IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 June 2023, accepted 15 July 2023, date of publication 20 July 2023, date of current version 17 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3297200

RESEARCH ARTICLE

Optimizing Numerical Weather Prediction Model Performance Using Machine Learning Techniques

SOOHYUCK CHOI AND EUN-SUNG JUNG, (Senior Member, IEEE)
Department of Software and Communications Engineering, Hongik University, Sejong 30016, Republic of Korea

Corresponding author: Eun-Sung Jung (ejung@hongik.ac.kr)

This work was supported by the Korea Meteorological Administration through the Project ‘‘Development of Application Technology for Climate and Climate Change Monitoring and Prediction Information’’ under Grant KMI-2021-01310.

ABSTRACT Weather forecasting primarily uses numerical weather prediction models that use weather observation data, including temperature and humidity, to predict future weather. The Korea Meteorological Administration (KMA) has adopted the GloSea6 numerical weather prediction model from the UK for weather forecasting. Besides utilizing these models for real-time weather forecasts, supercomputers are essential for running them for research purposes. However, owing to the limited supercomputer resources, many researchers have faced difficulties running the models. To address this issue, the KMA has developed a low-resolution model called Low GloSea6, which can be run on small and medium-sized servers in research institutions, but Low GloSea6 still uses numerous computer resources, especially in the I/O load. As I/O load can cause performance degradation for models with high data I/O, model I/O optimization is essential, but trial-and-error optimization by users is inefficient. Therefore, this study presents a machine learning-based approach to optimize the hardware and software parameters of the Low GloSea6 research environment. The proposed method comprised two steps. First, performance data were collected using profiling tools to obtain hardware platform parameters and Low GloSea6 internal parameters under various settings. Second, a machine learning model was trained using the collected data to determine the optimal hardware platform parameters and Low GloSea6 internal parameters for new research environments. The machine-learning model successfully predicted the optimal parameter combinations in different research environments, exhibiting a high degree of accuracy compared to the actual parameter combinations. In particular, the predicted model execution time based on the parameter combination showed a significant outcome with an error rate of only 16% compared to the actual execution time. Overall, this optimization method holds the potential to improve the performance of other high-performance computing scientific applications.

INDEX TERMS Scientific application, GloSea6, machine learning, I/O optimization, profiling.

I. INTRODUCTION

Significant advancements in computing performance have facilitated the emergence of numerical weather prediction (NWP) [1] models that use large-scale numerical computations for weather forecasting. Since 1999, the Korea Meteorological Administration (KMA) has been using a global data assimilation and prediction system based on the global spectral model, which is based on the global spectrum model from the Japan Meteorological Agency. The KMA introduced the global NWP model GloSea6 [2] from the UK Met Office in 2022 and has since used it for weather forecasting.

GloSea6 comprises two main models: ATMOS and OCEAN. The ATMOS model comprises atmospheric (UM) and land surface (JULES) models, while the OCEAN model comprises ocean (NEMO) and sea ice (CICE) models. Model execution begins after a preprocessing stage, during which the Earth is divided into grids, and initial and auxiliary data called analysis fields are collected for each grid. Subsequently, the analysis fields are used to prepare input fields for the forecast model, after which numerical model calculation begins.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wei.

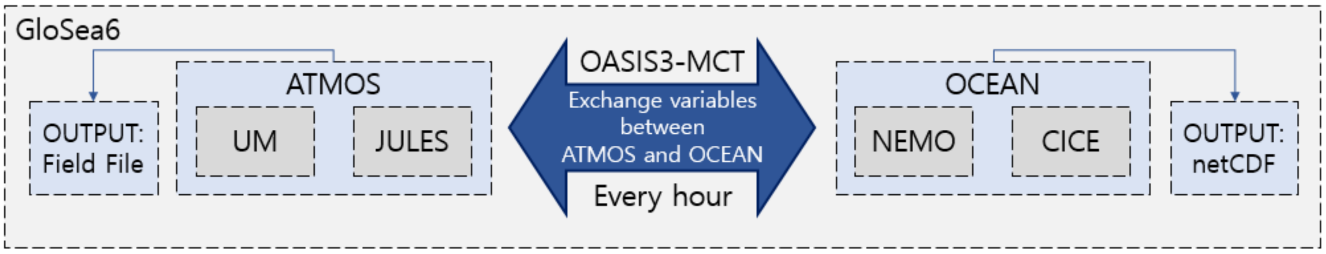


FIGURE 1. **Combined model implementation process.**

Owing to its high demand for computing resources, the KMA provides a low-resolution version of GloSea6 called Low GloSea6 for researchers who lack access to supercomputers. However, even Low GloSea6 requires significant computing resources, and as the model has a high data input/output (I/O) nature, I/O optimization is essential. Notably, general users, who are atmospheric science researchers and not computer scientists, may find conducting performance optimization through trial-and-error inefficient. This paper presents a machine learning-based approach to optimize the hardware and software parameters of the Low GloSea6 research environment.

This study proposes a new cross-inference optimization method for the NWP model Low GloSea6 using machine learning and benchmark tools. Specifically, the following are detailed:

- We defined the entire workflow for performance cross-validation and validated it through experiments.
- Necessary data for cross-inference were categorized into two types: execution hardware platform parameters and internal software parameters of Low GloSea6, and important parameters among them were extracted through model/data validation.
- We used Darshan to collect detailed data on I/O characteristics and verified the final results using runtime data to perform I/O performance cross-validation.
- This study demonstrates the applicability of various machine-learning techniques to explain the complex interactions between the execution hardware platform parameters and the Low GloSea6 internal software parameters, thereby making it feasible to cross-infer performance on a new execution hardware platform.
- The proposed method has been generalized throughout the workflow, demonstrating that it is a general methodology that is not limited to Low GloSea6, which is the subject of this paper.

This paper is structured as follows: Section II describes related research, while Section III provides a detailed description of GloSea6, a numerical model for weather prediction, and the profiling tool used for performance data collection. Section IV explains the hardware/software optimization methodology in the research environment, including the dataset and model used. In Section V, the experiments conducted using the optimization methodology after the model and data verification are described and analyzed. Section VI presents the conclusion and future plans.

II. RELATED RESEARCH

Optimization studies for applications running in real-world or research environments have been conducted in various fields. One such approach is the modification of I/O library codes to achieve I/O optimization of applications. Howison et al. [3] demonstrated performance improvements for high-performance computing (HPC) applications through code modifications and optimizations of HDF5 and MPI-IO libraries, considering the file system characteristics.

Another research method is to achieve I/O optimization by deriving optimal file systems and I/O library parameters. In addition, Behzad et al. [4], [5] used a genetic algorithm to optimize the I/O performance of an application. They created a set of parameters by exploring the file system and I/O library parameter space, measured the I/O performance of the benchmark tool using the parameter set, and iteratively optimized the parameter set based on the measurements until the best I/O performance was achieved. Robert et al. [6] optimized an I/O accelerator using black-box optimization techniques that find input parameters with maximum and minimum performance metrics without considering internal mechanisms. They optimized three input parameters (I/O throughput, I/O latency, and I/O memory usage) of the Atos Flash Accelerator, an I/O accelerator that accelerates I/O operations of various HPC applications using NAND flash memory technology, and used basic metrics, such as I/O operation processing time, as performance indicators. Finally, they validated that the I/O accelerator performance can be improved by applying black-box optimization. Bağbaba et al. [7] implemented an automated tuning solution for the optimal parameters of Lustre parallel file system and MPI-IO ROMIO library, a high-performance implementation of MPI-IO, using I/O monitoring and performance prediction. The solution employed a random forest-based machine-learning algorithm and was validated using two benchmarking tools (IOR-IO and MPI-Tile-IO) and a molecular dynamics model (ls1 Mardyn.”).

Our research differs from previous studies in two ways. First, our study enables easy optimization, even without prior I/O optimization knowledge. While Howison et al. [3] achieved I/O performance optimization by modifying the I/O library code, this approach requires a developer's expertise and is not easily accessible to general users. In contrast, our research focuses on machine learning-based performance optimization that is easily modifiable and accessible by considering the hardware and software parameters of the research environment. Second, our study simultaneously considers hardware platform parameters and internal software parameters. Behzad et al. [4], [5] optimized I/O using adjustable parameters in the parallel I/O stack, specifically related to file systems, HDF5, and MPI-IO libraries. However, the research did not consider benchmark tool parameter optimization. Robert et al. [6] used the parameters of I/O throughput, I/O latency, and I/O memory usage of the Atos Flash Accelerator I/O accelerator for its optimization. These parameters are internal software parameters mentioned in this paper, and hardware platform parameters were not considered.

Our research has broad applicability. Bağbaba et al. [7] study focused on the MPI-IO ROMIO library and Lustre parallel file system in a single research environment, which limits its generalizability. In contrast, we collected data in two different research environments and conducted validation on different hardware platform environments using Low GloSea6. In addition, we used MPICH, an MPI-IO implementation with high accessibility that can be applied regardless of the specific implementation version of MPICH. To verify this, we conducted experiments in research environments using different versions of MPICH.

