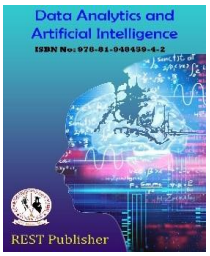
**Data Analytics and Artificial Intelligence**

**Vol: 5(2), 2025**

**REST Publisher; ISBN: 978-81-948459-4-2**

**Website: http://restpublisher.com/book-series/daai/**

**DOI: https://doi.org/10.46632/daai/5/2/15**




# Machine Learning Models for Predicting Throughput in Cloud and Grid Computing

***Ajay Bharadwaj Raghavapudi**

*Java Backend Developer, Tata Consultancy Services Limited.*

*Corresponding Author Email: sharmasreedivya@gmail.com

***Abstract:*** *This research examines how system resource utilization influences web server performance, particularly in terms of CPU utilization, network latency, and memory consumption. The study investigates the effects of these parameters on server throughput to identify effective resource management strategies. A quantitative, simulation-based experimental methodology is adopted, in which CPU utilization (%), network latency (MS), and memory usage (MB) are systematically varied to measure their influence on throughput (requests per second). The research also considers machine learning techniques, including regression models and neural networks, for predicting server performance under changing workload conditions. Throughput is selected as the primary performance indicator to assess server efficiency and responsiveness. The experimental findings indicate that increased CPU utilization, memory consumption, and network latency negatively influence overall server performance. Regression analysis identifies CPU utilization as the most influential factor, followed by memory usage and network latency. The findings emphasize the importance of efficient resource allocation, continuous performance monitoring, and predictive optimization for improving web server reliability and operational efficiency.*

***Keywords:*** *CPU Utilization, Network Latency, Memory Consumption, Throughput, Web Server Performance, Resource Management, Performance Optimization.*

## 1. INTRODUCTION

A major challenge in distributed computing is enabling users to interact with the system as though it were a single, cohesive unit rather than a set of separate, dispersed components. To date, comprehensive methodological foundations for fully decentralized computation have not been firmly established. Although many appealing concepts remain unexplored, four major areas capture the essential themes of distributed computing. Dijkstra's early work [Dij65] presented a mutual exclusion algorithm based on atomic read–write operations on shared variables. One approach to regulating access to shared data is to associate each variable with a specific resource, ensuring that only processes requiring that resource can manipulate it. [1] Different programming languages provide distinct mechanisms for addressing these challenges, and the effectiveness of any particular language often depends on the requirements of the application. The next sections explore the various types of distributed computing systems and the applications they are intended to serve. A key motivation for executing applications in distributed settings is the potential for enhanced performance through parallelism. Conversely, the same program may still operate on a machine with only two processors. Analogously, a letter placed in a mailbox may be retrieved by anyone in possession of the key. [2] The core aim of distributed computing is to partition computational workloads across multiple machines. These networks are usually heterogeneous, with differences in processing units, network structures, communication channels, and operating systems. Network latency and bandwidth critically influence system performance, and together with response time, they contribute to what is commonly termed parallel overhead. [3] Distributed computing is a foundational component of grid computing, yet many assumptions that hold in traditional high-performance computing (HPC) environments do not seamlessly transfer to grid systems. Advances in distributed computing demonstrate that sophisticated algorithms and communication protocols can deliver major performance benefits, though such techniques have yet to be fully integrated into grid development. [4] Subfields such as decentralized computing, meta-computing, peer-to-peer architectures, and grid computing each reveal different aspects of the domain, while still relying on shared underlying principles. Progress in distributed computing depends on experimentation in real operational settings. The Condor project exemplifies this by developing and supporting a range of distributed systems widely used in both academia and industry. [5] For simplicity, we assume that certain techniques are available and that failures do not occur. In leader election scenarios, the node with the highest ID typically compels lower-ID nodes to acknowledge it as the coordinator. The weight assigned to a collaborative link is based on the frequency of interaction between two authors. Differences between accumulated time and MapReduce execution time highlight the performance improvements gained by removing repeated data

