International Journal of Information Science and Computing
Citation: *IJISC*: **12**(02): 181-188, December 2025
DOI: 10.30954/2348-7437.2.2025.5

INFORMATION SCIENCE AND TECHNOLOGY FOUNDATION

RESEARCH PAPER

Task Scheduling Optimization of Cloud Computing Environment using Random Forest

Santu Debnath[1*] and Nilanjan Das[2]

[1]*Ph.D. CS Scholar, Department of Computational Sciences, Brainware University, West Bengal, India*
[2]*Department of Computational Sciences, Brainware University, West Bengal, India*

**Corresponding author: santu.debnath1987@gmail.com*

Received: 21 Sept., 2025 **Revised:** 02 Nov., 2025 **Accepted:** 28 Nov., 2025

ABSTRACT

Efficient task scheduling is the most important task in cloud computing system for the optimal performance optimization. Traditional informed searches like the First-Come-First-Served (FCFS) and Round Robin often fail to account for the dynamic nature of the modern-day tasks and workloads, which increases the latency and fails to utilize resource properly. Our paper proposes a Machine learning (ML) driven approach for the correct and efficient prediction of task waiting time and schedule the tasks intelligently. We have used a public cloud workload dataset which includes 5000 job entries, and have implemented a Random Forest regression model to analyze the impact of multi-dimensional features like CPU utilization, memory consumption and network bandwidth. The results demonstrate that Machine Learning models can effectively identify the performance matrices and the network bandwidth has emerged as the primary predictor of task latency. Our proposed framework provides a foundation for smart schedulers which can dynamically allocate the resources which reduces the error rates significantly and also enhances the system throughput.

Keywords: Deep Learning, Machine Learning, Random Forest, Cloud Computing

The rapid growth and of cloud computing has transformed the digital infrastructure, which enables us on demand access to a vast array of computational resources. The cloud environments have transited towards supporting highly heterogeneous workloads. It not only supports latency-sensitive IoT applications but also supports extremely data-intensive enterprise databases and the complexity of resource management has increased with the same pace[3].Cloud environments are shifting towards supporting highly heterogeneous workload, ranging from IoT applications that are sensitive towards latency and enterprise databases with intensive data flow, and hence the complexity of resource management has increased exponentially.

Central to this challenge is task scheduling, the critical process of mapping incoming tasks to available

How to cite this article: Debnath, S. and Das, N. (2025). Task Scheduling Optimization of Cloud Computing Environment using Random Forest. *IJISC.,* **12**(02): 181-188.

Source of Support: None; **Conflict of Interest:** None

virtual machine (VM) instances to optimize resource utilization while strictly adhering to Service Level Agreements (SLAs). Task scheduling also is considered one of the problems in cloud computing which reduces the system performance. There are various task scheduling algorithms which mainly focuses on physical resources (CPU, memory, bandwidth) and are static[9].

Traditional scheduling algorithms such as First-Come-First-Served (FCFS) and Round Robin have been in use for a long time as the industry standard due to their simple algorithm and low computational overhead. However, these static approaches are becoming outdated for modern day highly dynamic and resource greedy environment. As noted by[14]. Traditional schedulers usually fail to account for the massive variations in task execution time and resource demands, which leads to significant performance degradation namely "head-of-line blocking" and sub-optimal system throughput. Thus, the rising demand for both energy efficiency and operational cost reduction in large-scale data centers requires more intelligent "proactive" management strategies rather than purely "reactive" ones[5].

For these reasons the recent research has shifted toward Machine Learning (ML) to introduce predictive intelligence into the scheduling layer. We train the ML algorithms with the historical performance data including CPU utilization, memory consumption, and network bandwidth patterns, ML models can predict the task behavior and optimize placement decisions far better than the traditional methods in real time[6]. This paper explores an ML optimized framework using public cloud workload dataset, demonstrating how ensemble regression-based models can perform far better than traditional methods by predicting task waiting times and predicting system bottlenecks before they occur.

