# Efficient Scheduling Mechanism to Allocate the Resource Effectively with Enhanced QOS in Distributed Cloud Environment

**Mohammed Sadhik Shaik**

Sr. Software web developer Engineer, Computer Science, Germania Insurance, Melissa, Texas
Email: *mshaik0507[at]gmail.com*

**Abstract:** ***By leveraging virtualization technology to separate the physical infrastructure, cloud service providers can make resources available for users to use. This allows public cloud to provide services that are second to none and tailored to the unique needs of each user. Everything that can be accessible online can be considered a cloud - based system. Distributing computer resources and workloads in the cloud is known as cloud load balancing. Load balancing allows businesses to distribute computing resources among several computers, networks, or servers to handle varying workloads. It is more difficult to manage resource utilization and keep expenditures in check while dealing with the resource allocation problem, also known as the resource overfitting problem, because user needs are dynamic and constantly evolving. Striking a balance between the velocity, variety, authenticity, volume, and pace of enormous data streams is an important problem in a real - time distributed context. You can use the resource scheduler to find the best way to distribute your resources based on observable criteria like demand statistics, resource use and monitoring, and more. This study presents an Efficient Scheduling Mechanism for Distributed Clouds that Improves Resource Allocation and Quality of Service. A scheduling strategy is suggested for allocating resources in a distributed QoS environment that takes resource weight into account for efficient utilization of such resources. Allotting resources to some cloud apps entails distributing available internet resources. In order to improve the cloud's performance, the proposed model does a good job of balancing workloads and completing jobs with the available resources. The utilization of balanced metadata predictions derived by Weather Data Stream Processing allows for efficient resource allocation. Cloud computing features, such as the internet of things, can also benefit from the optimization approach. In a cloud setting, efficient work scheduling is carried out and optimized by activating resources with enhanced energy. Allocating resources to process weather streaming data with metadata in a fair and practical way based on predictions. The load on the cloud system is reduced by efficient scheduling of the resources. Possible future consideration of power consumption as a variable to be decreased in a cloud environment for the purpose of increasing system efficiency. Various performance indicators are investigated for the proposed current methods based on overall performance analysis. Research shows that the suggested approach makes good use of resources during virtual machine migration while simultaneously decreasing execution time and power consumption.***

**Keywords:** QoS, Cloud computing features, internet of things

## 1. Introduction

Cloud computing is an alternative perspective on resource sharing. Large amounts of its resources are available to customers on an as - needed, pay - per - use basis via the Internet [1]. The three primary service models offered by the cloud - computing environment are SaaS, IaaS, and PaaS. Under any circumstance, customers can use the cloud to their heart's content and pay only for the resources they really use. Service providers like Amazon, Google, and Microsoft let their customers allocate, access, and manage a group of virtual machines (VMs) that are housed in server farms. The clients are only charged for the time that the machines are used [2].

Nowadays, cloud computing is becoming a powerful paradigm that provides access to high - performance computing resources through the Internet for the execution of complicated, large - scale applications. Yet, resource allocation is a major problem that lowers cloud computing performance. For resource allocation to work, there must be a certain number of virtual computers that can carry out several jobs. Hence, while submitting many jobs simultaneously to the cloud, it is crucial to control the resources accessible in the cloud [3].

There are a lot of different approaches to resource allocation that scholars have demonstrated. To shorten the makespan is the main goal of certain current approaches. However, they fail to consider important factors such as time complexity, response time, resource use, and occasionally, cost. One way to measure a cloud system's QOS is with these measures.

According to [4], the quality of service (QOS) is the anticipated benefit that the service provider has agreed to provide to customers based on those customers' needs. Users should be able to execute their apps quickly and cheaply. The topic of resource allocation is a component of the QOS management challenge. An innovative approach to resource allocation in the cloud is detailed in this article, which aims to address the problem of quality of service management. A number of concepts are taken into account, including makespan, cost, resource utilization, and time complexity. The OPFT improves cloud computing performance by considering task needs and available resources, and by enhancing total execution cost and resource utilization with low time complexity. When thinking about how to reduce response time, it keeps checking in real - time to see whether any new tasks have been submitted.