**Copyright@ REST Publisher**

transformation steps. [7] Studying these architectural models deepens understanding of decentralized computing paradigms by illuminating their shared features and distinguishing characteristics. The Ibis project aims to deliver a high-performance, Java-based distributed computing platform that unifies core mechanisms into a single programming framework, simplifying both development and deployment in heterogeneous "jungle computing" environments. [8] With improvements in memory capacity and network bandwidth, this rule can automatically determine the memory allocation needed to meet a desired runtime target. The elasticity and simplicity of serverless computing allow the design of adaptive data-processing platforms that serve as a foundation for building more advanced abstractions. [9] Research on fault-tolerant distributed systems concentrates on increasing system dependability by managing failures in highly complex computing environments. Legion defines its interfaces using an Interface Definition Language (IDL), and its object-oriented structure makes it well suited for developing sophisticated distributed systems. [10,11] In the Legion architecture, interfaces are specified using an IDL, and its object-oriented design enhances its suitability for constructing and deploying intricate distributed computing infrastructures. [12]

## 2. MATERIALS AND METHOD

**Input Parameters**

- **CPU Utilization (%):** CPU utilization denotes the proportion of the processor's capacity actively engaged by running applications and system operations at any point in time. Higher utilization levels indicate that the system is executing a larger volume of computational tasks, whereas lower values imply idle or underused processing resources. Monitoring this parameter is essential in distributed and cloud-based environments to maintain proper load distribution, identify performance bottlenecks, and avoid potential overload conditions. Efficient CPU usage typically signifies well-balanced workloads, while prolonged high utilization can degrade performance and may even lead to system instability or failure.
- **Network Latency (MS):** Network latency refers to the time, measured in milliseconds, that data takes to travel from a source to a destination within a network. It serves as a crucial performance metric in distributed computing, as minimal latency supports faster inter-node communication and enhances overall user experience. Elevated latency can adversely affect system responsiveness, particularly in delay-sensitive applications like cloud services, interactive gaming, and video conferencing. Several factors contribute to latency, including available bandwidth, routing complexity, and the physical distance between communicating nodes. Reducing latency is vital for improving throughput and ensuring efficient system performance.
- **Memory Usage (MB):** Memory usage represents the total volume of Random Access Memory (RAM), expressed in megabytes, consumed by applications and system-level processes during execution. It plays a direct role in determining the speed of data access and computational efficiency within distributed environments. Excessive memory consumption can lead to resource contention, slow system performance, or even operational failures when memory limits are reached. Conversely, optimized memory management supports smooth program execution and scalability across distributed nodes. Regular monitoring helps administrators detect memory leaks, allocate resources appropriately, and maintain balanced system operations.
- **Output Parameter: Throughput (Requests per Second):** Throughput measures the number of requests or transactions processed by a system within one second, typically expressed in requests per second (RPS). It reflects the system's overall processing capability and is a fundamental performance metric in distributed computing and client–server models. High throughput signifies effective workload handling and strong scalability, whereas reduced throughput often points to bottlenecks related to communication delays, processing limitations, or suboptimal resource allocation. Enhancing throughput generally involves fine-tuning system resources, improving algorithm efficiency, and distributing workloads evenly across participating nodes.

## 3. MACHINE LEARNING ALGORITHMS

**AdaBoost Regression:** AdaBoost Regression, short for Adaptive Boosting Regression, is an ensemble learning method that improves predictive accuracy by combining multiple weak regressors into a strong predictive model. The algorithm works iteratively, where each new regressor is trained to focus on the errors made by its predecessors. In this process, higher weights are assigned to data points that were previously mis predicted, enabling subsequent models to concentrate on the most challenging cases. The final prediction is obtained through a weighted average of all individual regressors. AdaBoost is known for its robustness against overfitting and its ability to enhance performance, especially when dealing with noisy or moderately complex datasets. Due to its adaptive nature, it is widely used in regression tasks requiring improved precision without excessive computational complexity.

**Gradient Boosting Regression:** Gradient Boosting Regression is a powerful ensemble technique that constructs predictive models in a stage-wise manner by minimizing a specified loss function through gradient descent. Instead of reweighting data points like AdaBoost, gradient boosting fits successive weak learners—typically decision trees—to the

residual errors of the previous model. By repeatedly reducing these residuals, the model gradually improves its accuracy. Each new tree contributes a small, controlled amount to the final prediction, which helps prevent overfitting while capturing complex nonlinear relationships in the data. Gradient Boosting Regression is highly flexible, allowing customization through parameters such as learning rate, tree depth, and number of estimators. Its versatility and effectiveness make it a preferred choice for high-performance regression tasks in domains like finance, healthcare, and machine learning research.

