JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 19, No. 3, September 2025, pp. 101-107

Machine Learning–Driven Optimization of Graph Processing Performance on HPC Systems Using the Graph500 Benchmark

Hyungwook Shim, Myoungju Koh, Youn Keun Choi, and Minho Suh*
Division of National Supercomputing, Korea Institute of Science and Technology Information, Daejeon, Korea
shw@kisti.re.kr, myju@kisti.re.kr, ykchoi@kisti.re.kr, mhsuh@kisti.re.kr

Abstract
As large-scale artificial intelligence (AI) models such as GPT-4 and Gemini demand increasingly complex computations, optimizing graph processing performance in high-performance computing (HPC) systems has become essential. Unlike traditional benchmarks focusing on numerical operations, graph-based workloads emphasize connectivity and traversal efficiency, which are critical for AI training, data analytics, and large-scale knowledge modeling. This study develops an XGBoost-based performance prediction model using the Graph500 benchmark dataset to identify and optimize the factors affecting GTEPS (giga-traversed edges per second). By analyzing key system variables—such as memory size, problem scale, and node-core allocation—the model predicts graph processing performance with high accuracy (R^2 = 0.96). The results demonstrate that memory capacity and problem scale have the most significant influence, suggesting that balanced resource allocation can yield substantial performance gains without hardware expansion. This research contributes to the field by introducing a machine learning-driven approach for HPC optimization, enhancing both performance prediction accuracy and operational efficiency. The findings provide a practical framework for data-driven HPC resource management in future AI and graph analytics environments.

Category: Cloud computing / High-performance computing

Keywords: Graph500; HPC; AI computing; LLM; XGBoost

I. INTRODUCTION

With the rapid expansion of large-scale artificial intelligence (AI) models such as GPT-4, Gemini, and Llama, the optimization of high-performance computing (HPC) systems that support their training has become a crucial research challenge. These models require not only massive matrix computations but also efficient processing of graph-structured data that capture complex relationships between entities. As AI and data analytics workloads increasingly depend on graph computations, enhancing graph processing performance has emerged as a key determinant of system efficiency.

In this context, the Graph500 benchmark provides an essential measure of graph processing capability through the GTEPS (giga-traversed edges per second) metric. However, improving GTEPS performance is not merely a hardware scaling issue—such as adding more cores or increasing memory—but also an optimization problem that requires a data-driven understanding of how system variables interact to affect performance.

To achieve performance improvement, it is first necessary to predict performance accurately. Performance prediction serves as a prerequisite for performance improvement

Open Access http://dx.doi.org/10.5626/JCSE.2025.19.3.101 http://jcse.kiise.org

Received 28 July 2025; **Revised** 24 October 2025; **Accepted** 4 December 2025
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

because it identifies the dominant factors that determine computational efficiency, allowing resource allocation and configuration strategies to be optimized before actual deployment.

To this end, this study employs a tree-based ensemble learning model, XGBoost, to predict GTEPS performance using the Graph500 dataset. XGBoost was selected for its strong capability to model nonlinear interactions among multiple hardware and workload parameters and for its interpretability through feature importance analysis. The major features used in the model include memory size, problem scale, node-core ratio, computation time, and GTEPS performance per node/core, which collectively represent the structural and computational aspects of HPC systems.

The performance prediction results are not used in isolation; rather, they provide actionable insights for performance enhancement strategies. By quantifying the influence of each factor, the model enables identifying resource configurations that maximize GTEPS without requiring additional hardware expansion. This predictive–prescriptive connection establishes a logical link between performance prediction and performance improvement, ensuring that optimization decisions are guided by empirical evidence derived from the model's analysis.

Accordingly, the objective of this study is to develop an interpretable machine learning model that accurately predicts graph processing performance and to propose an optimization strategy based on these predictive insights. The results contribute to advancing both methodological transparency in HPC performance analysis and practical decision-making in AI-driven computing environments.

II. LITERATURE REVIEW

Research related to the Graph500 benchmark has continuously evolved in response to advances in HPC architectures and analytical methods. Earlier studies mainly focused on benchmarking methodologies, evaluation results, and implementation strategies, while more recent works have explored performance optimization and machine-learning-based prediction approaches.

Bonifati et al. [1] investigated benchmarking methods for graph-processing systems, emphasizing their role in evaluating system performance and identifying optimization opportunities. Gan et al. [2] enhanced graph computation efficiency on the Tianhe supercomputer through buffering and vectorization techniques, and Ren et al. [3] proposed an AI-HPC benchmark integrating automated machine learning. Bai et al. [4] introduced a vector-based path compression algorithm to improve breadth-first search (BFS) performance, and Kolganov [5] compared BFS implementations across diverse architectures, highlighting hybrid traversal approaches for maximizing efficiency.