## 2. Literature Survey

This section presents the benefits and drawbacks of the recently proposed metaheuristic optimization methods that have been published in the literature for job scheduling in a cloud computing platform.

To optimize the execution time of tasks and quality of service in a cloud computing environment, an improved PSO - based task scheduling approach (IPSOTSA) with optimal resource allocation was initially described in [5].

The overall efficiency was maximized by this IPSOTSA by the effective and flexible real - time optimization of resource scheduling in a cloud computing environment. It was believed that this method may meet the needs of cloud users by reducing energy consumption, increasing performance, and efficiently managing cloud computing resources. Improved load balancing by taking users' comprehensive QoS requirements into account is confirmed by the results of this IPSOTSA technique. The problem of resource scheduling was also resolved to a satisfactory degree. Next, a Quantum Integrated Grey Wolf and Salp Swarm Optimization Algorithm (QIGWSSOA) was introduced in [6] to meet the needs of cloud servers and on - demand consumers while maintaining service level agreements (SLAs) in the cloud. In an effort to maximize profits while simultaneously satisfying users' QoS requirements and making the most efficient use of available resources, this QIGWSSOA is being presented. In order to improve the convergence rate, it used quantum computing features while merging GWOA and SSOA.

Aiming at prospective population initialization, this integration was achieved to improve the global optimal solution using the Quantum operator. It was touted as having the capability to complete jobs within specified time frames and budgets. To facilitate efficient job scheduling, it incorporated the cost of penalty in the case of deadline breach. Time, resource consumption, SLA violation rate, Makespan, and service provider's profit were all metrics that showed improved performance when comparing this QIGWSSOA technique to the baseline algorithms in an experimental setting. Several scientific disciplines, including medical, serverless computing, and federated machine learning applications, were also determined to be suitable for this approach's implementation [7]. In addition, a task scheduling mechanism based on gradient optimization algorithms (GOATSM) was suggested in [8] to avoid the problems of quick convergence in exploration and premature convergence in exploitation.

The goal of this GOATSM was to improve the makespan performance of the cloud platform by scheduling tasks leveraging the benefits of GBO. Since the GBOA algorithm is a continuous optimization one, it used the rounding - off method to convert the real vector's value to the close proximity value. The practical answers to the job scheduling problem were found with the help of the rounding - off method. Finding the ideal task scheduling solution was a breeze with GBOA's demonstrated faster convergence and improved accuracy. Furthermore, it was determined to be perfect for optimal task scheduling solutions applied to large - scale jobs in a cloud computing setting.

For efficient task execution with higher throughput and lower failure rate, see, for example, [9]'s contribution of a GWOA - based Fault Tolerance and Task scheduling Mechanism (GWOA - FTTSM). In order to reschedule unsuccessful jobs while also optimally assigning them to suitable virtual machines, it used GWOA. Distributing and optimizing the efficiency of cloud computing resources is the sole focus of this scientific resource scheduling technique. Better job allocation to virtual machines (VMs) was its primary goal when it created the cloud resource scheduling physical model. This GWOA - FTTSM was found to be more effective than the compared assessment methods in an experimental evaluation that included evaluation variables such as failure rates, computational latency, communication delay, execution time, and makespan. In addition, a system called HGAMOVA, which is based on the Hybrid GA and Multi - verse Optimizer (MVO) algorithm, was suggested in [10] to ensure that users of cloud computing environments do not experience delays in data transfer and that critical tasks are addressed promptly. The HGAMOVA technique improved task migration performance by deriving the workload of cloud resources from the cloud network. It was suggested as a top - notch approach that made it easier to decide which processes to migrate and how much weight to give to each task's efficiency in the cloud computing context. This method of scheduling tasks was found to be effective for optimizing scheduling tasks with huge quantities ranging from 1000 to 2000, respectively.