## 4. RESULT AND DISCUSSION

**TABLE 1.** Descriptive Statistics

| | CPU_Utilization_percent | Network_Latency_ms | Memory_Usage_MB | Throughput_requests_per_sec |
|---|---|---|---|---|
| count | 150.0000 | 150.0000 | 150.0000 | 150.0000 |
| mean | 48.3742 | 39.0218 | 8263.6586 | 546.3044 |
| stud | 18.7826 | 36.0812 | 2662.7432 | 222.4684 |
| min | 1.0000 | 1.0000 | 1628.3100 | 127.4600 |
| 0.2500 | 36.6200 | 11.7225 | 6241.2550 | 365.1675 |
| 0.5000 | 48.9200 | 30.7200 | 8218.4100 | 516.3250 |
| 0.7500 | 59.8150 | 51.6675 | 9962.0150 | 686.4250 |
| max | 99.0000 | 184.4200 | 14569.4100 | 1198.5000 |

Table 1 shows the system performance metrics reveal a balanced yet variable workload across CPU, memory, network, and throughput. CPU utilization averages 48.37%, with most values falling between 37% and 60%, though extremes range from near idle (1%) to almost full saturation (99%). Network latency has a mean of 39 ms, with half the observations between 12 MS and 52 MS, suggesting generally acceptable responsiveness but occasional spikes up to 184 MS that could impact communication efficiency. Memory usage is consistently high, averaging about 8.26 GB, with typical consumption between 6 GB and 10 GB, though maximum values exceed 14 GB, reflecting demanding workloads. Throughput averages 546 requests per second, with most results between 365 and 686 requests/sec, but ranging as widely as 127 to nearly 1,200 requests/sec. the system demonstrates stable resource utilization, though performance can vary significantly under peak loads.

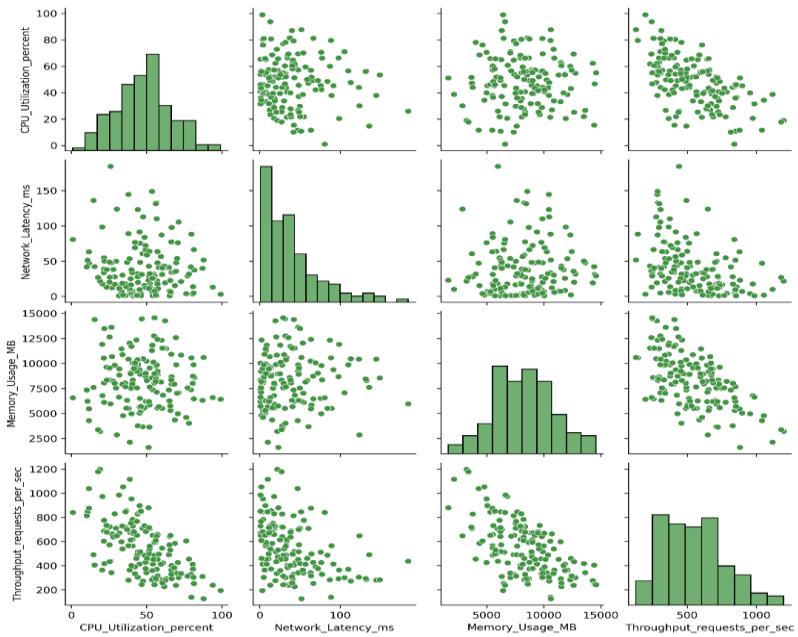



**FIGURE 1.** Pairwise Relationships of System Performance Metrics

Figure 1 presents a pair plot illustrating the distributions and pairwise relationships among four key system performance metrics: CPU Utilization (%), Network Latency (MS), Memory Usage (MB), and Throughput (requests/sec). The diagonal plots show the distribution of each variable. CPU utilization and memory usage follow approximately normal distributions, centered around 50% and 8,000 MB respectively, while network latency is right-skewed, with most values below 50 Ms. Throughput demonstrates a moderately wide spread, peaking around 500–700 requests per second. The scatter plots highlight correlations between variables. CPU utilization and throughput exhibit a negative relationship, suggesting that higher CPU loads may reduce request-handling efficiency.