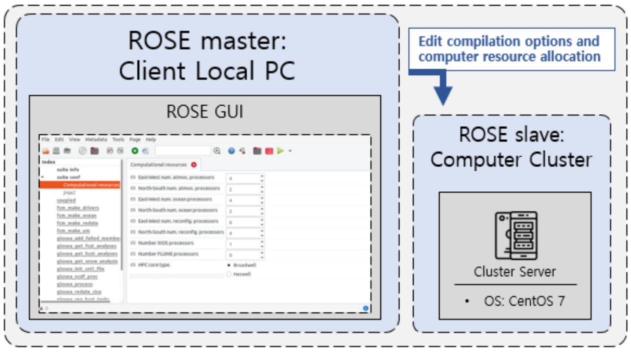


FIGURE 2. **ROSE graphical user interface. (Used image sources [11].)**

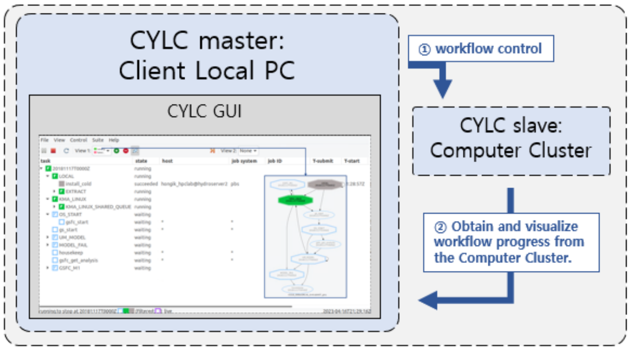


FIGURE 3. **CYLC graphical user interface.**

III. BACKGROUND

A. NUMERICAL WEATHER FORECASTING: GLOSEA

The ATMOS and OCEAN models are coupled using the Ocean Atmosphere Sea Ice Soil 3-Model Coupling Toolkit (OASIS3-MCT) Coupler [8] to share their results. Figure 1 illustrates the collaboration structure of the two models. The Gloseа6 model performs simulations in 15-day increments, saving the results to a file and proceeding to the next step. The saved file is used as the initial data for the simulation in the following step. Therefore, the numerical model calculations restart at each step. As shown in Fig. 1, the ATMOS model produces Fieldsfile (ff) [9] data, while the OCEAN model produces data in NetCDF (nc) [10] file format. Forecast (FCST) and Hindcast (HCST) data are generated to predict future and reproduce past situations, respectively. Ensemble probability predictions are generated by comparing the FCST and HCST data using the average of multiple models. The generated ensemble prediction data is verified by comparing it with observation data using various verification metrics. Deterministic verification techniques such as Bias, RMSE, and Correlation are used for data comparison, and probabilistic verification techniques such as Brier Skill Score and Reliability Diagram are used.

The operating system of GloSea6 is based on special software implemented with Jinja2 called ROSE and CYLC. ROSE is a tool used to easily create, edit, and execute suites, which are units that manage one or more consecutive tasks or processes. ROSE can set compile options and computer resources, etc., through the suite configuration files "rose-app.conf" and "rose-suite.conf." Figure 2 shows the configuration screen using the ROSE graphical user interface, which allows for easy configuration. CYLC is a workflow engine used to execute suites. The task sequence is set through the "suite.rc" file, and as shown in Fig. 3, workflow visualization and control are possible.

Table 1 shows the sequence and content of GloSea6's suite operations, where "gsfc" represents the process of producing Forecast data and "gshc" represents the process of producing Hindcast data. Relevant source codes are obtained through FCM, a scientific application wrapper for SubVersion (SVN), during the 'Compilation step'.

Complex models like GloSea6 require the use of supercomputers not only for actual weather forecasting but also for research purposes. However, considering the limited resources of supercomputers, many researchers have faced challenges in running the model. To address this issue, the KMA developed a low-resolution model called Low GloSea6, which can be run on small- to medium-sized servers in research institutions.

Low GloSea6 is a low-resolution coupled model similar to GloSea6 but with a grid size of 60 km extended up to 170 km. Unlike GloSea6, which runs both ROSE and CYLC on a single supercomputer platform, Low GloSea6 uses multiple platforms such as client local PC and computer cluster, as shown in Fig. 4. The suite's working environment and operational settings are configured through ROSE/CYLC,

TABLE 1. Glosea6/low glosea6 suite work order.

Steps	Tasks	Descriptions
Compilation	Install_cold	Install utilities and configuration of ancillaries
	fcm_make_drivers, fcm_make2_drivers	Install python scripts
	fcm_make_ocean, fcm_make2_ocean	Install ocean/sea-ice model
	fcm_make_um	Install atmospheric model
Model run	gsfc[/gshc]_get_analysis	Get atmospheric/ocean/sea-ice initial data (only GloSea6)
	gsfc[/gshc]_redate_cice	Correct meta data of CICE dump files
	gsfc[/gshc]_recon	Reconfigure atmospheric initial data
	gsfc[/gshc]_model_m1_s01	Run coupled model for member 1 during step 01
	gsfc[/gshc]_post_model_m1_s01	Move/remove output files and dump files
	gsfc[/gshc]_ncdf_proc_m1_s01	Merge ocean output files
	gsfc[/gshc]_process_m1_s01	Convert format of output files

'gsfc' means GloSea6 forcast data production process, and 'gshc' means GloSea6 hindcast data production process.

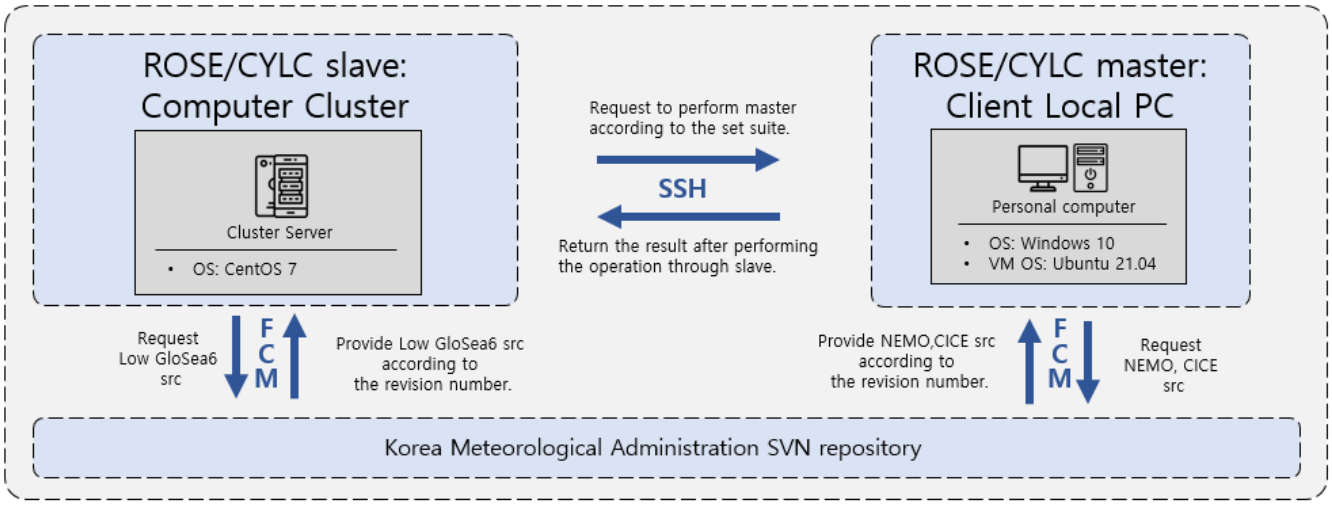

ROSE/CYLC slave: Computer Cluster
Cluster Server
OS: CentOS 7
Request to perform master according to the set suite.
SSH
Return the result after performing the operation through slave.
ROSE/CYLC master: Client Local PC
Personal computer
OS: Windows 10
VM OS: Ubuntu 21.04
Request Low GloSea6 src
FCM
Provide Low GloSea6 src according to the revision number.
Provide NEMO,CICE src according to the revision number.
FCM
Request NEMO, CICE src
Korea Meteorological Administration SVN repository

FIGURE 4. The overall composition of low GloSea6. (Used image sources [11].)

and the suite is then transmitted to the computer cluster using secure shell (SSH).

The computer cluster refers to the assigned values in ROSE to proceed with compilation and resource allocation of the received suite, and performs all tasks sequentially or in parallel through CYLC. In addition, the order and content of the suite in Low GloSea6 differ from those in GloSea6. While GloSea6 performs the "Model run step" after the "Compilation step," according to Table 1, Low GloSea6 does not perform "gshc" and "gsfc_redate_cice" tasks because the initial fields are already set.

B. PROFILING TOOL

The ATMOS and OCEAN models of Low GloSea6, which are the performance measurement targets, are HPC applications based on the message passing interface (MPI). We used Darshan [12], [13], an HPC I/O profiling tool that can measure and analyze the performance of the MPI I/O and POSIX I/O of HPC applications.