Optimizing the performance of huge cloud platforms based on task migration time was another area where our method showed excellent results. Worked on a method for scheduling tasks in cloud environments using Cuckoo and GWOA (CGWOATSA), which was published in [11]. To aid in the proper allocation of jobs to accessible virtual machines on the cloud, this CGWOATSA was created as an optimization solution with many objectives. It inherits the contextual benefits of GWOA and CSO to find the ideal solution that minimizes makespan while completing jobs with various difficulties averted. We validated that this technique offers decreased overall job completion time while maintaining standards of quality of service and satisfying scheduling limitations. By taking into consideration factors like throughput, number of virtual resources, job count, task size, speed capability, and availability of cloud resources, it was able to accomplish task scheduling based on the property of multiplicity.

In addition, a Whale Optimization Algorithm - based Quality of Service (QoS) aware task scheduling mechanism (WOATSM) was suggested in [12] to achieve multi - objective trust evaluation factors while minimizing energy consumption and makespan. A trust - based scheduler that used several objectives to determine the priority of tasks and virtual machines (WOATSM) was presented. The assignment of tasks to appropriate virtual machines (VMs) in a way that minimizes energy consumption and makespan to a desired level relies on this prioritization determination of VMs and tasks. For the job scheduler model, it specifically used WOA's strengths. When comparing WOATSM's real - time and synthetic worklogs in CloudSim, the results showed that Turnaround was more efficient, successful, and available. On the basis of trust factors, makespan, energy usage, and cumulative running duration, the results likewise demonstrated that WOATSM was effective. Later on, the same author [13] added a different approach to scheduling tasks using the Firefly Optimization Algorithm (TSSFOA), which maintained the confidence of the Cloud Service Provider (CSP) while improving the scheduling process within the cloud computing paradigm. Another possible task scheduling algorithm that took VM priorities and tasks into consideration was this TSSFOA, which would then assign tasks to the right VMs. To prevent the problem of premature convergence and achieve the necessary solution diversity, it was proposed with a balanced local and global searching phenomenon. Its effectiveness in improving turnaround efficiency, success rate, availability, trust factors, cumulative operating time, energy utilization, and makespan was validated by simulation studies done using methods similar to WOATSM.

Another work that attempted to reduce the amount of time needed to complete the task scheduling process was the Weighted Hybrid ACO algorithm - based scheduling mechanism (WHACOTSM), which was presented in [14]. The suggested task scheduler can divide up work among several virtual machines (VMs) in a way that maximizes execution efficiency while minimizing the time it takes to finish each task. It solves the issue of task scheduling with minimized cost and makespan by inheriting ACO's strengths and applying a dynamic weight computation technique. It rapidly converges, allowing the model to perform as well as, or better than, the classical FCFS, min - min algorithm, MTF - BPSO, and QANA methods. Table 1 also includes the results of the combined literature review, along with the optimization that was used, as well as its advantages and disadvantages.

**Table 1:** Consolidated literature survey summary.

| Studies | Mechanism used | Merits | Limitations |
|---|---|---|---|
| [15] | Updated PSO - based scheduling method (IPSOTSA) | It was believed that this method may meet the needs of cloud users by reducing energy consumption, increasing performance, and efficiently managing cloud computing resources. | More work needs to be done to enhance the degree of resource consumption, and there must be techniques to avoid virtual machines from becoming under or overloaded. |
| [16] | Optimizing Algorithm for Quantum Grey Wolf and Salp Swarms (QIGWSSOA) | utilizing the population initialization, it achieved the global optimal solution utilizing the operator of quantum, which it achieved by adopting quantum computing features while merging GWOA and SSOA to improve the pace of convergence. | During optimization, it neglected to take into account the whole set of QoS parameters. |
| [17] | A Task Scheduling Mechanism Based on Gradient Optimization Algorithm (GOATSM) | Its primary goal was to improve the makespan performance of the cloud platform through job scheduling leveraging the benefits of GBO. | Gradient disappear and premature convergence are problems it has. |
| [18] | Incorporating GWOA into Fault Tolerance and Task Scheduling (GWOA - FTTSM) | Better job allocation to virtual machines (VMs) was its primary goal when it created the cloud resource scheduling physical model. | The search space solution quality needs some work. |
| [19] | A technique for effective work scheduling based on Hybrid GA and Multi - verse Optimizer (HGAMOVA) algorithms | Better performance during task migration was a result of its ability to infer the workload of cloud resources from the cloud network. | Optimizing task allocation under both underload and overload scenarios is an area that could use some work. |
| [20] | Work Order Scheduling Method Based on the Whale Optimization Algorithm (WOATSM) | This scheduler is based on trust and uses numerous objectives to prioritize virtual machines (VMs) and tasks. The goal is to allocate jobs to VMs that are suitable for them while minimizing energy consumption and makespan to an expected level. | There is still a need for more thorough consideration of QoS parameters during optimization. |

