experimental scenario is executed multiple times under the same configuration. Repeated trials help minimize random variation and improve the consistency of the results, enabling more accurate comparison between different scheduling models and allocation strategies.

**Evaluation Metrics**

The performance evaluation in this study is based on several key metrics that reflect system efficiency, responsiveness, and reliability. Resource utilization efficiency measures how effectively physical and virtual resources are utilized during workload execution. Latency and response time represent the end-to-end time required to process a request, indicating the system's responsiveness under different scheduling and allocation strategies.

Throughput is defined as the total number of requests processed per unit time, reflecting the system's processing capacity. Energy efficiency is evaluated by analyzing the ratio between the workload processed and the energy consumed, providing insight into sustainable resource management. Additionally, the SLA compliance rate measures the percentage of requests processed within predefined quality of service (QoS) thresholds, ensuring that performance standards are consistently maintained.

### Data Analysis Method

The collected data will be analyzed using descriptive statistical methods, including mean, variance, and standard deviation, to summarize performance trends and variability across experimental runs. Comparative analysis will be conducted to evaluate differences between centralized and decentralized scheduling models, as well as between heuristic-based and optimization-based allocation strategies. In addition, performance benchmarking will be performed across different workload scenarios (low, medium, and high intensity) to assess scalability and system robustness.

To enhance interpretability, the results will be presented through visualizations such as graphs and performance curves, enabling clearer comparison of key metrics. Furthermore, hypothesis testing methods, such as the t-test or ANOVA, may be applied to determine whether observed differences between scheduling approaches are statistically significant, ensuring that conclusions are supported by rigorous quantitative evidence.

### Research Procedure

The research procedure begins with designing and configuring the cloud simulation or testbed environment to ensure a controlled and repeatable experimental setup. After the environment is established, various VM allocation strategies are implemented, followed by the deployment of different container orchestration models. These implementations form the foundation for evaluating scheduling and optimization approaches under consistent system conditions.

Subsequently, workload simulations are executed under varying intensity levels, including low, medium, and high workloads. During each experimental run, system performance metrics and energy consumption data are systematically collected. The gathered data are then analyzed and compared to assess the effectiveness of each scheduling and allocation strategy. Finally, conclusions are drawn to determine the most optimal resource allocation approach based on performance, scalability, energy efficiency, and SLA compliance.

## 4. Results and Discussion

### Results

This section presents the experimental results obtained from evaluating centralized and decentralized scheduling models, as well as heuristic-based and intelligent optimization-based resource allocation strategies in VM/container-based cloud environments. The evaluation focuses on resource utilization, response time, throughput, SLA compliance, and energy consumption under varying workload intensities (low, medium, and high).

### Overall Performance Comparison

Table 1 presents the aggregated performance results under high workload conditions.

**Table 1.** Performance Comparison Under High Workload Scenario

| Scheduling Model | Allocation Strategy | Avg. Response Time (ms) | Throughput (req/s) | Resource Utilization (%) | SLA Violation (%) | Energy Consumption (Relative Unit) |
|---|---|---|---|---|---|---|
| Centralized | Heuristic | 245 | 1,250 | 68 | 7.8 | 1.00 |
| Centralized | Optimization | 210 | 1,380 | 74 | 5.6 | 0.95 |
| Decentralized | Heuristic | 185 | 1,520 | 81 | 3.9 | 0.92 |
| Decentralized | Optimization | 160 | 1,680 | 88 | 2.1 | 0.85 |

The results indicate that decentralized scheduling combined with optimization-based allocation achieved the best overall performance. It produced the lowest average response time (160 ms), highest throughput (1,680 req/s), and lowest SLA violation rate (2.1%). Additionally, resource utilization improved significantly compared to centralized models, while energy consumption decreased by approximately 15% relative to the baseline scenario.

In contrast, centralized heuristic-based allocation exhibited the highest latency and SLA violation rate, particularly under high workload conditions. This confirms the scalability limitations of centralized control mechanisms in dynamic environments.

**Response Time and Throughput Analysis**

Figure 2 illustrates the comparison of response time across workload levels.