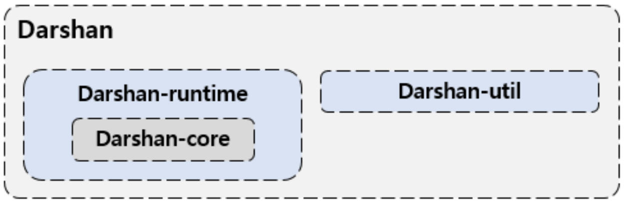


FIGURE 5. Darshan source tree.

Darshan is an open-source I/O profiling tool used to understand the I/O characteristics of HPC applications. As shown in Fig. 5, Darshan comprises two components: Darshan-runtime, which generates I/O activity logs for HPC applications, and Darshan-util, which analyzes log contents.

Figure 6 shows a summary of Darshan's operation. To explain the operation of Darshan using MPI-IO as an example, existing MPI API calls are intercepted and replaced with equivalent calls implemented by Darshan using the dynamic library preload mechanism provided by Linux.

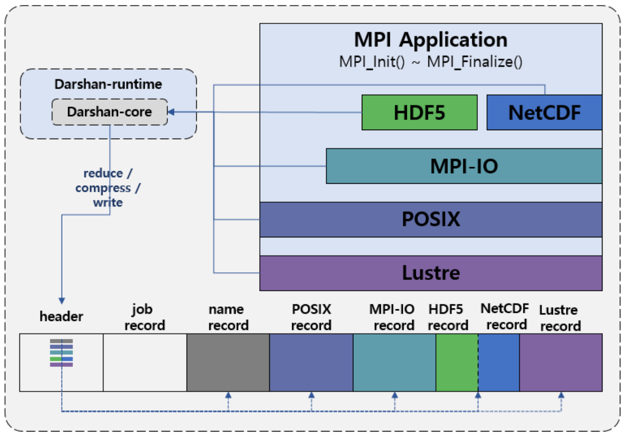
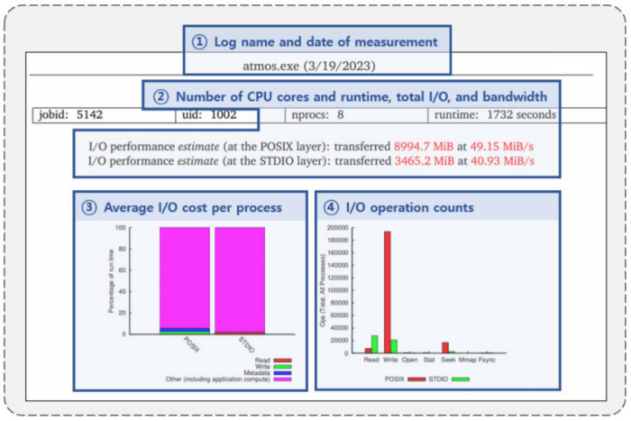


FIGURE 6. Darshan-core process.


① Log name and date of measurement
② Number of CPU cores and runtime, total I/O, and bandwidth
③ Average I/O cost per process
④ I/O operation counts
Percentage of run time
POSIX
STDIO
Read
Write
Metadata
Other (including application compute)
Ops (Total, All Processes)
Read Write Open Stat Seek Mmap Fsync
POSIX
STDIO

atmos.exe (3/19/2023)

jobid: 5142 | uid: 1002 | nprocs: 8 | runtime: 1732 seconds

I/O performance *estimate* (at the POSIX layer): transferred 8994.7 MiB at 49.15 MiB/s
I/O performance *estimate* (at the STDIO layer): transferred 3465.2 MiB at 40.93 MiB/s

FIGURE 7. Darshan PDF log.

When Darshan intercepts the MPI library start function, "MPI_Init()," it initializes itself and stores statistics to a file in the first come first served (FCFS) order while the MPI application runs. When the MPI application calls "MPI_Finalize()" to terminate the MPI library environment, Darshan intercepts it and records the saved I/O log of the Darshan core library into a consolidated log file.

Figure 7 shows the partial Darshan log analysis files for the ATMOS and OCEAN models.

For item 1, the filename of the profiled HPC application and the measurement date are recorded. When ATMOS and OCEAN models are executed simultaneously, the measurement is performed concurrently, but the filename is written based on the application name that appears first in the execution command, which is "atmos.exe" in this case. For item 2, "nprocs" represents the number of processes used in the application, and runtime indicates the execution time. The ATMOS and OCEAN models recorded in the PDF file were assigned four processes each, and nprocs is recorded as 8, which is the sum of the process allocation values for both models. The model ran for 1732 seconds, and the amount of data transferred (in MiB) and bandwidth (in MiB/s) through MPI-IO, POSIX, and STDIO I/O functions can be checked in the 'I/O performance estimate. Notably, even though the HPC application is based on MPI, MPI-IO is not observed. For item 3, the X-axis represents the libraries used, and the Y-axis represents the percentage of time spent on reading, writing, metadata I/O, and computation. Most time is spent on computation, and I/O can be seen at the bottom of each chart. The X-axis of item 4 represents specific I/O operations, and the Y-axis shows the number of operations. Notably, POSIX (in red) dominates the I/O workload. We aim to obtain the optimization benefits and I/O performance metrics of the ATMOS and OCEAN models through Darshan logs.

We reviewed the values from the Darshan results, which can be used as performance metrics. The fourth graph indicates no MPI I/O operations during the model execution but shows over 180,000 read operations in POSIX, while the third graph reveals that I/O operations account for approximately 6% of the runtime. This study employed "I/O performance estimate" metrics such as POSIX bandwidth and STDIO bandwidth as I/O performance optimization indicators and used runtime as a metric for overall performance optimization, not just for I/O.

IV. PROPOSED OPTIMIZATION METHOD

To optimize I/O in HPC scientific applications, two approaches can generally be used.

The first involves direct modification of the program's implementation method, while the second focuses on identifying performance-boosting parameters by changing the hardware platform parameters and software internal parameters of the HPC scientific application. Both methods have limitations. The first method of directly modifying the implementation of HPC scientific applications may not be feasible for many users who are not developers. The second method involves finding optimal performance parameters by changing the hardware platform and internal software parameters of HPC scientific applications, which may not always be possible due to hardware and software constraints. This may be impractical if the number of parameters is large or if it takes a long time to complete multiple runs of the program to find the optimal parameters.

In this paper, we propose a new cross-inference optimization method that considers both the hardware platform and internal parameters of the application program using machine learning and benchmark tools to improve the performance of Low GloSea6. Further details on this approach will be discussed in the following section.

A. HARDWARE/SOFTWARE PARAMETER OPTIMIZATION

The proposed method comprises four steps (Fig. 8). First, we used Darshan to collect performance data of Low GloSea6 and the benchmark tool based on the internal parameters and hardware platform parameter settings. Second, we used machine-learning techniques to find the benchmark parameter set B with the closest relationship between the parameters of Low GloSea6 and the benchmark tools. Third, we used a machine-learning model trained on the relationship between the benchmark parameter set B/hardware platform parameter

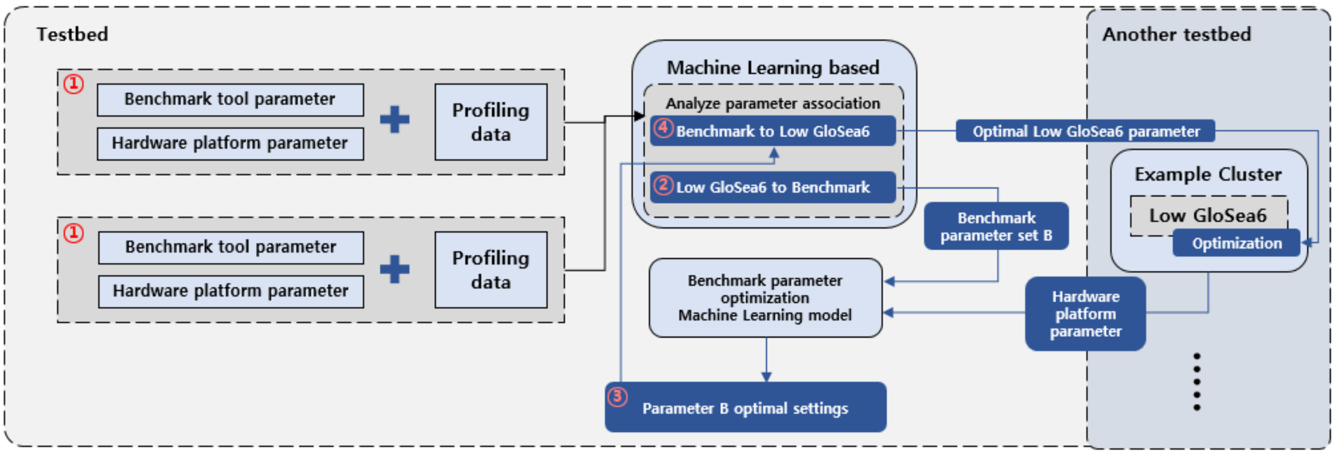

Testbed
① Benchmark tool parameter
Hardware platform parameter
Profiling data
① Benchmark tool parameter
Hardware platform parameter
Profiling data
Machine Learning based
Analyze parameter association
④ Benchmark to Low GloSea6
② Low GloSea6 to Benchmark
Benchmark parameter set B
Benchmark parameter optimization Machine Learning model
③ Parameter B optimal settings
Optimal Low GloSea6 parameter
Another testbed
Example Cluster
Low GloSea6
Optimization
Hardware platform parameter

FIGURE 8. Schematic for future research: Optimizing parameters using benchmarking tools.