Beyond benchmark-specific optimization, recent research has increasingly focused on data-driven prediction models to improve HPC performance and resource utilization. For example, Menear et al. [6] developed a methodological framework for "HPC runtime prediction" using data-driven techniques on an 11-million-job dataset from the National Renewable Energy Laboratory's Eagle system. Their study formalized best practices for machine-learning-based runtime estimation and emphasized reproducible methodologies for large-scale HPC workloads. Similarly, Coelho et al. [7] applied regression-based machine learning and decision-tree models to predict execution time and energy consumption in bioinformatics workloads on the Santos Dumont supercomputer. Their results demonstrated that machine learning-based models can accurately forecast execution times and identify key application parameters influencing computational performance.

These recent works illustrate the growing convergence between graph processing benchmarks and machine-learning-based performance prediction frameworks, providing methodological support for this study. Building upon these insights, the present research leverages Graph500 data and a tree-based learning model (XGBoost) to predict and analyze GTEPS performance, thereby contributing to the optimization of HPC system efficiency through interpretable and data-driven methods.

III. OVERVIEW OF GRAPH500

Graph500 is a benchmark designed to evaluate the performance of HPC systems, focusing on measuring the efficiency of graph-based algorithms. Unlike traditional benchmarks such as LINPACK, which primarily assess numerical computation performance, Graph500 evaluates computational performance in solving unstructured problems such as graph traversal and data structure processing. It provides a standard for testing algorithms related to graph exploration and data structure processing, making it particularly significant for applications involving big data and graph theory. Currently, Graph500 has established itself as one of the most important benchmarks for testing graph-based algorithms in HPC environments. Research institutions and corporations worldwide utilize Graph500 to compare supercomputer performance and explore hardware and software optimization techniques for efficient graph processing. Graph500 is widely recognized across various domains, particularly in AI, machine learning, and big data analytics, where demand is high. It plays a crucial role in handling large-scale datasets in applications such as social network analysis, natural language processing, and bioinformatics. Furthermore, Graph500 introduces a new paradigm in HPC system performance evaluation. While traditional benchmarks focus primarily on numerical calculations, Graph500 emphasizes the efficiency of data structures and graph algorithms. This makes it highly practical in fields that require large-scale data processing.

In addition, Graph500 shifts away from conventional numerical performance evaluation, offering performance assessment criteria better suited for modern HPC environments that handle complex data structures and unstructured data processing. As AI and big data analytics continue to gain importance, Graph500 has become a critical benchmark in this domain. With recent technological advancements, AI and machine learning models are processing increasingly large datasets, necessitating efficient graph computations for optimal functionality. Consequently, the significance of Graph500 is expected to grow further. As supercomputer performance continues to improve, Graph500 will evolve as a key component of HPC. In fields requiring large-scale data analysis, Graph500's performance evaluation will remain a crucial benchmark [8–11].

IV. THEORETICAL BACKGROUND OF THE XGBOOST MODEL

XGBoost is a powerful ensemble learning technique based on decision trees. It utilizes a boosting approach, where multiple weak learners are sequentially trained, each improving upon the errors of the previous model to enhance overall predictive performance. The process begins by initializing the prediction values, typically set as the mean of the target variable. The model then computes the residuals, which represent the differences between the predicted and actual values. A new decision tree is trained to minimize these residuals. This iterative process continues, with each tree compensating for the errors of the previous ones. Ultimately, the predictions of all trees are aggregated using weighted summation, progressively refining the model's accuracy. XGBoost introduces several improvements over traditional boosting techniques. It incorporates regularization methods to prevent overfitting and utilizes pruning techniques to remove unnecessary nodes, reducing model complexity. Additionally, it leverages parallel computing to optimize training speed, ensuring high efficiency and predictive performance even with large-scale datasets. The general form of the XGBoost model is expressed as follows: $\hat{y}_i$ represents the predicted value for sample i, K denotes the total number of decision trees, and $f_k(x_i)$ corresponds to the prediction made by the kth decision tree for the input x_i.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \tag{1}$$

The error correction of the existing model can be mathematically expressed as Eq. (2). At the t-th stage, the new tree $f_t(x)$ is added to the previous prediction $\hat{y}_i^{(t-1)}$, gradually improving the model.