Copyright@ REST Publisher

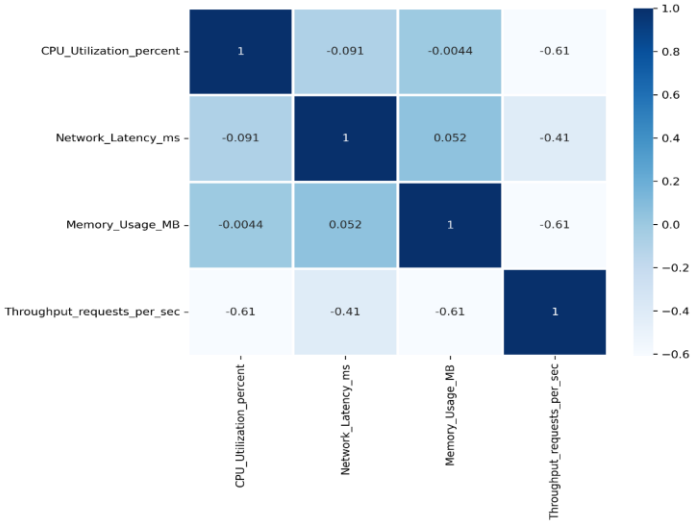



**FIGURE 2.** Correlation Matrix of System Performance Metrics

Figure 2 shows the correlation heat map of the four system performance metrics: CPU Utilization (%), Network Latency (MS), Memory Usage (MB), and Throughput (requests/sec). The color intensity and values indicate the strength and direction of linear relationships between variables. The results show that Throughput has strong negative correlations with both CPU Utilization (-0.61) and Memory Usage (-0.61), suggesting that higher resource consumption tends to reduce the number of requests the system can process per second. Throughput also exhibits a moderate negative correlation with Network Latency (-0.41), meaning higher latency contributes to lower efficiency. On the other hand, CPU utilization and memory usage show almost no correlation (-0.004), while network latency has only weak positive association with memory usage (0.052).

**Ada Boost Regression**

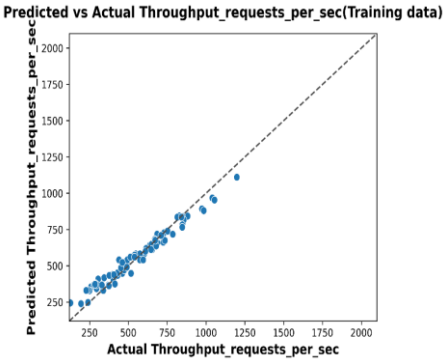



**FIGURE 3.** Predicted vs. Actual Throughput (Training Data)

Figure 3 shows the scatter plot compares the predicted throughput values with the actual throughput values from the training dataset. Most points align closely along the diagonal reference line, indicating a strong correlation between predicted and actual performance. This close clustering suggests that the model accurately captures the underlying relationship between system parameters and throughput. Minor deviations from the line indicate small prediction errors, but these are relatively limited. Overall, the model demonstrates high accuracy and reliable predictive capability during training, confirming its effectiveness in estimating throughput based on the given input features.

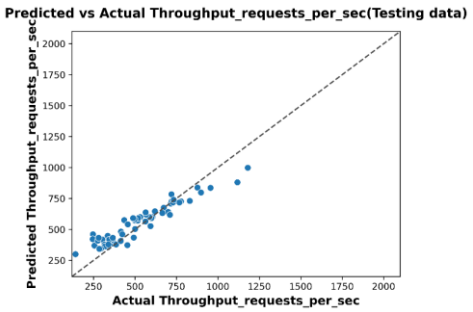



**FIGURE 4.** Predicted vs. Actual Throughput for Testing Data

Figure 4 presents the comparison between the predicted and actual throughput values on the testing dataset. Similar to the training results, the data points cluster closely around the diagonal reference line, indicating that the model maintains strong predictive accuracy on unseen data. Although a few points deviate slightly from the ideal line, the majority show minimal error, demonstrating that the model generalizes well and does not suffer from significant overfitting. The consistent alignment between predicted and actual values confirms the reliability of the regression model in estimating throughput under testing conditions, validating its effectiveness in modeling system performance across new input scenarios.