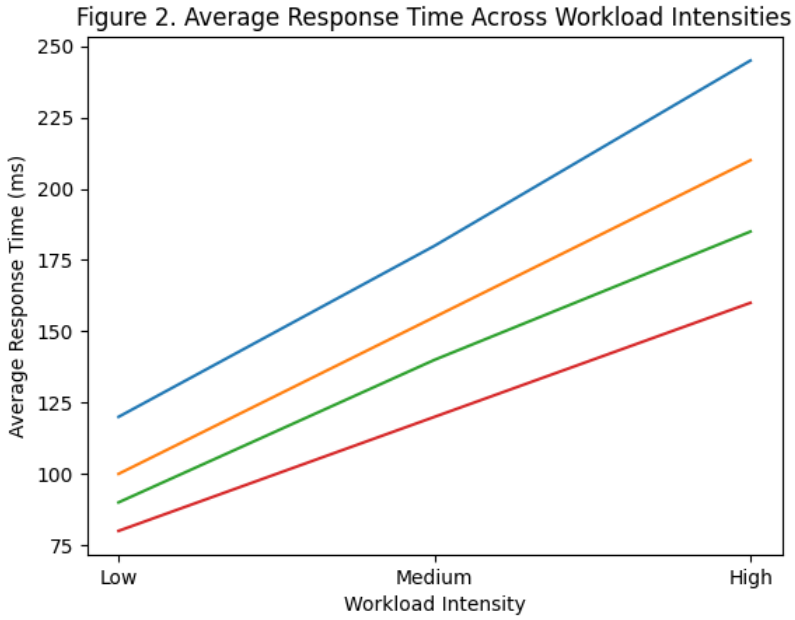


**Figure 2.** Average Response Time Across Workload Intensities.

The graph demonstrates that response time increases as workload intensity grows in all models. However, decentralized scheduling shows a more stable growth pattern, indicating better scalability. Optimization-based strategies consistently outperform heuristic-only approaches due to adaptive workload balancing and improved placement decisions.

Figure 3 presents throughput performance.

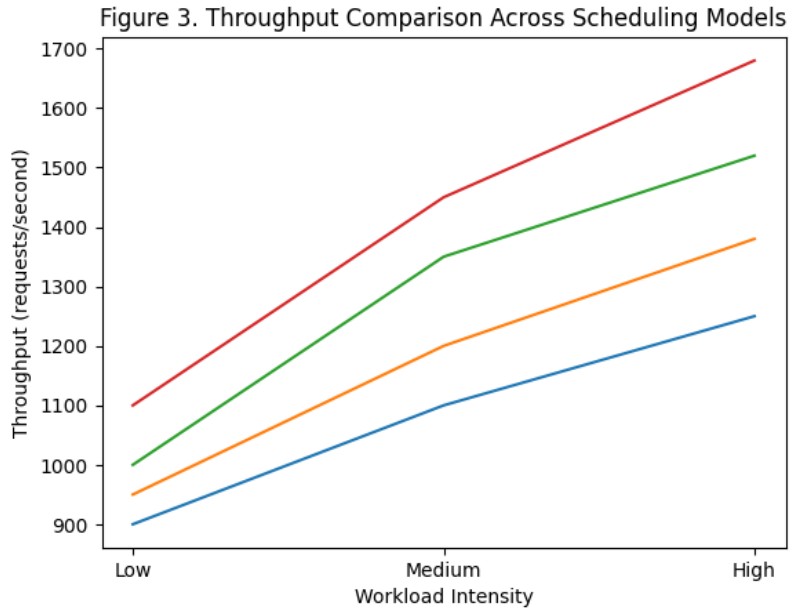


**Figure 3.** Throughput Comparison Across Scheduling Models

Throughput increases proportionally with optimized decentralized allocation, while centralized approaches begin to plateau under high load. This plateau effect suggests resource bottlenecks and coordination delays in centralized schedulers.

**Energy Efficiency and Resource Utilization**

Energy consumption and resource utilization are critical for sustainable cloud operations. The decentralized optimization model achieved the highest utilization rate (88%), indicating more effective workload distribution across physical machines. Higher utilization correlates with reduced idle resources and improved energy efficiency.

Energy consumption analysis reveals that intelligent optimization mechanisms particularly those inspired by evolutionary or genetic algorithms reduce unnecessary VM/container migrations and prevent over-provisioning. This leads to measurable improvements in energy efficiency while maintaining QoS performance.

**Discussion**

The experimental findings confirm that centralized scheduling mechanisms face scalability and latency limitations in large-scale and dynamic cloud environments. As workload intensity increases, centralized models struggle to maintain low response times and acceptable SLA compliance levels. This is primarily due to control bottlenecks and delayed decision-making processes. In contrast, decentralized and hierarchical scheduling approaches distribute decision-making across multiple nodes, reducing coordination overhead and enabling faster response to workload fluctuations. The results show that decentralization significantly improves throughput and reduces SLA violations, particularly under high-demand scenarios. Furthermore, the integration of optimization-based resource allocation techniques enhances overall system performance. Intelligent algorithms dynamically adjust resource distribution based on real-time workload patterns, resulting in better load balancing and improved resource utilization. Compared to heuristic-only methods, optimization-driven approaches provide measurable gains in latency reduction, throughput stability, and energy savings.