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{2}$$

The training of this model is performed in a direction that minimizes the objective function (Eq. 3), which includes the loss function $l(y_i, \hat{y}_i)$ and the regularization term $\Omega(f_k)$.

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k) \tag{3}$$

The minimization of the objective function utilizes a second-order Taylor expansion, leveraging gradient (first derivative) and curvature (second derivative) information to improve the model. The final prediction can be expressed as Eq. (4). In this study, a logarithmic transformation was applied during the data preprocessing stage, and thus, the corresponding mathematical formulation was used [12–16].

$$\hat{y}_i = e^{f(x_i)} - 1 \tag{4}$$

V. ANALYSIS

A. Overview

For optimizing HPC AI performance, the XGBoost model was applied to learn the nonlinear relationships among variables in Graph500. Since the dataset used in this study is structured (non-time series), relatively small in size, and allows for variable importance analysis, the XGBoost model was deemed the most suitable. Python 3.10 was used as the analysis tool, with 80% of the dataset allocated for training and the remaining 20% for testing.

The XGBoost model was implemented using the scikit-learn API of the XGBoost (v2.1.0) library in a Python 3.10 environment. All experiments were conducted on Google Colab, utilizing Intel Xeon CPUs without GPU acceleration. The implementation relied solely on open-source frameworks (XGBoost and scikit-learn) to ensure reproducibility and transparency.

Given that the XGBoost model was employed for optimization purposes, the model was iteratively improved until its explanatory power (R^2 score) and performance (5-fold cross-validation mean RMSE [5-RMSE]) reached high levels. We used the default hyperparameter configuration of XGBoost (learning_rate = 0.3, max_depth = 6, n_estimators = 100, reg_alpha = 0.0, reg_lambda = 1.0, subsample = 1.0), which yielded stable convergence and high predictive accuracy without signs of overfitting. The key evaluation metrics of the model are shown in Table 1. Generally, an R^2 score above 0.9 indicates a very high level of explanatory power, while a 5-RMSE below 10% of the output variable suggests excellent model performance. These criteria were applied throughout the optimization process.

B. Research Data

The research data utilized Graph500_bfs, which was published in November 2024. Among the various data

Table 1. Performance evaluation metrics

Metric	Description
MAE (mean absolute error)	The average of the absolute differences between predicted and actual values. A lower value indicates less error.
MSE (mean squared error)	The average of the squared differences between predicted and actual values. Larger errors are enalized more.
RMSE (root mean squared error)	The square root of MSE. A lower value indicates better model performance.
R^2 score (R-squared, coefficient of determination)	Indicates how well the model explains the data. A value closer to 1 means better predictive power.
5-RMSE (5-fold cross-validation mean RMSE)	The average RMSE after performing 5-fold cross-validation. A lower value indicates better generalization performance.

Table 2. Variables

Code	Variable	Meaning	Unit
A	Memory (TB)	Total memory size	Terabytes
B	Problem scale	Problem size (e.g., graph size)	Integer
C	Nodes per core	Number of nodes allocated per core	Count
D	Memory per node	Memory size per node	Gigabytes
E	C_Time per node	Computation time per node	Seconds
F	C_Time per core	Computation time per core	Seconds
G	GTEPS per core	GTEPS performance per core	Giga-traversed edges per second
H	GTEPS per node	GTEPS performance per node	Giga-traversed edges per second

provided, five variables—nodes, cores, memory, problem scale, and C_Time—were selected under the assumption that they influence GTEPS. The study focused on supercomputers that provided complete datasets for these variables. During the model improvement process, these variables were modified and supplemented to enhance the model's explanatory power. A summary of these variables is presented in Table 2.

Before conducting the analysis, a preliminary characteristic analysis of the data was performed, revealing large variations and high correlations. To address these issues, data preprocessing was carried out. First, since many values were close to zero and exhibited significant variance, a logarithmic transformation was applied to all variables, followed by data scaling (normalization). Additionally, to maximize model performance, new derived variables from C to H were created, and outliers were handled using the interquartile range (IQR) method.