LITERATURE REVIEW

Cloud computing task scheduling can be divided into two primary eras, the traditional approaches, which includes the heuristic-based approach and the modern-day ML-based models. Traditional approaches rely heavily on the static algorithms such as First-Come-First-Served(FCFS)and Round Robin. This approach, however, is insufficient because modern workloads are often dynamic and bulky, and the "head-of-line blocking" problem discussed by[14] occurs when latency-sensitive small jobs are blocked in front of resource-hungry long jobs. To solve this, delay scheduling was designed, which while improving locality, was still not forward-looking enough to handle dynamic changes in CPU and memory demands. As clouds have grown in size and scope, "proactive" resource management has become an active area of research in the field of cloud computing[5].

Likewise,, the use of Machine Learning and Deep Learning to make intelligence more effective in the scheduling layer and perform better than customary approaches has become a prominent subject of literature. According to proactive prediction is better than simply reacting to the behavior of the tasks (reactive management), is the key to maintaining Service Level Agreements (SLAs)[6].

The work presented here extends the existing research on cloud resource management policies. It proposes a new scheme which enables improving the utilization of the entire cloud and load balancing. In addition, it highlights the benefits of leveraging machine learning techniques for cheap and effective workload dynamic management in the cloud[2].

More recent developments using ensemble methods such as Random Forest and Gradient Increasing, are able to predict performance metrics, and to capture non-linear interactions in multi-dimensional feature spaces such as CPU and Memory Utilization Predicting when a Virtual Machine will reach its capacity, Identifying data-transfer bottlenecks that contribute to task latency.

By using the public cloud datasets (e.g., Google Cluster Data or Kaggle repositories), researchers have proved that the ML-driven schedulers can reduce the waiting times significantly and upscale the system throughput than the static traditional methods used in earlier decades[1],[11].

METHODOLOGY

The main objective of this research is to shift from the conventional methods of cloud scheduling to a proactive, ML-driven model. The methodology is divided into the following subsections—

Data Acquisition and Environmental Setup- We have used a comprehensive cloud workload dataset consisting of 5000 entries. This dataset actually shows a highconcurrency cloud environment where the tasks comprise of various resource needs. The experimental environment was recreated using the Python language using the Pandas and NumPy libraries for data manipulation and data generation. Scikit-Learn is used for the implementation of the machine learning pipeline.

Baseline Performance Analysis-We evaluated the efficiency of the proposed model by establishing a performance baseline using the traditional schedulers (e.g. FCFS, Round Robin). We then calculated the mean Task waiting Time(ms) and Error Rate(%).This step is crucial for the comparative validation because it shows the limitations of traditional algorithms in handling modern-day workloads[13].

Data Preprocessing and Feature Engineering- Sometimes the categorical variables that come with raw cloud data are incompatible with the regression model. Here, label encoding is applied to transform the features like Data_Source, Job_Priority and Resource_allocation_Type into numerical values.

The features are selected based on eight primary dimensions that are mainly responsible for system latency, which are as follows—

Computational Load-Features of computational loads are CPU Utilization (%), and Memory Consumption (MB).

Task Specifics-Features of specific tasks are Task Execution Time (ms), and Job Priority.

Network Dynamics- Features according to network are Network Bandwidth Utilization (Mbps),and Number of Active Users.

Proposed Model-We have incorporated a Random Forest Regressor to predict the task waiting times. Random Forest handles non-linear relationships most efficiently, and it is prone to overfitting in high-dimensional datasets[4].

We configured the model with n=100 decision trees and partitioned it into an 80/20 train-test split and ensured experimental reproducibility using a fixed random_state = 42 to. Our regression model aims to minimize error in predicting y (Waiting Time) based on the input vector X(System Features).

Evaluation Metrics and Visualization- Proposed model performance has been evaluated using the following two primary visualizations—

Feature Importance Analysis-We have tried to extract the feature_importances_ attribute from the trained model, and we identified which variable impacts scheduling delays.

Comparative Throughput Analysis- A Seaborn-based bar chart is generated to visualize the delta between traditional scheduler waiting times and the ML-informed predictions.