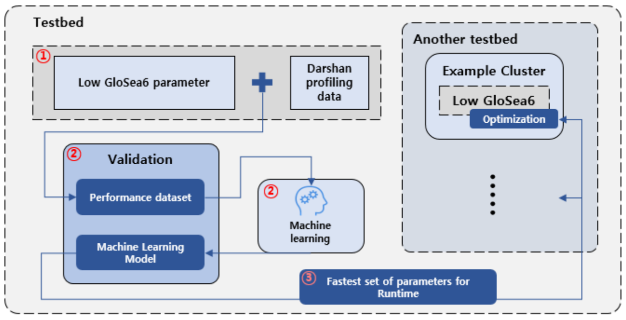


FIGURE 9. Schematic methods for optimizing software parameters. (Used image sources [11].)

settings and performance to determine the optimal settings of the benchmark parameter set B using the hardware platform parameters of the production or research environment. Fourth, we used machine-learning techniques to convert the optimal settings of benchmark parameter set B into the optimal internal parameters of Low GloSea6.

However, this study employed a simple method for performance optimization in the 2nd and 3rd steps without using a benchmark tool. Further exploration of utilizing all steps is reserved for future research. Benchmark tools are typically used to extract hardware information for a specific platform quickly, particularly for HPC applications such as GloSea6, which require many computing resources and have long execution times. However, because performance prediction accuracy can decrease, this study proposes a 3-step approach.

Figure 9 shows the new hardware/software parameter optimization process, which comprises three steps. The first step involves collecting performance data using Darshan for different hardware platforms and Low GloSea6 internal parameter settings. The second step involves training and validating a machine-learning model using the collected performance data. The third step involves identifying optimal hardware platform parameters and corresponding Low GloSea6 internal parameters for a new research environment.

TABLE 2. Low GloSea6 parameter.

Names	Descriptions
ATMOS_NPROCX	Number of cores assigned to ATMOS X-axis direction
ATMOS_NPROCY	Number of cores assigned to ATMOS Y-axis direction
NEMO_NPROCX	Number of cores assigned to OCEAN X-axis direction
NEMO_NPROCY	Number of cores assigned to OCEAN Y-axis direction
XIOS_NPROC	Number of cores assigned to XIOS
GSFC_PP_REINIT_DAYS	PP file production cycle
GSFC_RESUB_DAYS	Dump file production cycle

This study utilized performance data that include the internal parameters of Low GloSea6 and hardware platform parameters, as well as I/O performance estimates and runtime as performance metrics for both parameter sets. Table 2 summarizes the internal parameters of Low GloSea6.

The “ATMOS_NPROCX” and “TMOS_NPROCY” represent the number of CPU cores allocated to the X and Y axes of the ATMOS model, respectively, while “NEMO_NPROCX” and “NEMO_NPROCY” represent the number of CPU cores allocated to the X and Y axes of the OCEAN model. Low GloSea6 assigns CPU cores to each grid point of the Earth, divided into “X x Y” grid units, as global numerical models like Low GloSea6 are designed to perform calculations for predictions at each grid point (Fig. 10). “XIOS_NPROC” represents the number

TABLE 3. Low GloSea6 performance data.

Types	Names		Cautions
Low GloSea6 parameter	Days	GSFC_PP_REINIT_DAYS	Low GloSea6 does not support restarting, so it allocates the desired forecast date.
		GSFC_RESUB_DAYS	
	ATMOS_NPROCX, NEMO_NPROCX		Depending on the characteristics of the model, set to even, and must be greater than Y-axis.
	ATMOS_NPROCY, NEMO_NPROCY		Depending on the characteristics of the model, it is set to an even number and must be less than X-axis.
	XIOS_NPROC		Assign according to the CPU core margin of the server. When set to 0, it operates inside the OCEAN model.
Hardware platform parameter	Node		The total number of nodes in the hardware platform.
	Block size		In bytes, with the same value set for wsize and rsize configured via NFS.
	Switch data transfer speed		The network switch speed is in bps (bits per second).
	Disk IO speed	Disk IO speed_write	The read/write speed of the hardware platform memory in MB/s (megabytes per second).
		Disk IO speed_read	
Performance Indicator	Runtime		The Runtime of the ATMOS and OCEAN models was measured by Darshan, in seconds.
	I/O performance estimate	POSIX bandwidth	The bandwidth of POSIX and STDIO was measured by Darshan, in MiB/s.
		STDIO bandwidth	

of CPU cores assigned to the XML IO SERVER (XIOS) created to manage the NetCDF output of the OCEAN model. If "XIOS_NPROC" is set to 0, XIOS operates within the OCEAN model without additional CPU core allocation. The CPU cores used in Low GloSea6 must be less than or equal to the CPU cores on the hardware platform, and this can be calculated as in Equation (1).

$$\begin{aligned}\text{CPUcore} = {} & \text{ATMOS\_NPROCX} \times \text{ATMOS\_NPROCY} \\ & + \text{NEMO\_NPROCX} \times \text{NEMO\_NPROCY} \\ & + \text{XIOS\_NPROC} \end{aligned} \tag{1}$$

"GSFC_PP_REINIT_DAYS" and "GSFC_RESUB_DAYS" are dump files for restart, which means the number of days for the production cycle of weather forecasting data, and these two parameters should be set to the same number of days.

Owing to diverse hardware platform parameters, considering all of them is challenging. Therefore, we selected parameters that are closely related to computing and I/O performance. This study focused on the hardware platform parameters of node number, file system block size, switch network speed, and disk I/O speed, which have been considered in several studies [14], [15], aimed at improving system software to enhance I/O performance in HPC environments. These parameters were named "Node" and "Block size," "Switch data transfer speed," and "Disk IO speed."

Table 3 summarizes the descriptions and limitations for all parameters and I/O performance metrics. First, we describe the internal parameters of Low GloSea6. The global data

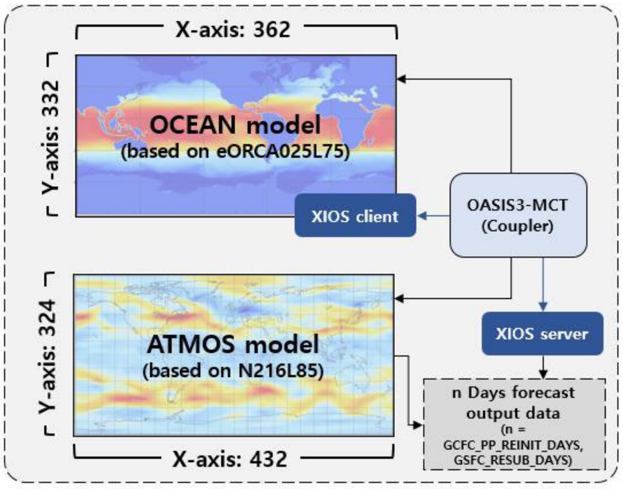


FIGURE 10. Relationship between low GloSea6 and internal para-meters.

used in Low GloSea6 are divided into even grids (Fig. 10), and to efficiently use CPU cores without leaving any idle, the number of CPU cores on the X and Y axes must be divided into even numbers. As the initial field has a larger X-axis than the Y-axis, more CPU cores should be used for X-axis calculations of the ATMOS and OCEAN models. If no spare CPU core exists on the hardware platform, XIOS operates inside the OCEAN model by allocating 0. Conversely, if a surplus exists, it allocates to the ATMOS and OCEAN models and assigns the remaining

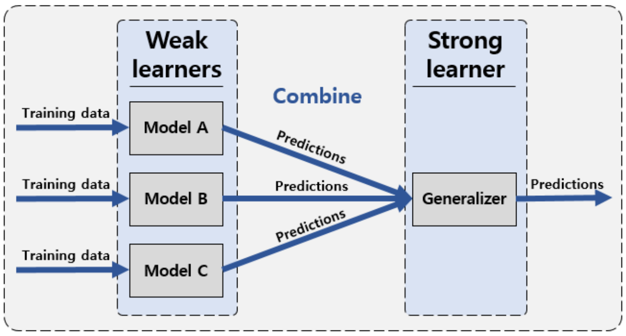


FIGURE 11. Ensemble.