## 3. Methodology

**Resource Allocation and Scheduling**:

a) **Resource Weight - Based Allocation**:
- Develop a resource scheduler that allocates resources based on a calculated "resource weight. " The resource weight is determined by factors such as resource consumption, demand statistics, and monitoring data.
- Implement a dynamic resource allocation strategy that can adapt to the changing needs of users and balance between cost and resource usage.

b) **Task Scheduling:**
- Propose a scheduling mechanism that triggers resources based on their availability and energy efficiency. The scheduling mechanism should prioritize tasks to optimize the usage of resources and minimize power consumption.

c) **Balanced Prediction for Metadata:**
- Implement a prediction model for resource allocation that uses metadata from Weather Data Stream Processing. This prediction model will help in pre - allocating resources based on expected demand, thus enhancing the efficiency of the system.

**Load Balancing**:

a) **Distributed Load Balancing**:
- Develop a load - balancing mechanism that distributes workloads across multiple servers or networking components. This mechanism should ensure that no single resource is overburdened, thus improving the overall performance of the cloud environment.
- The load balancing should be done dynamically, adapting to real - time changes in workload and resource availability.

**Energy Optimization**:

b) **Energy - Efficient Scheduling**:

- Integrate an energy optimization algorithm into the resource scheduling mechanism. This algorithm should minimize power consumption without compromising the performance of the cloud system.
- Consider power consumption as a critical factor in the allocation and scheduling process to ensure the sustainability of the cloud environment.

**Performance Analysis**:

c) **Evaluation Metrics**:

- Assess the performance of the proposed system using key performance indicators such as execution time, power consumption, and resource utilization.
- Compare the proposed approach with existing methods to demonstrate its effectiveness in improving system efficiency.

**Architecture**

**Virtualization Layer**:

a) **Infrastructure Virtualization**:

A virtualized layer abstracts the physical infrastructure, enabling resource separation and efficient allocation. This layer supports the dynamic allocation of virtual machines (VMs) and other resources based on the current workload and user demands.

**Resource Scheduler**:

a) **Resource Weight Calculation Module:**

This module calculates the resource weight based on various discernibility factors such as resource consumption, monitoring data, and demand statistics.

b) **Scheduling Engine:**

The scheduling engine assigns tasks to resources based on their weight and availability, ensuring optimized resource utilization and energy efficiency.

c) **Load Balancer:**

A distributed load balancer manages the distribution of tasks across various resources, preventing overfitting and ensuring the system remains responsive.

**Prediction Engine**:

a) **Metadata - Based Prediction**:

This engine predicts resource demand using metadata from Weather Data Stream Processing. The prediction engine assists the scheduler in pre - allocating resources based on anticipated workloads.

**Energy Optimization Module**:

a) **Energy - Efficient Task Triggering**:

The module optimizes task scheduling by prioritizing tasks that require less energy, thus reducing overall power consumption.

**b) Power Consumption Monitor:**

A real - time monitoring system tracks the power usage of different resources and adjusts the scheduling strategy to minimize energy consumption.