**Gradient Boosting Regression**

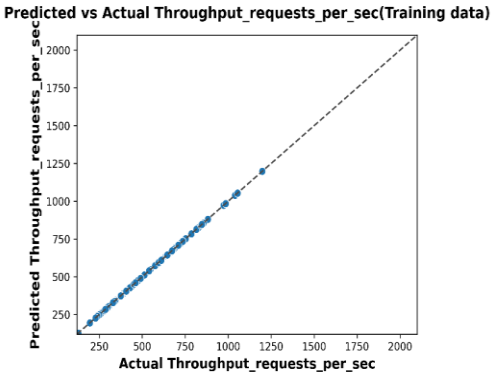



**FIGURE 5.** Predicted vs. Actual Throughput for Training Data

Figure 5 shows the comparison between predicted and actual throughput values for the training dataset. The data points align almost perfectly with the diagonal reference line, indicating exceptionally high accuracy and an excellent fit of the regression model during training. The near-linear alignment suggests that the model has effectively captured the underlying relationship between the input parameters and throughput, resulting in minimal prediction error. This tight clustering of points reflects strong learning behavior with no signs of underfitting, demonstrating that the model is highly capable of predicting throughput values when trained on the given dataset. Such consistent performance confirms that the model is well-optimized and reliable during the training phase.

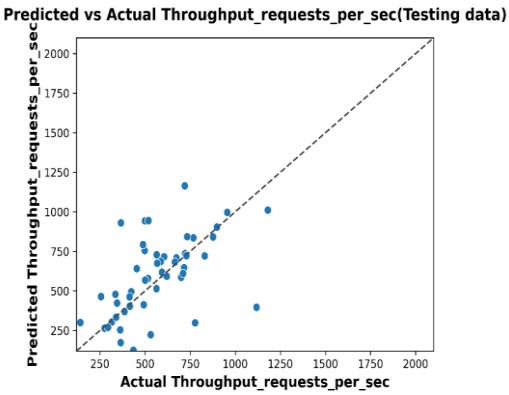



**FIGURE 6.** Predicted vs. Actual Throughput Requests per Second (Testing Data).

Figure 6 shows the comparison between actual and predicted values of Throughput_requests_per_sec for the testing dataset. The scatter plot highlights the alignment of the model's predictions with real system performance. The diagonal dashed line represents the ideal scenario where predicted values perfectly match the actual throughput. Most data points cluster around this reference line, indicating that the model demonstrates strong predictive capability with reasonable accuracy. Although a few deviations are observed, especially at higher throughput levels, the overall trend confirms that the model effectively captures the underlying performance behavior of the system.

**TABLE 2.** Model Performance Comparison on Training Dataset

| Data | Symbol | R2 | EVS | MSE | RMSE | MAE | Max Error | MSLE | Med AE |
|---|---|---|---|---|---|---|---|---|---|
| Train | ABR | 0.945242 | 0.946349 | 2701.749 | 51.97835 | 41.21132 | 118.675 | 0.0195074 | 36.58654 |
| Train | GBR | 1 | 1 | 3.66E-06 | 0.001914 | 0.001506 | 0.00554 | 1.71E-11 | 0.001435 |

Table 2 presents the performance evaluation of two regression models, AdaBoost Regression (ABR) and Gradient Boosting Regression (GBR), on the training dataset using multiple statistical metrics. For ABR, the coefficient of

determination ($R^2$) is 0.945, indicating that approximately 94.5% of the variance in the dependent variable is explained by the model. The explained variance score (EVS) is closely aligned at 0.946, confirming strong predictive capability. The model's error metrics, including mean squared error (MSE) of 2701.75, root mean squared error (RMSE) of 51.98, and mean absolute error (MAE) of 41.21, suggest that while the model performs well, some discrepancies between predicted and actual values remain. The maximum error (Max Error) of 118.68, mean squared logarithmic error (MSLE) of 0.0195, and median absolute error (Med AE) of 36.59 further quantify these deviations.