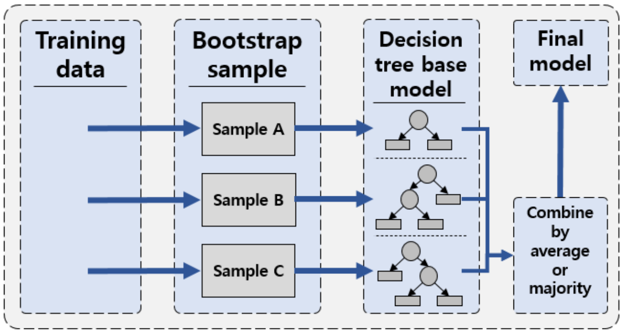


FIGURE 12. Bagging.

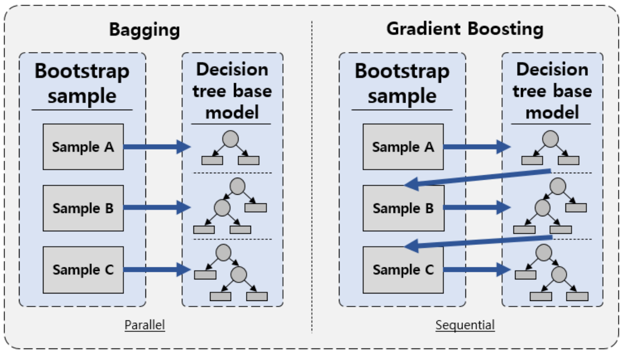


FIGURE 13. Bagging vs gradient boosting.

CPU cores to XIOS. Given that Low GloSea6 currently does not support model restart, "GSFC_PP_REINIT_DAYS" and "GSFC_RESUB_DAYS" produce prediction data once according to the set cycle and then terminate. Therefore, the two parameters in Low GloSea6 are utilized to set the desired weather forecast days, and in this study, we will refer to these two parameters collectively as "Days."

The hardware platform parameter "Node" refers to the total number of nodes in the hardware platform. "Block size" is the "wsize/rsize" value set through the network file system (NFS), and "Switch data transfer speed" refers to the network switch speed. "Disk IO speed_write" and "Disk IO speed_read" are the write and read speeds of the hardware platform memory, respectively.

This study used several I/O performance metrics, including POSIX Bandwidth (MiB/s) and STDIO Bandwidth (MiB/s) measured with Darshan, as well as runtime (sec). The performance data comprised three datasets, each containing 10 parameters, based on three I/O performance metrics. Each dataset contained 549 data points.

The software/hardware parameters in Table 3 are adjusted to collect performance data on the testbed, which is used to train a machine-learning model. Subsequently, the trained model can predict software/hardware parameter configurations outside the collected performance data.

B. MACHINE LEARNING MODEL

In this section, we describe the machine-learning techniques and models to be used for optimization. We aim to predict the I/O performance metrics of Low GloSea6 operating in a production or research environment using the collected performance data for I/O optimization. This study utilized the R package [16] to build multiple linear regression (MLR) models, as well as decision tree-based random forest and gradient boosting models.

MLR is a method for predicting the dependent variable through independent variables, assuming a linear relationship between them. Random forest and gradient boosting are ensemble models based on decision trees. The ensemble is a technique used to compensate for the instability of decision trees by combining weak models to create a strong model (Fig. 11).

Bagging is a model that uses bootstrap samples of the data (Fig. 12) to create weak models, combines them using the average of the predicted values, and performs the final prediction. Random forest is similar to bagging in that it uses bootstrap samples but randomly selects split variables during the formation of weak models.

The gradient boosting model combines weak models into strong models using weights and adds a sequential characteristic to the traditional bagging method. As shown in Fig. 13, the first model makes a prediction, and based on that prediction, weights are assigned to the data, which then influence the next model.

The hyperparameter settings for each model used in this study are as follows: The MLR model lacks hyperparameters, as it is a characteristic of linear regression estimation. The random forest model has a hyperparameter called "mtry," which determines the number of features used for each tree. Following Genuer et al. [17], setting the "mtry" hyperparameter to the value of "the number of independent variables divided by 3" is expected to result in superior performance regarding RMSE for low-dimensional regression prediction. Therefore, we set the "mtry" value to 3. This value was obtained by dividing the number of the independent variables we used, which is 10, by 3 and rounding the first decimal place. The "interaction.depth" of the gradient boosting model is a hyperparameter that controls the depth of each tree. According to Hastie et al. [18], values of "interaction.depth" exceeding 3 are not recommended due to the risk of overfitting. However, if complex relationship modeling is needed, 6 is expected to yield better results. Hence, we tested

TABLE 4. Computer cluster and client local PC SPEC.

	Cluster1		Cluster2 (NFS Client/Server)	Client Local PC
	NFS Client	NFS Server		
OS	CentOS Linux release 7.9.2009	CentOS Linux release 7.9.2009	CentOS Linux release 7.9.2009	Windows 10 pro 64bit (10.0, build 19044)
Virtual OS	-	-	-	Ubuntu 21.04
CPU	AMD EPYC 7302 16-Core Processor (16 Core 32 Thread) * 2	Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz * 2	AMD Threadripper PRO 3955WX (16cores, 3.9GHz), Liquid-cooled * 1	Intel® Core™ i9-10900X CPU @ 3.70 GHz
Storage	4 TB Crucial MX500 SSD	4 TB HGST Deskstar HDD	4 TB Samsung 870 SSD	256GB SK Hynix NVMe SSD
RAM	132 GB	256 GB	256 GB	32 GB
MPICH	3.1.4	3.1.4	4.0.3	3.4.1

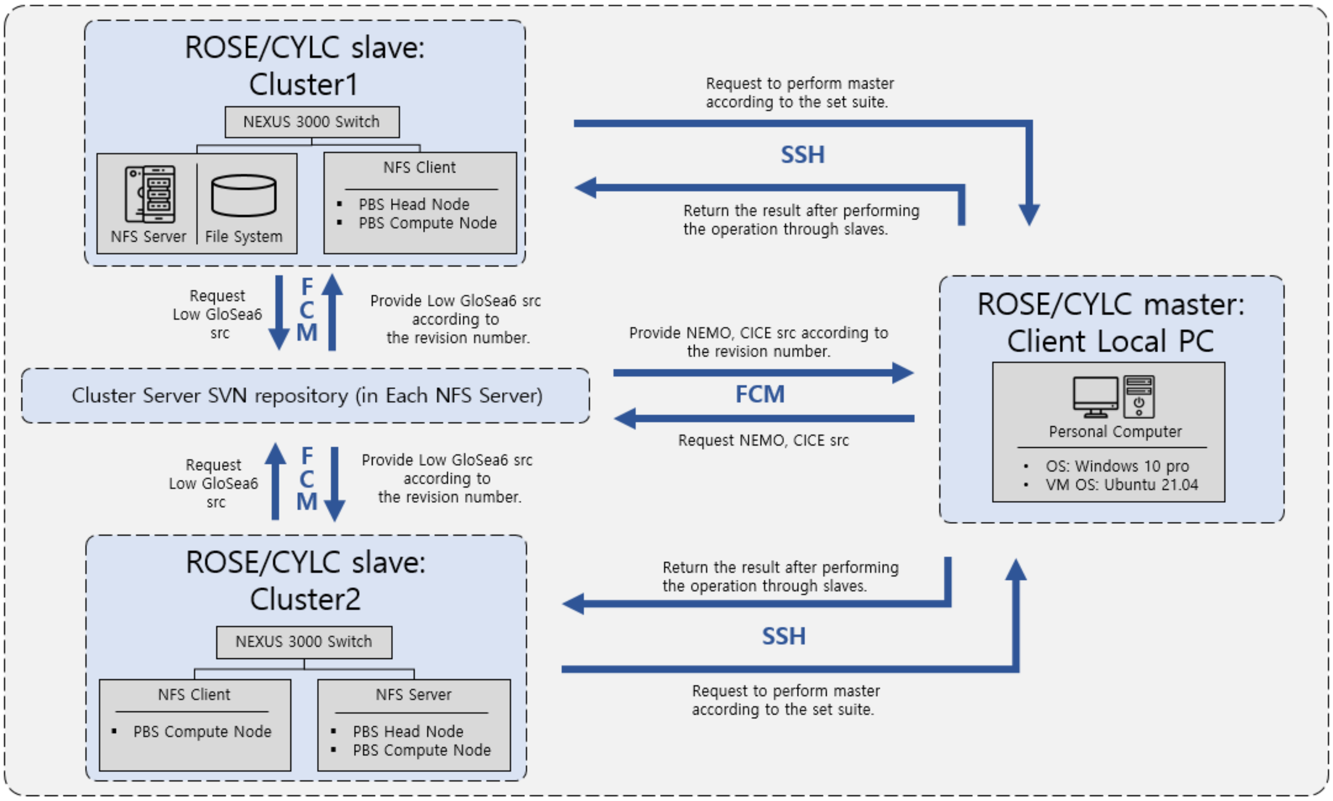

ROSE/CYLC slave: Cluster1
NEXUS 3000 Switch
NFS Server
File System
NFS Client
PBS Head Node
PBS Compute Node
Request to perform master according to the set suite.
SSH
Return the result after performing the operation through slaves.
Request Low GloSea6 src
FCM
Provide Low GloSea6 src according to the revision number.
Provide NEMO, CICE src according to the revision number.
ROSE/CYLC master: Client Local PC
Personal Computer
OS: Windows 10 pro
VM OS: Ubuntu 21.04
Cluster Server SVN repository (in Each NFS Server)
FCM
Request NEMO, CICE src
Request Low GloSea6 src
FCM
Provide Low GloSea6 src according to the revision number.
ROSE/CYLC slave: Cluster2
NEXUS 3000 Switch
NFS Client
PBS Compute Node
NFS Server
PBS Head Node
PBS Compute Node
Return the result after performing the operation through slaves.
SSH
Request to perform master according to the set suite.