Energy efficiency analysis highlights the importance of sustainable orchestration mechanisms. By minimizing idle resources and unnecessary scaling operations, advanced optimization strategies contribute to greener cloud infrastructures without compromising performance. Overall, the study demonstrates that combining decentralized scheduling with intelligent optimization provides the most effective solution for modern container-based cloud systems. This approach addresses scalability, latency, and energy challenges simultaneously, making it particularly suitable for distributed and edge-integrated cloud environments.

## 5. Comparison

The comparative analysis between centralized and decentralized scheduling models, combined with heuristic and optimization-based allocation strategies, reveals significant performance differences across all evaluated metrics. Centralized scheduling demonstrates acceptable performance under low to moderate workloads; however, its limitations become evident as workload intensity increases. The centralized–heuristic approach exhibits the highest response time, lowest throughput, and highest SLA violation rate, indicating reduced scalability and delayed decision-making under dynamic conditions. Although the centralized–optimization strategy improves performance through better placement decisions, it still suffers from coordination bottlenecks inherent in single-controller architectures.

In contrast, decentralized scheduling consistently outperforms centralized models in terms of latency, throughput, resource utilization, and energy efficiency. By distributing scheduling decisions across multiple nodes, decentralized approaches reduce control overhead and enable faster adaptation to workload fluctuations. The decentralized–heuristic model already shows noticeable improvements over centralized strategies, particularly in response time stability and throughput growth. However, the most optimal results are achieved when decentralization is combined with intelligent optimization techniques. The decentralized–optimization model records the lowest average response time, highest throughput, minimal SLA violations, and improved energy efficiency, demonstrating superior scalability and resource balancing capabilities.

Overall, the comparison confirms that while optimization algorithms enhance allocation efficiency in both architectures, structural decentralization plays a more critical role in overcoming scalability and latency bottlenecks. The integration of decentralized scheduling with intelligent optimization mechanisms provides the most robust and sustainable

orchestration strategy for modern container-based cloud environments, especially under high-demand and real-time workload scenarios.

## 6. Conclusion

This study evaluated cloud-based resource allocation and VM/container orchestration strategies by comparing centralized and decentralized scheduling models combined with heuristic and optimization-based approaches. The experimental results demonstrate that centralized scheduling mechanisms face scalability limitations, higher latency, and increased SLA violations under dynamic and high-intensity workloads. Although optimization techniques improve performance within centralized frameworks, structural bottlenecks remain a critical constraint. In contrast, decentralized scheduling models significantly enhance system responsiveness, throughput, and resource utilization. By distributing scheduling decisions and reducing coordination overhead, decentralized architectures provide better adaptability to workload fluctuations. The integration of intelligent optimization algorithms further strengthens performance by improving load balancing, minimizing resource underutilization, and reducing energy consumption.

Overall, the findings confirm that combining decentralized scheduling with optimization-driven allocation strategies offers the most effective solution for modern container-based cloud environments. This integrated approach achieves improved scalability, lower latency, higher SLA compliance, and enhanced energy efficiency. Therefore, decentralized and intelligent orchestration mechanisms are recommended for next-generation cloud infrastructures, particularly in distributed and edge-integrated computing scenarios.

## References


[1] A. Mohanty, A. G. Mohapatra, S. K. Mohanty, and S. Nayak, "Harnessing the power of IoT and big data: Advancements and applications in smart environments," in *Internet of Things and Big Data Analytics-Based Manufacturing*, CRC Press, 2024, pp. 19–58. doi: 10.1201/9781032673479-3.

[2] S. Liu, "Case studies of big data applications for IoT," in *Empowering IoT with Big Data Analytics: A volume in intelligent data-centric systems*, Elsevier, 2024, pp. 265–311. doi: 10.1016/B978-0-443-21640-4.00010-7.

[3] H. Zhang *et al.*, "How far have edge clouds gone? A spatial-temporal analysis of edge network latency in the wild," in *IEEE International Workshop on Quality of Service (IWQoS)*, IEEE, 2023. doi: 10.1109/IWQoS57198.2023.10188741.

[4] H. Zhang *et al.*, "Large-scale measurements and optimizations on latency in edge clouds," *IEEE Trans. Cloud Comput.*, vol. 12, no. 4, pp. 1218–1231, 2024, doi: 10.1109/TCC.2024.3452094.

[5] N. Tabassum and C. R. K. Reddyy, "Review on QoS and security challenges associated with the internet of vehicles in cloud computing," *Meas. Sensors*, vol. 27, p. 100562, 2023, doi: 10.1016/j.measen.2022.100562.