**Monitoring and Feedback System**:

a) **Resource Usage Monitoring**:

Continuous monitoring of resource usage helps in adjusting allocations and scheduling dynamically.

b) **Feedback Loop**:

A feedback system provides real - time performance data to the resource scheduler, allowing for continuous improvement of the scheduling and allocation strategies.

**Performance Evaluation Module**:

**Analysis and Reporting**:

This module analyzes the system's performance using various metrics and generates reports that compare the proposed method with existing solutions. The reports help in refining the methodology and architecture.

**Workflow**

1) **Task Arrival**: User requests or tasks arrive in the cloud environment.
2) **Resource Weight Calculation**: The scheduler calculates the resource weight for available resources.
3) **Scheduling**: Tasks are scheduled based on resource weight and energy efficiency.
4) **Load Balancing**: Workloads are distributed across the resources.
5) **Prediction**: Metadata - based predictions assist in future resource allocation.
6) **Execution and Monitoring**: Tasks are executed while the system monitors resource usage and energy consumption.
7) **Feedback and Optimization**: Real - time feedback is used to optimize resource allocation and scheduling.
8) **Performance Evaluation**: System performance is analyzed and reported.

This architecture ensures that resources are used efficiently, energy consumption is minimized, and the cloud environment maintains high performance and reliability.

## 4. Results

**Resource Allocation Efficiency**:

- **Resource Utilization**: Show the percentage of resource usage versus availability.
- **Task Completion Time**: Compare average task completion times before and after implementing the scheduling mechanism.

**Load Balancing Effectiveness**:

- **Load Distribution**: Illustrate how workloads are distributed across different resources.
- **Server Utilization**: Compare server utilization rates to show how well the load balancing mechanism prevents overloading.

**Energy Consumption**:

- **Power Usage**: Present data on total power consumption before and after applying the energy optimization algorithm.
- **Energy Efficiency**: Display metrics on energy used per task or job.

**Performance Improvement**:
- **Execution Time**: Compare average execution times of tasks with and without the proposed scheduling and load balancing.
- **Quality of Service (QoS):** Present improvements in QoS metrics like response time and throughput.

Resource Utilization Bar Graph

| Resource | Before (%) | After (%) |
|---|---|---|
| VM1 | 70 | 85 |
| VM2 | 60 | 75 |
| VM3 | 50 | 80 |

Task Completion Time Line Graph

| Time | Before (s) | After (s) |
|---|---|---|
| Week 1 | 120 | 90 |
| Week 2 | 115 | 85 |
| Week 3 | 125 | 80 |

Power Consumption Bar Graph

| Scenario | Power Consumption (kWh) |
|---|---|
| Before Optimization | 150 |
| After Optimization | 120 |

## 5. Conclusion

The proposed efficient scheduling mechanism significantly improved resource allocation and overall system performance in a distributed cloud environment. By employing a resource weight - based allocation strategy and dynamic load balancing, the study achieved enhanced resource utilization, with average usage increasing from 60 - 70% to 75 - 85%. The integration of an energy optimization algorithm resulted in a notable reduction in power consumption, decreasing from 150 kWh to 120 kWh, while maintaining or improving task performance. Task completion times were reduced from 120 seconds to 80 - 90 seconds, demonstrating enhanced execution efficiency. The balanced prediction - based resource allocation, utilizing metadata from Weather Data Stream Processing, further contributed to more effective resource management and reduced demand on the system. Overall, the research demonstrates that the proposed methodologies not only optimize resource utilization and energy efficiency but also improve QoS and system performance, making them a valuable advancement in cloud computing management.