FIGURE 14. The overall composition of the low GloSea6 testbed. (Used image sources [11].)

the range of 1, 2, 3, and 6 for "interaction.depth." Default values [19], [20] were used for the random forest and gradient boosting models for hyperparameters that were not mentioned.

V. EXPERIMENTAL EVALUATION

Setting up a client local PC for the ROSE/CYLC primary role and computer cluster system for the ROSE/CYLC secondary role is necessary for the community version of Low GloSea6, which is not for supercomputers. We designated Clusters 1, 2, and 3 for the experimental computer cluster. Clusters 1 and 2 were clusters installed at Hongik University, while Cluster 3 was installed at Changwon National University. Table 4 lists the detailed hardware specifications of Hongik University. Fig. 14 shows the configured experimental environment, and Cluster 3 will be introduced in detail in the next section. We established SVN within Clusters 1 and 2 to minimize the time required in the compilation phase.

The experimental plan involved collecting performance data by varying the hardware platform parameters of Clusters 1 and 2 and the internal parameters of Low

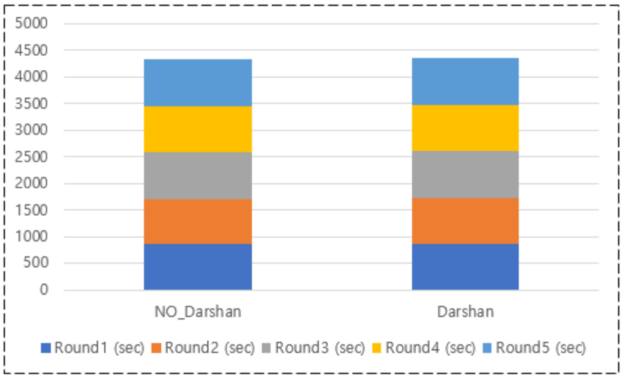


FIGURE 15. 5-round measurement cumulative bar graph before and after darshan application.

TABLE 5. 5-round low GloSea6 runtime measurements and average values.

Rounds	Runtime (sec)	
	NO_Darshan	Darshan
Round1	871	859
Round2	843	868
Round3	873	878
Round4	876	878
Round5	873	882
Average	867.2	871

GloSea6. Subsequently, the collected performance data and machine-learning techniques were utilized to predict the optimal internal parameter values of Low GloSea6 for the hardware platform parameters of Cluster 3. Finally, the predicted internal parameter values were applied to Low GloSea6 in Cluster 3 to complete the optimization and analyze the results.

With this rough outline of the overall experimental plan, we will now describe the experiment's details.

A. EXPERIMENTAL ENVIRONMENT VALIDATION

First, we verified the reliability of the experiment by measuring the overhead of the profiling tool (Darshan) to ensure that it did not significantly affect the experiment. We ran five Low GloSea6 instances with the same parameters before and after applying Darshan and measured the runtime of the "gsfc_model_m1_s01" step, where the ATMOS and OCEAN models are executed. Figure 15 shows the cumulative bar graph for the five measurements, and Table 5 presents the individual runtimes and their averages. While a significant difference in the cumulative bar graph was unobserved, an overhead of approximately four seconds was observed in the runtime average. Considering that the average runtime was approximately 870 seconds, we confirmed that Darshan's overhead was negligible.

B. OPTIMIZING GloSea6/HARDWARE PLATFORM PARAMETERS

The proposed research environment hardware/Low GloSea6 optimization process in this study comprises three steps, which are explained below.

1) STEP1: DATA COLLECTION

Run the model by changing the hardware platform parameters and Low GloSea6 internal parameters on Clusters 1 and 2 and collect performance data using Darshan.

Table 6 summarizes all the parameter settings. As the runtime and I/O amount are linearly proportional to the prediction criterion "Days," we fixed the prediction criterion to 1 day for quick data collection. With 32 cores in the experimental testbed, the X-axis CPU core for ATMOS and OCEAN models can be 2, 4, or 6, while the Y-axis can be 2 or 4, so we varied the number of cores within these ranges for performance data collection. XIOS_NPROC was set from 0, 1, 2, 3, and 4, considering CPU core resource usage during model execution.

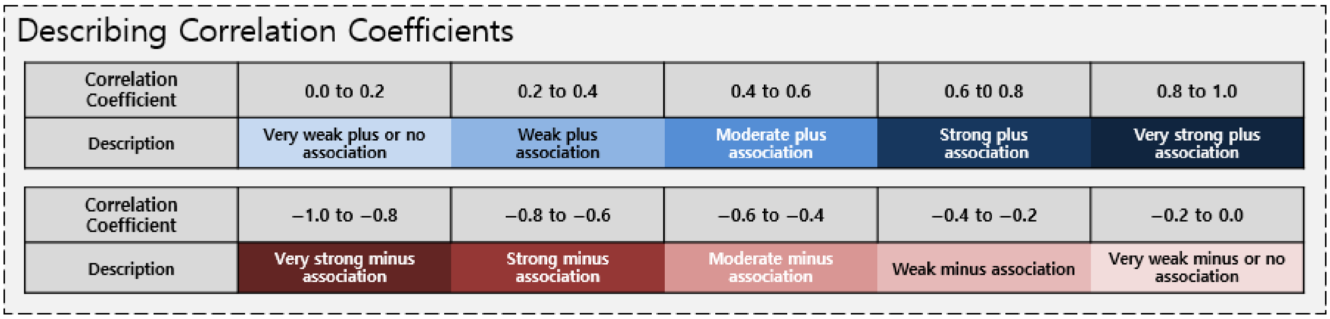
The hardware platform parameter "Node" was configured with a maximum of 2, and the "Block size" was configured with values of 32768 and 65536 bytes, which are typical settings for NFS, as well as the default value of 524288 bytes in our experimental environment. The "Switch data transfer speed" has settings of 100 Mbps, 1 Gbps, and 10 Gbps due to the configuration limit of the Cisco Nexus 3000 series switch used in our experiment. The "Disk IO speed" was configured using Linux control groups (Cgroups) based on TEKIE's September 2022 survey [21]. The parameter values are the average read/write speed of Data Description (dd) commands measured 100 times on each of the Clusters 1 and 2 hardware platforms, as well as the average read/write speed of TEKIE's SSD and HDD. The write speed of the SSD was set to the speed in the experimental environment because it exceeded the average speed in our research environment.

2) STEP2: VALIDATION AND MODEL LEARNING

The collected performance data were then subjected to validation, followed by model training and validation.

A primary concern during data validation is multicollinearity, which is a common problem that arises due to a high correlation between independent variables (parameters), leading to distorted analysis results. Referring to Fig. 16, summarizing previous research by Rea and Parker [22], a correlation coefficient of 0.6 or higher indicates a correlation between each parameter. Therefore, we selected four parameters suspected of multicollinearity from the performance data. However, a correlation does not necessarily imply multicollinearity, as it simply means that each parameter tends to move together.

Therefore, we calculated the variance inflation factor (VIF) among four parameters. VIF measures the extent to which the variance or standard error of the estimated



Describing Correlation Coefficients

Correlation Coefficient	0.0 to 0.2	0.2 to 0.4	0.4 to 0.6	0.6 t0 0.8	0.8 to 1.0
Description	Very weak plus or no association	Weak plus association	Moderate plus association	Strong plus association	Very strong plus association

Correlation Coefficient	−1.0 to −0.8	−0.8 to −0.6	−0.6 to −0.4	−0.4 to −0.2	−0.2 to 0.0
Description	Very strong minus association	Strong minus association	Moderate minus association	Weak minus association	Very weak minus or no association

FIGURE 16. Describing correlation coefficients.

TABLE 6. Low GloSea6 performance data.