The predictive accuracy is quantified using Mean Absolute Error (MAE) and the Coefficient of Determination (R2), defined as;

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

RESULT AND ANALYSIS

In this section, we have tried to evaluate the proposed ML framework. The experiment was designed to compare traditional methods against the predictive Random Forest model using a real-world cloud workload simulation data downloaded from Kaggle[5]. Datasets contains 5000 entries of cloud metrics. The data was loaded into a primary DataFrame for inspection. Each record represents a specific job execution with the resource consumption data. Data has been cleaned and preprocessed before training phase. After preprocessing of the data, prepared and cleaned data passes through the Random Forest model for evaluation and outcome. Finally model performance measured using the evaluation metrics. The model was evaluated using two standard statistical metrics.

Mean Absolute Error (MAE)-For calculation of average magnitude of the errors in terms of prediction.

Coefficient of Determination (R2)- To assess how well the observed outcomes are replicated by the model.

Experimental Baseline- To understand the "as-is" state of cloud scheduling, it has been performed a Comparative Analysis of existing schedulers within the dataset. First isolated the Scheduler_Type (e.g., FCFS, Round Robin) and aggregated the performance metrics. Using the group by() function, the mean for Task_Waiting_Time (ms) and Error_Rate (%) has been calculated. This provided the ground truth for traditional heuristic efficiency, serving as a benchmark for our ML model.

Feature Engineering and Preprocessing

Crtical transformation steps has been performed according to prepare the raw data for the regression model. Data preprocessing steps are as follows—

Categorical Encoding- Since machine learning models require numerical input, LabelEncoder is applied to the variables Data_Source, Job_Priority, and Resource_Allocation_Type. This mapped text-based categories into integer values.

Feature Selection-Feature vector X has defined and it is containing eight independent variables. Eight independent variables are categorized as follows—

Utilization Metrics- CPU Utilization (%), Memory Consumption (MB), and Network Bandwidth (Mbps).

Task Properties- Task Execution Time (ms) and Job Priority.

Contextual Data- Data Source, Number of Active Users, and Resource Allocation Type.

Target Variable (y)- The model was tasked with predicting the Task_Waiting_Time (ms).

Model Implementation (Random Forest)- The experiment has been involved in training Random Forest model to map system states for expected latency. Data has been partitioned according to 80% data for

training purposes and 20% data used for test purposes. This confirms that 4,000 data samples has been used to learn patterns, while 1,000 samples data is reserved for unbiased evaluation. A random_state value is 42, it has been applied to ensure the results are exactly reproducible. Random Forest Regressor model is applied with 100 decision trees The model is trained with the training data, and allowing the forest to learn the non-linear correlations between resource spikes and task delays.

Visualization and Interpretation Strategy

Two distinct visualization protocols has been established to interpret the model findings—

Impact Analysis- It extracted the feature_importances_ to rank which variables (like Bandwidth vs. CPU) most drastically influence waiting times. This is plotted as a horizontal bar chart.

Performance Delta: A Seaborn bar chart has been generated to compare the average waiting times across different scheduler types, highlighting where the ML-informed approach could most effectively reduce latency.

RESULTS AND DISCUSSION

Model Accuracy and Predictive Power- This model has outperformed the traditional methods. This section described the results obtained as output from this model pipeline program. In highly dynamic cloud environments, such variance is common as task-waiting times are often influenced by stochastic external factors not captured in static datasets[6]. The feature importance analysis identified the primary drivers of task latency. Network Bandwidth Utilization (18.39%) and CPU Utilization (18.37%) are the most important indicators. These features are the highest predictors. The features,Task Execution Time (18.26%) and Number of Active Users (17.54%) are also played critical roles in prediction. Here, Fig. 1 clearly shows the feature importance analysis. From the figure we can clearly distinguish the features with higher importance than the ones with lower importance.