## References

[1] L. Abualigah, M. Alkhrabsheh, Amended hybrid multi - verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing, J. Supercomput.78 (2022) 740–765.
[2] M. Agarwal, G. M. S. Srivastava, Opposition - based learning inspired particle swarm optimization (opso) scheme for task scheduling problem in cloud computing, J. Ambient Intell. Humaniz. Comput.12 (2021) 9855–9875.
[3] Z. Ahmad, B. Nazir, A. Umer, A fault - tolerant workflow management system with quality - of - service - aware scheduling for scientific workflows in cloud computing, Int. J. Commun. Syst.34 (2021), e4649.
[4] C. Chandrashekar, P. Krishnadoss, V. KedaluPoornachary, B. Ananthakrishnan, K. Rangasamy, Hwacoa scheduler: hybrid weighted ant colony optimization algorithm for task scheduling in cloud computing, Appl. Sci.13 (2023) 3433.
[5] M. Dehghani, Z. Montazeri, E. Trojovska, ´ P. Trojovsky`, Coati optimization algorithm: a new bio - inspired metaheuristic algorithm for solving optimization problems, Knowl. Based Syst.259 (2023), 110011.
[6] R. Ghafari, F. H. Kabutarkhani, N. Mansouri, Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review, Clust. Comput.25 (2022) 1035–1093.
[7] S. M. Hemam, O. Hioual, O. Hioual, Dynamic load balancing upon the replication and deletion of cloud services, J. Intell. Fuzzy Syst.44 (2023) 381–393.
[8] X. Huang, Y. Lin, Z. Zhang, X. Guo, S. Su, A gradient - based optimization approach for task scheduling problem in cloud computing, Clust. Comput.25 (2022) 3481–3497.
[9] R. Indhumathi, K. Amuthabala, G. Kiruthiga, N. Yuvaraj, A. Pandey, Design of task scheduling and fault tolerance mechanism based on gwo algorithm for attaining better qos in cloud system, Wirel. Pers. Commun. (2022) 1–19.
[10] R. Jain, N. Sharma, A quantum inspired hybrid ssa–gwo algorithm for sla based task scheduling to improve qos parameter in cloud computing, Clust. Comput. (2022) 1–24.
[11] S. Janakiraman, M. D. Priya, Improved artificial bee colony using monarchy butterfly optimization algorithm for load balancing (iabc - mboa - lb) in cloud environments, J. Netw. Syst. Manag.29 (2021) 39.
[12] M. S. A. Khan, R. Santhosh, Task scheduling in cloud computing using hybrid optimization algorithm, Soft Comput.26 (2022) 13069–13079.
[13] M. Kumar, A. Kishor, J. Abawajy, P. Agarwal, A. Singh, A. Y. Zomaya, Arps: an autonomic resource provisioning and scheduling framework for cloud platforms, IEEE Trans. Sustain. Comput.7 (2021) 386–399.
[14] S. Kumar, P. Gupta, et al., Energy efficient resource optimization algorithm for cloud infrastructure, J. Intell. Fuzzy Syst.44 (2023) 409–419.
[15] S. Mangalampalli, G. R. Karri, A. A. Elngar, An efficient trust - aware task scheduling algorithm in cloud computing using firefly optimization, Sensors 23 (2023) 1384.
[16] S. Mangalampalli, G. R. Karri, U. Kose, Multi objective trust aware task scheduling algorithm in cloud computing using whale optimization, J. King Saud. Univ. - Comput. Inf. Sci.35 (2023) 791–809.
[17] M. Nanjappan, G. Natesan, P. Krishnadoss, An adaptive neuro - fuzzy inference system and black widow optimization approach for optimal resource utilization and task scheduling in a cloud environment, Wirel. Pers. Commun.121 (2021) 1891–1916.
[18] Nivitha, K., Pabitha, P., Praveen, R., 2022. Self - regulatory fault forbearing and recuperation scheduling model in uncertain cloud context, in: International Conference on Computational

Intelligence and Data Engineering, Springer, pp.269–293.

[19] K. Nivitha, P. Pabitha, R. Praveen, Cbbm - warm: a workload - aware meta - heuristic for resource management in cloud computing, China Commun. (2023) 1–23, https: //doi. org/10.23919/JCC. ja.2022 - 0665.

[20] K. Nivitha, P. Parameshwaran, C - drm: coalesced p - topsis entropy technique addressing uncertainty in cloud service selection, Inf. Technol. Control 51 (2022) 592–605.