Types	Names		Set values	
Low GloSea6 parameter	Days		1	
	ATMOS_NPROCX, NEMO_NPROCX		2, 4, 6	
	ATMOS_NPROCY, NEMO_NPROCY		2, 4	
	XIOS_NPROC		0, 1, 2, 3, 4	
Hardware platform parameter	Node		1, 2	
	Block size		32768 byte, 65536 byte, 524288 byte	
	Switch data transfer speed		100 Mbps, 1 Gbps, 10 Gbps	
	Disk IO speed_write/read	Cluster 1	Write: 13.2 MB/s	Read: 699 MB/s
		Cluster 2	Write: 444.5 MB/s	Read: 541 MB/s
		SSD	Write: 444.5 MB/s	Read: 200 MB/s
		HDD	Write: 160 MB/s	Read: 80 MB/s

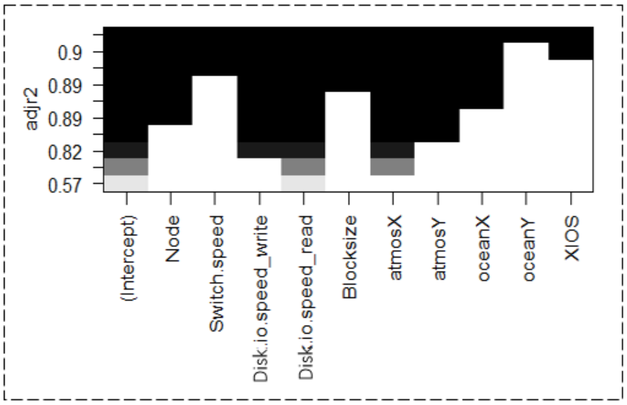


FIGURE 17. Best subset selection result of runtime data.

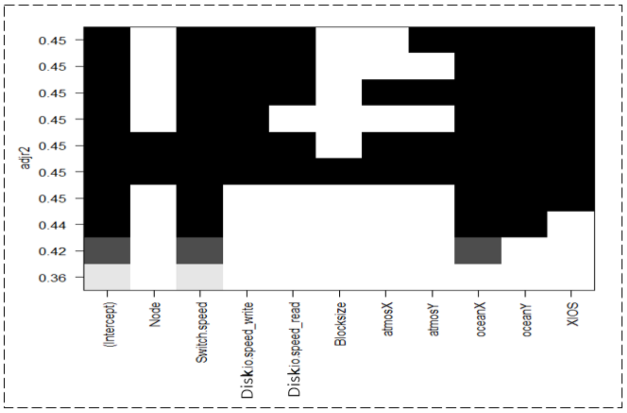


FIGURE 18. Best subset selection result of POSIX bandwidth data.

TABLE 7. VIF for each parameter.

Parameters	VIF
Node	2.3
Disk IO speed_write	2.3
Disk IO speed_read	2.9
Blocksize	2.1

regression coefficient is distorted by multicollinearity and is calculated using Equation (2). R^2 represents the coefficient of determination and can be calculated as the reciprocal of tolerance.

$$\mathrm{VIF_i} = \frac{1}{1 - R_i^2} = \frac{1}{\text{tolerance}} \tag{2}$$

If the VIF exceeds 10, it is considered multicollinearity [23]. For ease of analysis, we summarized the VIF values of the parameters suspected of multicollinearity in Table 7. As all parameter VIF values are less than 10, we confirmed that the performance data of Low GloSea6 did not exhibit multicollinearity.

TABLE 8. **Measuring RMSE of machine learning models according to hyperparameter settings.**

Dependent variables	Runtime			POSIX bandwidth			STDIO bandwidth		
Models	Hyperparameters		RMSE	Hyperparameters		RMSE	Hyperparameters		RMSE
	tree	interaction.depth	(sec)	tree	interaction.depth	(MiB/s)	tree	interaction.depth	(MiB/s)
Random Forest	100		228	100		2116	100		**1081**
	500		227	500		2107	500		1095
	1000		**225**	1000		**2101**	1000		1085
	5000		227	5000		2110	5000		1087
Gradient Boosting	100	1	466	100	1	2313	100	1	1372
		2	188		2	2030		2	1232
		3	176		3	1998		3	1094
		6	157		6	1999		6	1328
	500	1	414	500	1	2302	500	1	1329
		2	175		2	2031		2	1161
		3	159		3	2011		3	960
		6	149		6	1978		6	985
	1000	1	411	1000	1	2295	1000	1	**719**
		2	173		2	2035		2	1161
		3	156		3	2004		3	919
		6	**148**		6	1978		6	876
	5000	1	411	5000	1	**1586**	5000	1	1328
		2	165		2	2026		2	985
		3	153		3	1978		3	876
		6	149		6	1974		6	764

The best RMSE values for each model were bolded.

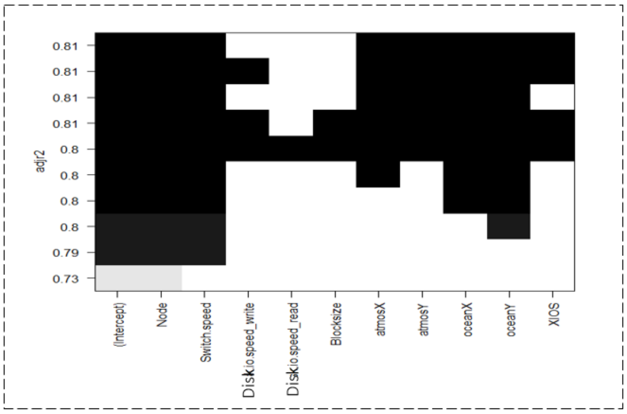
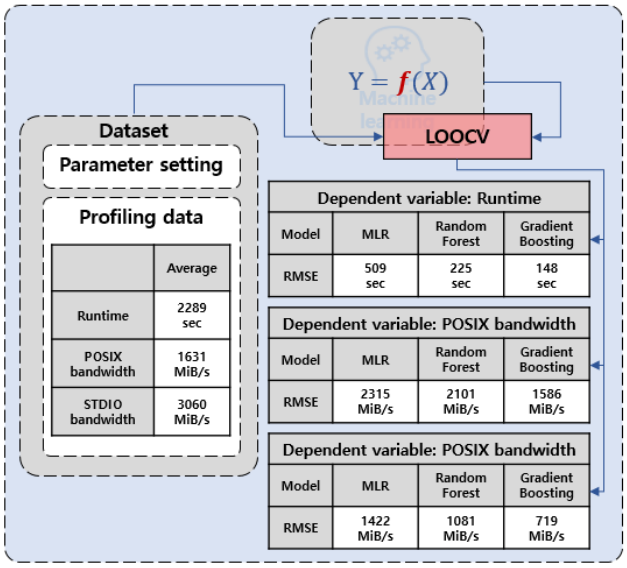


FIGURE 19. **Best subset selection result of STDIO bandwidth data.**

TABLE 9. **Changwon national university parameter.**

Parameters	Values
Node	3
Block size	32768 byte
Switch data transfer speed	100 Gbps
Disk IO speed_write	194 MB/s
Disk IO speed_read	179 MB/s



	Average
Runtime	2289 sec
POSIX bandwidth	1631 MiB/s
STDIO bandwidth	3060 MiB/s

Dependent variable: Runtime

Model	MLR	Random Forest	Gradient Boosting
RMSE	509 sec	225 sec	148 sec

Dependent variable: POSIX bandwidth

Model	MLR	Random Forest	Gradient Boosting
RMSE	2315 MiB/s	2101 MiB/s	1586 MiB/s

Dependent variable: POSIX bandwidth

Model	MLR	Random Forest	Gradient Boosting
RMSE	1422 MiB/s	1081 MiB/s	719 MiB/s

FIGURE 20. **Applying to leave one out cross validation. (Used image sources [11].)**

We then used best subset selection, a technique that fits a regression model for each subset of the parameters and identifies the optimal subset, to identify significant parameters in the performance data. Figures 17–19 show the results for each performance data. The black portion of the graph indicates that the corresponding variable on the X-axis was included in the regression model, and the Y-axis represents the model's goodness of fit based on the included