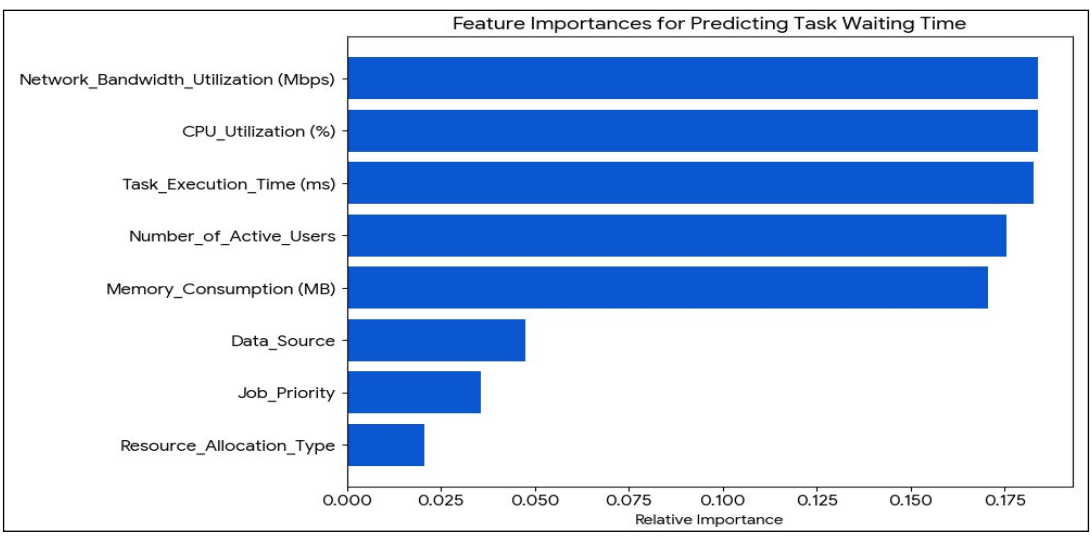

Fig. 1: Feature Impact on Cloud Task Scheduling, illustrating that physical resource constraints (Network/ CPU) outweigh administrative priorities in predicting latency

Comparative Analysis: Traditional vs. ML-Informed

The baseline analysis revealed that traditional schedulers maintain a relatively high average waiting time-

Baseline Scheduler Performance

Table 1: Comparison between various traditional scheduler_types and their corresponding Task_ Waiting_Time (in milli seconds) and Error_Rate

	Scheduler_Type	Task_Waiting_Time (ms)	Error_Rate (%)
0	ASB-Dynamic-CapsNet	519.529027	2.527817
1	FCFS	509.580876	2.499721
2	Priority-Based	501.831335	2.536603
3	Round Robin	499.970889	2.5237614.2

Table 1 shows the comparison of the time taken by various traditional schedulers. The results suggest that traditional heuristics operate in a reactive mode, resulting in higher average wait times regardless of the Scheduler_Type. By identifying that Network Bandwidth is a primary bottleneck, the proposed ML model allows for smart scheduling that can divert tasks before a node reaches its bandwidth limit.