C. Analysis Results

As a result of training the XGBoost model, the function $f(x)$ from the previous section's equation can be expressed as Eq. (5):

$$f(X) = b + w_1 \cdot \log(1 + A) + w_2 \cdot \log(1 + B) + w_3 \cdot \log(1 + C) + \cdots + w_8 \cdot \log(1 + H) \quad (5)$$

where w_n represents the weight indicating the influence of each feature, and b is the model's bias term. The performance evaluation of the model (Table 3) shows that the mean absolute error (MAE) for the test data is 2,465.32, indicating that the model's predictions deviate from the actual values by an average of approximately 2,465.32. The mean squared error (MSE) is 6,321,379.29, which represents the average squared error. The RMSE is 2,514.24, meaning the model has an average prediction error of approximately 2,514.24. Additionally, the R^2 score is 0.96, demonstrating that the model effectively predicts GTEPS values using the input variables. To examine overfitting, a common drawback of XGBoost, the average 5-fold cross-validation RMSE was computed, resulting in 0.84, which is extremely low compared to the average GTEPS value. This suggests that the model maintains

Table 3. Performance evaluation results

Metric	Value
Test data MAE	2,465.32
Test data MSE	6,321,379.29
Test data RMSE	2,514.24
Test data R^2 score	0.96
5-fold cross-validation mean RMSE	0.84

Table 4. Feature importance results

Code	Variable	Weights
A	Memory (TB)	0.28
B	Problem scale	0.21
C	Nodes per core	0.15
D	Memory per node	0.12
E	C_Time per node	0.08
F	C_Time per core	0.07
G	GTEPS per core	0.05
H	GTEPS per node	0.04

stable performance across different data samples.

Several input variables play a crucial role in the GTEPS performance prediction model. Memory (TB) significantly impacts performance, serving as a key factor in large-scale parallel processing. As the problem scale increases, more resources are required, and performance tends to improve linearly. Additionally, a higher nodes per core ratio leads to improved GTEPS performance, while a larger memory per node positively influences computational performance. The C_Time metric, which evaluates performance per node and core relative to execution time, can lead to performance degradation if execution time increases. Finally, GTEPS per core and GTEPS per node reflect existing GTEPS performance and play an essential role in training the prediction model. The feature importance results of the model are presented in Table 4. In the GTEPS performance prediction model, memory (0.28) and problem scale (0.21) are the most influential variables, playing a decisive role in improving performance for large-scale parallel computations. Additionally, nodes per core (0.15) and memory per node (0.12) are key factors in determining performance, depending on the efficient allocation and utilization of computational resources. On the other hand, C_Time per node (0.08) and C_Time per core (0.07) have relatively lower influence, but prolonged execution times can cause performance degradation, necessitating optimization. Lastly, GTEPS per core (0.05) and GTEPS per node (0.04) act as adjustment factors by incorporating existing GTEPS performance data, enabling more precise performance predictions.

VI. CONCLUSION

This study developed an XGBoost machine learning model to optimize the graph processing performance of HPC systems based on the Graph500 benchmark and analyzed the key variables that affect GTEPS performance. The results indicated that memory and problem scale are the most influential factors in enhancing performance. As memory capacity and node count increase, GTEPS tends to rise correspondingly, showing a near-proportional scalability pattern. For instance, a twofold increase in total memory capacity was associated with approximately a 70%–90% improvement in GTEPS, suggesting that sufficient memory resources are essential for large-scale parallel graph processing. Additionally, Nodes per core and memory per node were identified as critical for performance optimization through efficient resource allocation and batching strategies. On the other hand, while C_Time per node and C_Time per core exhibited lower relative influence, excessive execution time can degrade system performance, indicating the need for time-efficient workload scheduling.

By training the performance prediction model with the existing GTEPS per core and GTEPS per node data, this study proposed a framework for more accurate performance forecasting and resource management in HPC systems. Overall, the findings demonstrate that maximizing graph processing performance requires not only hardware expansion but also intelligent scheduling and allocation strategies. Moreover, machine learning-based predictive modeling offers a practical pathway for balancing system scalability and computational efficiency. Although this study is limited to 2024 supercomputer data, continuous monitoring of these quantitative relationships could provide valuable insights into evolving AI and HPC integration trends in the near future.

CONFLICT OF INTEREST

The authors have declared that no competing interests exist.