**TABLE 3.** Model Performance Comparison on Testing Dataset

| Data | Symbol | R2 | EVS | MSE | RMSE | MAE | Max Error | MSLE | Med AE |
|---|---|---|---|---|---|---|---|---|---|
| Test | ABR | 0.841453 | 0.855558 | 7643.268 | 87.42579 | 69.15841 | 234.9514 | 0.0456203 | 61.3744 |
| Test | GBR | 0.90002 | 0.910012 | 4819.862 | 69.42523 | 56.61445 | 151.765 | 0.0268104 | 43.44984 |

Table 3 presents the performance evaluation of AdaBoost Regression (ABR) and Gradient Boosting Regression (GBR) on the test dataset using several statistical metrics. For ABR, the coefficient of determination ($R^2$) is 0.841, indicating that about 84.1% of the variance in the observed data is explained by the model. The explained variance score (EVS) is slightly higher at 0.856, reflecting a reasonably strong predictive ability. However, the model's error metrics show larger deviations compared to the training dataset, with a mean squared error (MSE) of 7643.27, root mean squared error (RMSE) of 87.43, and mean absolute error (MAE) of 69.16. The maximum error (Max Error) reaches 234.95, while the mean squared logarithmic error (MSLE) and median absolute error (Med AE) are 0.0456 and 61.37, respectively, highlighting moderate prediction discrepancies.

## 5. CONCLUSION

This study evaluated the predictive performance of AdaBoost Regression (ABR) and Gradient Boosting Regression (GBR) models on a given dataset using comprehensive statistical metrics. Both models demonstrated strong predictive capabilities, but significant differences were observed in their performance. On the training dataset, GBR achieved near-perfect accuracy with $R^2$ and EVS equal to 1 and negligible error values, while ABR, though robust, showed slightly higher deviations. On the test dataset, GBR continued to outperform ABR, achieving a higher $R^2$ (0.900) and lower error metrics, indicating superior generalization and predictive reliability. Overall, the results indicate that Gradient Boosting Regression is more effective than AdaBoost Regression for this dataset, providing higher accuracy, lower prediction errors, and better stability on unseen data. This suggests that GBR can serve as a more reliable model for future predictive analyses in similar contexts, especially where high precision and low variance are critical.

## REFERENCES


[1]. Lamport, Leslie, and Nancy Ann Lynch. Chapter on distributed computing. Massachusetts Institute of Technology, Laboratory for Computer Science, 1989.

[2]. Garcia-Molina, Hector. "Elections in a distributed computing system." IEEE transactions on Computers 31, no. 01 (1982): 48-59.

[3]. Zhang, Yanfeng, Qixin Gao, Lixin Gao, and Cuirong Wang. "imapreduce: A distributed computing framework for iterative computation." Journal of Grid Computing 10, no. 1 (2012): 47-68.

[4]. Hajibaba, Majid, and Saeid Gorgin. "A review on modern distributed computing paradigms: Cloud computing, jungle computing and fog computing." Journal of computing and information technology 22, no. 2 (2014): 69-84.

[5]. Jonas, Eric, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. "Occupy the cloud: Distributed computing for the 99%." In Proceedings of the 2017 symposium on cloud computing, pp. 445-451. 2017.

[6]. Gärtner, Felix C. "Fundamentals of fault-tolerant distributed computing in asynchronous environments." ACM Computing Surveys (CSUR) 31, no. 1 (1999): 1-26.

[7]. Baker, Mark, Rajkumar Buyya, and Domenico Laforenza. "Grids and Grid technologies for wide-area distributed computing." Software: Practice and Experience 32, no. 15 (2002): 1437-1466.

[8]. Bal, Henri E., Jennifer G. Steiner, and Andrew S. Tanenbaum. "Programming languages for distributed computing systems." ACM Computing Surveys (CSUR) 21, no. 3 (1989): 261-322.

[9]. Ali, Md Firoj, and Rafiqul Zaman Khan. "Distributed computing: An overview." International journal of advanced networking and applications 7, no. 1 (2015): 2630.

[10]. Casanova, Henri. "Distributed computing research issues in grid computing." ACM SIGAct News 33, no. 3 (2002): 50-70.

[11]. Thain, Douglas, Todd Tannenbaum, and Miron Livny. "Distributed computing in practice: the Condor experience." Concurrency and computation: practice and experience 17, no. 2-4 (2005): 323-356.


**Copyright@ REST Publisher** 