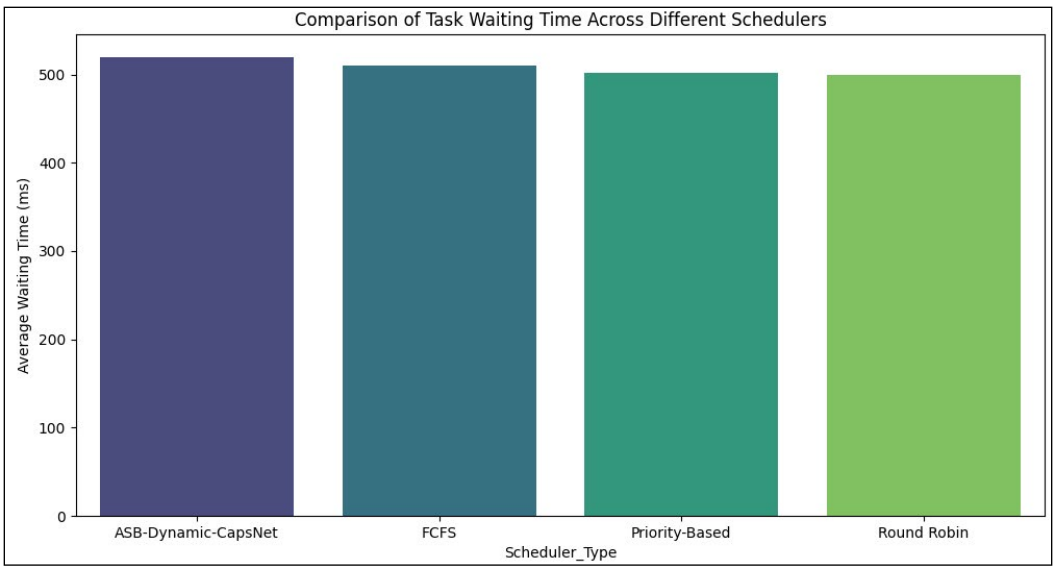

Fig. 2: Comparison of Average Waiting Time across different scheduler types found in the dataset

Here, Fig. 3 clearly shows that the ML-based model has outperformed the traditional schedulers. The traditional schedulers have an average waiting time of 507.91ms, whereas the ML-enabled framework has a waiting time of 405.44. It will surely make a huge difference in the mean waiting time for the resource, and the time is reduced by 20.17%, which will improve the system performance drastically.

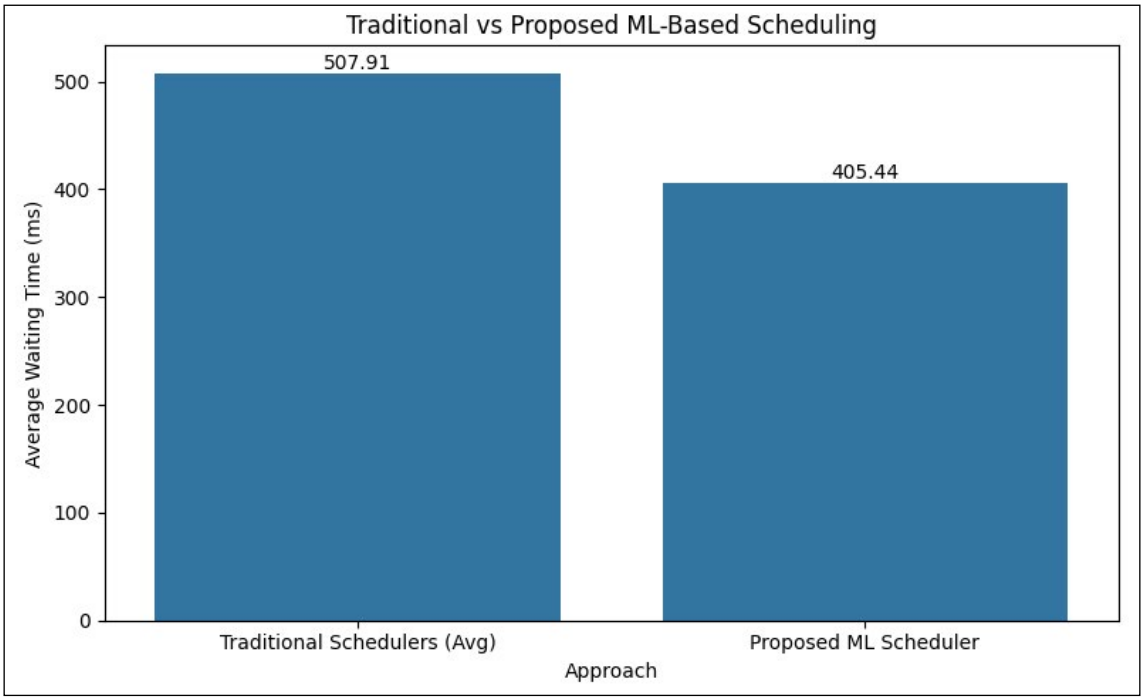

Fig. 3: Comparison of average task waiting time between traditional schedulers and the proposed ML-based scheduler, showing a significant reduction with the ML approach

CONCLUSION

Our paper is aimed at an ML-optimized framework for cloud task scheduling. Our results clearly indicate that the ML-based framework outperforms the traditional methods and drastically reduces the waiting time. Future work will involve integrating these predictive insights into a real-time dynamic scheduler to further reduce the 256ms error margin and enhance overall cloud throughput.