[6] R. R. Nikam and D. Motwani, "Towards decentralized fog computing: A comprehensive review of models, architectures, and services," in *Lecture Notes in Networks and Systems*, vol. 818, 2024, pp. 135–147. doi: 10.1007/978-981-99-7862-5_11.

[7] X. Merino, C. Otero, D. Nieves-Acaron, and B. Luchterhand, "Towards orchestration in the cloud-fog continuum," in *IEEE SoutheastCon 2021*, IEEE, 2021. doi: 10.1109/SoutheastCon45413.2021.9401822.

[8] C.-X. Wang, Y.-Z. Shan, P.-F. Zuo, and H.-M. Cui, "Reinvent cloud software stacks for resource disaggregation," *J. Comput. Sci. Technol.*, vol. 38, no. 5, pp. 949–969, 2023, doi: 10.1007/s11390-023-3272-0.

[9] A. Bandi and J. A. Hurtado, "Edge computing as an architectural solution: An umbrella review," in *Lecture Notes in Electrical Engineering*, vol. 869, 2022, pp. 601–616. doi: 10.1007/978-981-19-0019-8_45.

[10] D. Panda, B. K. Mishra, and C. R. Panigrahi, "A study of computation offloading techniques used by mobile edge computing in an IoT environment," in *The role of IoT and blockchain: Techniques and applications*, 2022, pp. 73–86.

[11] P. K. Sharma, J. H. Ryu, K. Y. Park, J. Y. Park, and J. H. Park, "Li-Fi based on security cloud framework for future IT environment," *Human-Centric Comput. Inf. Sci.*, vol. 8, no. 1, p. Article 23, 2018, doi: 10.1186/s13673-018-0146-5.

[12] K. Sampath Kini and K. Pai, "Exploring real-time data processing using big data frameworks," *Commun. Appl. Nonlinear Anal.*, vol. 31, no. 8S, pp. 620–634, 2024, doi: 10.52783/cana.v31.1561.

[13] M. Hanif, E. Kim, S. Helal, and C. Lee, "SLA-based adaptation schemes in distributed stream processing engines," *Appl. Sci.*,

vol. 9, no. 6, p. Article 1045, 2019, doi: 10.3390/app9061045.

[14] P. S. Rawat and P. K. Soni, "Efficient virtual machine allocation technique based on hybrid approach," in *Advanced computing techniques for optimization in cloud*, 2024, pp. 87–109. doi: 10.1201/9781003457152-5.

[15] K. Netaji Vhatkar and G. P. Bhole, "Self-improved moth flame for optimal container resource allocation in cloud," *Concurr. Comput. Pract. Exp.*, vol. 34, no. 23, p. e7200, 2022, doi: 10.1002/cpe.7200.

[16] B. Tan, H. Ma, Y. Mei, and M. Zhang, "A cooperative coevolution genetic programming hyper-heuristics approach for on-line resource allocation in container-based clouds," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1500–1514, 2022, doi: 10.1109/TCC.2020.3026338.

[17] Z. Fang, H. Ma, G. Chen, and S. Hartmann, "A group genetic algorithm for energy-efficient resource allocation in container-based clouds with heterogeneous physical machines," in *Lecture Notes in Computer Science*, vol. 14472, 2024, pp. 453–465. doi: 10.1007/978-981-99-8391-9_36.

[18] J.-Y. Roh, S.-H. Choi, and K.-W. Park, "CO-TRIS: Container orchestration transforming container using resource inspection system," in *Proceedings of the 2023 IEEE International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, IEEE, 2023, pp. 121–124. doi: 10.1109/AIKE59827.2023.00027.

[19] J. A. Murali and T. Brindha, "An advanced hierarchical virtual resource management model in cloud data centers," *AIP Conf. Proc.*, vol. 2587, no. 1, p. Article 050005, 2023, doi: 10.1063/5.0150551.

[20] Z. Becvar, P. Mach, M. Elfiky, and M. Sakamoto, "Hierarchical scheduling for suppression of fronthaul delay in C-RAN with dynamic functional split," *IEEE Commun. Mag.*, vol. 59, no. 4, pp. 95–101, 2021, doi: 10.1109/MCOM.001.2000697.

[21] J. Qi, X. Su, and R. Wang, "Toward distributively build time-sensitive-service coverage in compute first networking," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 582–597, 2024, doi: 10.1109/TNET.2023.3289830.

[22] R. Kiruthiga and D. Akila, "Prediction-based cost-efficient resource allocation scheme for big data streams in cloud systems," in *Advances in Intelligent Systems and Computing*, vol. 1292, 2021, pp. 233–242. doi: 10.1007/978-981-33-4389-4_22.