ACKNOWLEDGMENTS

This research was supported by Korea Institute of Science and Technology Information (KISTI) (No. K25L2M1C1)

REFERENCES

1. A. Bonifati, G. Fletcher, J. Hidders, and A. Iosup, “A survey of benchmarks for graph-processing systems,” in *Graph Data Management*. Cham, Switzerland: Springer, 2018, pp. 163-186. https://doi.org/10.1007/978-3-319-96193-4_6
2. X. Gan, Y. Zhang, R. Wang, T. Li, T. Xiao, R. Zeng, J. Liu, and K. Lu, “TianheGraph: customizing graph search for Graph500 on tianhe supercomputer,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 941-951, 2022. https://doi.org/10.1109/TPDS.2021.3100785
3. Z. Ren, Y. Liu, T. Shi, L. Xie, Y. Zhou, J. Zhai, Y. Zhang, Y. Zhang, and W. Chen, “AIPerf: automated machine learning as an AI-HPC benchmark,” *Big Data Mining and Analytics*, vol. 4, no. 3, pp. 208-220, 2021. https://doi.org/10.26599/BDMA.2021.9020004

4. H. Bai, X. Gan, T. Xu, M. Jia, W. Tan, J. Chen, and Y. Zhang, "VPC: pruning connected components using vector-based path compression for Graph500," *CCF Transactions on High Performance Computing*, vol. 3, no. 3, pp. 271-285, 2021. https://doi.org/10.1007/s42514-021-00070-z
5. A. S. Kolganov, "The fastest and energy-efficient breadth-first search algorithm on a single node with various parallel architectures according to Graph500," *Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta. Seriya" Vychislitelnaya Matematika i Informatika"*, vol. 7, no. 2, pp. 5-21, 2018. https://doi.org/10.14529/cmse180201
6. K. Menear, A. Nag, J. Perr-Sauer, M. Lunacek, K. Potter, and D. Duplyakin, "Mastering HPC runtime prediction: from observing patterns to a methodological approach," in *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, Portland, OR, USA, 2023, pp. 75-85. https://doi.org/10.1145/3569951.3593598
7. M. Coelho, K. Ocana, A. Pereira, A. Porto, D. O. Cardoso, A. Lorenzon, R. Oliveiram P. O. A. Navaux, and C. Osthoff, "Machine learning regression-based prediction for improving performance and energy consumption in HPC platforms," in *High Performance Computing*. Cham, Switzerland: Springer, 2024, pp. 186-200. https://doi.org/10.1007/978-3-031-80084-9_13
8. The Graph500 Benchmark [Online]. Available: https://graph500.org/.
9. T. Suzumura, K. Ueno, H. Sato, K. Fujisawa, adn S. Matsuoka, "Performance characteristics of Graph500 on large-scale distributed environment," in *Proceedings of 2011 IEEE International Symposium on Workload Characterization (IISWC)*, Austin, TX, USA, 2011, pp. 149-158. https://doi.org/10.1109/IISWC.2011.6114175
10. T. Dwyer, A. Fedorova, S. Blagodurov, M. Roth, F. Gaud, adn J. Pei, "A practical method for estimating performance degradation on multicore processors, and its application to HPC workloads," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, UT, USA, 2012, pp. 1-11. https://doi.org/10.1109/SC.2012.11
11. R. M. Karp and G. Bell, "High-performance graphy algorithm and their benchmarking," in *Proceedings of International Conference on High-Performance Computing*, 2013.
12. T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 785-794. https://doi.org/10.1145/2939672.2939785
13. J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189-1232, 2001. https://doi.org/10.1214/aos/1013203451
14. R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197-227, 1990. https://doi.org/10.1007/BF00116037
15. S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.
16. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. New York, NY: Springer, 2009. https://doi.org/10.1007/978-0-387-84858-7

Hyungwook Shim https://orcid.org/0000-0001-8688-0309

Hyungwook Shim received Ph.D. degree from the Seoul National University, Republic of Korea in 2021. He is senior researcher in Korea Institute of Science and Technology Information. His research interests are supercomputer policy and strategy.

Myoungju Koh https://orcid.org/0000-0001-8708-518X

Myoungju Koh received Ph.D. degree in industrial engineering from Sungkyunkwan University, Korea in 2014. From 2009 to 2015, she was with the Science and Technology Policy Institute (STEPI). Since 2015, she has been with the Korea Institute of Science and Technology Information (KISTI). Her research interests include high-performance computing (HPC) policy and systems.

Youn Keun Choi https://orcid.org/0000-0002-5468-3412

Youn Keun Choi received M.Sc. degree in computing from the University of Cardiff, UK in 1998. He joined in the Korea Institute Science Technology and Information (KISTI), in May 2001. His research interests are HPC systems for parallel computing environments and parallel computing education for HPC beginners.

Minho Suh

Minho Suh received Ph.D. degree in chemical engineering from the KAIST, Korea in 2002. He joined in the the Korea Institute Science Technology and Information (KISTI) in July 2004. His research interests are applications and infrastructure services of HPC.