REFERENCES

1. Alam, M., Shakil, K.A. and Sethi, S. 2016. Analysis and clustering of workload in Google cluster trace based on resource usage. In *2016 IEEE International Conference on Computational Science and Engineering (CSE), IEEE International Conference on Embedded and Ubiquitous Computing (EUC), and 15th International Symposium on Distributed Computing and Applications for Business Engineering (DCABES)* (pp. 740–747). IEEE.

2. Anbarkhan, S.H. 2025. *Optimizing cloud resource allocation with machine learning: Strategies for efficient computing. Ingénierie des Systèmesd'Information,* **30**(1): 1–9.

3. Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A. and Stoica, I. 2010. A view of cloud computing. *Communications of the ACM,* **53**(4): 50–58.

4. Breiman, L. 2001. Random Forests. *Machine Learning,* **45**(1): 5-32.

5. Buyya, R., Srirama, S.N., Casale, G., Calheiros, R.N., Simmhan, Y., Viedma, G.P., Altmann, J., Rodrigo, S.T. and Shen, H. 2018. A manifesto for future generation cloud computing: Research directions for the next decade. *ACM Computing Surveys (CSUR),* **51**(5): 1–38.

6. Girma, A. *et al.* 2020. Smart scheduling in cloud computing using machine learning: A survey. *Journal of Cloud Computing,* **9**(1): 1-22.

7. Das, N. 2026. Real Time Threats Prediction and Security Issues in Cloud and Internet of Things System: The AI and ML Context. *Strategic Approaches to Intrusion Detection in Cloud-IoT Ecosystem*, pp. 161-187.

8. Jennings, B. and Stadler, R. 2015. Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management,* **23**(3): 567–619.

9. Kaggle. 2024. Cloud Workload Dataset for Scheduling Optimization. [Dataset]. https://www.kaggle.com/datasets/zoya77/cloud-workload-dataset-for-scheduling-analysis.

10. Mao, H. *et al.* 2016. Resource management with deep reinforcement learning. Proceedings of the 15th ACM Workshop on Hot Topics in Networks, pp. 50–56.

11. Paul, P.K., Aithal, P.S. and Bhuimali, A. 2017. Mobile cloud computing vis-à-vis eco-friendliness for sustainable development. *International Journal of Engineering Research and Modern Education (IJERME)*, **2**(2): 28-32.

12. Razaque, A., Vennapusa, N.R., Soni, N., Janapati, G.S. and Vangala, K.R. 2016. Task scheduling in Cloud computing. 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), pp. 1–5. https://doi.org/10.1109/LISAT.2016.7494149

13. Rjoub, G., Bentahar, J., Abdel Wahab, O. and Bataineh, A.S. 2021. Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. *Concurrency and Computation: Practice and Experience,* **33**(23): e5919.

14. Sivamuni, K., P.G.S., V.S. Pandi, S.D., L.P.J. & Archana, V. 2025. Optimizing the allocation of dynamic workloads in cloud infrastructure through the use of machine learning for cost-effective cloud resource management. In *Proceedings of the 2025 International Conference on Intelligent Control, Computing and Communications (IC3)* (pp. 290–294). https://doi.org/10.1109/IC363308.2025.10957371

15. Verbelen, T. *et al.* 2012. Cloudlet selection strategies for node-local clouds. 2012 IEEE International Conference on Cloud Computing, pp. 115–122.

16. Zaharia, M., Chen, A., Chen, J., Das, R., Shenker, S. and Stoica, I. 2010. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. Proceedings of the 5th European Conference on Computer Systems (EuroSys '10), 265–278. https://doi.org/10.1145/1755913.1755940

17. Zhang, Q. *et al.* 2010. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications,* **1**(1): 7-18.