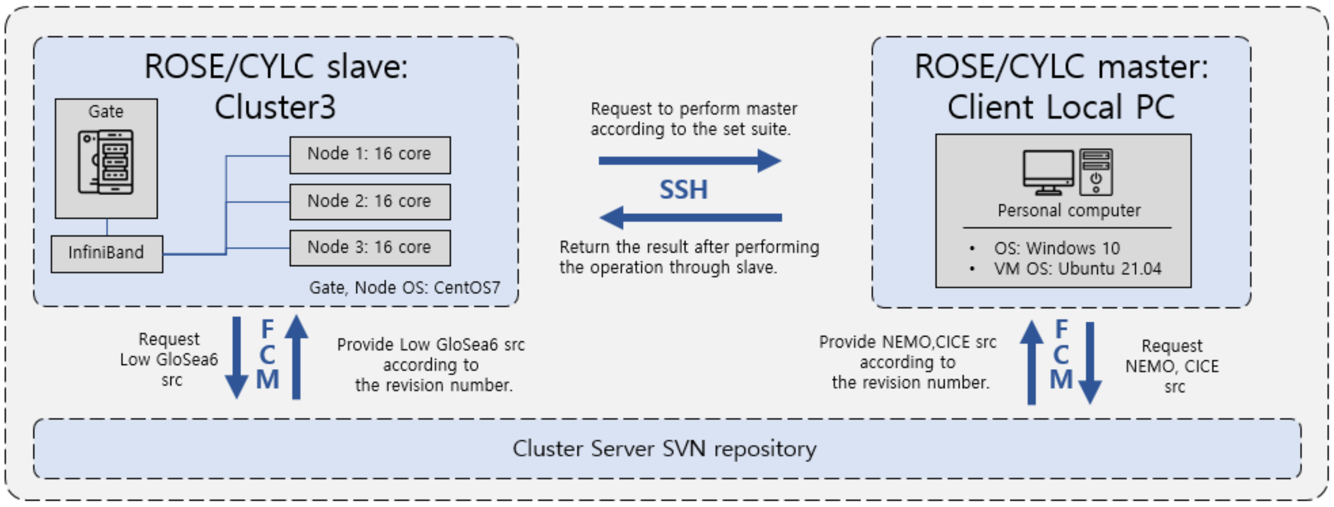

ROSE/CYLC slave: Cluster3
Gate
InfiniBand
Node 1: 16 core
Node 2: 16 core
Node 3: 16 core
Gate, Node OS: CentOS7
Request to perform master according to the set suite.
SSH
Return the result after performing the operation through slave.
ROSE/CYLC master: Client Local PC
Personal computer
OS: Windows 10
VM OS: Ubuntu 21.04
Request Low GloSea6 src
FCM
Provide Low GloSea6 src according to the revision number.
Provide NEMO,CICE src according to the revision number.
FCM
Request NEMO, CICE src
Cluster Server SVN repository

FIGURE 21. Overall composition of Changwon national university Low GloSea6 testbed. (Used image sources [11].)

variables. The higher the position on the Y-axis, the better the goodness of fit, and the goodness of fit evaluation metric, "adjr2," is calculated using Equation (3).

$$\text{adjr2} = 1 - \frac{n-1}{n-p-1} \times \left(1 - R^2\right) \tag{3}$$

where n represents the sample size, p denotes the number of parameters, and R^2 is the coefficient of determination.

Figures 17–19 show the best subset selection results when the dependent variables are runtime, POSIX bandwidth, and STDIO bandwidth, respectively. When runtime was the dependent variable, using all parameters produced the best-fit model, while excluding three parameters (Node, Blocksize, and atmosX [ATMOS_NPROCX]) yielded the best-fit model when POSIX bandwidth was the dependent variable. Excluding three parameters (Disk IO speed_write/read, Blocksize) resulted in the best-fit model when STDIO bandwidth was the dependent variable. Based on these results, we trained the model using the most suitable parameter combinations. With this, we completed the validation and adjustment of the data and proceeded with the validation and adjustment of the model.

As the number of performance data used for Low GloSea6 in this study is insufficient for model training, validating the model's effectiveness is imperative. To address this issue, we used leave-one-out cross-validation (LOOCV). LOOCV is a method that sets one of the n data as the test data and the other $n-1$ data as the training data, repeating the process n times and estimating the error by averaging the mean square error (MSE) of each model. In this study, we used RMSE instead of MSE for ease of analysis, and the LOOCV results of each model are shown in Fig. 20. In particular, the gradient boosting and random forest models used hyperparameter settings, with the smallest RMSE measured in Table 8.

We calculated the variability by dividing the RMSE of each model by the mean value of the dependent variable, multiplying it by 100, and using this as an evaluation metric. The average performance data value with runtime as the dependent variable was 2289 sec. When comparing the RMSE of the average value with those of the MLR, random forest, and gradient boosting models, they showed fluctuations of 22%, 10%, and 7%, respectively.

The average value of the performance data with the dependent variable of the POSIX bandwidth was 1631 MiB/s. When compared to the average value, the RMSE values of the MLR, random forest, and gradient boosting models showed high variability of 141%, 129%, and 97%, respectively.

The average value of the performance data with the dependent variable of the STDIO bandwidth was 3060 MiB/s. When compared to the average value, the RMSE values of the MLR, random forest, and gradient boosting models showed variability of 46%, 35%, and 23%, respectively.

Finally, we excluded the MLR model with the highest RMSE across all dependent variables in LOOCV and proceeded to the next step using the random forest and gradient boosting models with simple parameter tuning.

3) STEP3: PARAMETER PREDICTION AND OPTIMIZATION

In the final step, we optimized Low GloSea6, which operates in new industrial and research environments. Specifically, we optimized Low GloSea6 operating on Cluster 3 at Changwon National University. The university's research environment comprised three nodes with 16 cores and an InfiniBand speed of 100 Gbps, as shown in Fig. 21. To optimize the system, we set the hardware platform parameters to the values in Table 9 and input them into the machine-learning model.

The optimization system implemented in this study considers a 1-day prediction and takes into account the hardware platform parameters, CPU cores, and XIOS_NPROC values to limit the predicted values. Regarding the number of CPU

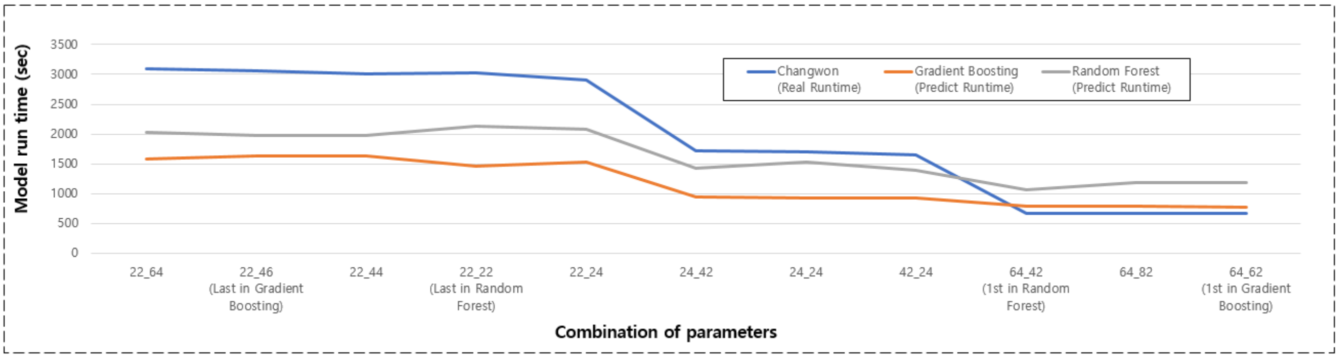


FIGURE 22. Runtime's predicted value versus actual value.

TABLE 10. Real/prediction runtime value according to parameter combination.

Sources	Indicators	Combination of parameters ('ATMOS_NPROCXY'_'NEMO_NPROCXY')										
		22_64	22_46	22_44	22_22	22_24	24_42	24_24	42_24	64_42	64_82	64_62
Changwon (Real)	Runtime (sec)	**3087**	3053	3013	3016	2903	1721	1697	1658	678	675	**671**
Gradient Boosting (Predict)		1575	**1627**	1627	1455	1532	946	929	923	799	790	**782**
Random Forest (Predict)		2037	1970	1969	**2131**	2081	1425	1425	1397	**1066**	1183	1183

The worst/best values for each model were bolded.

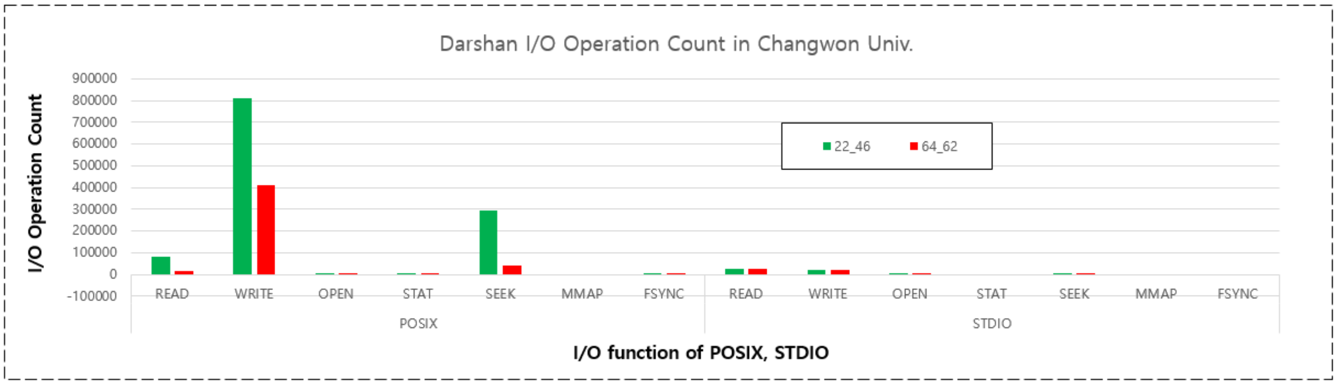
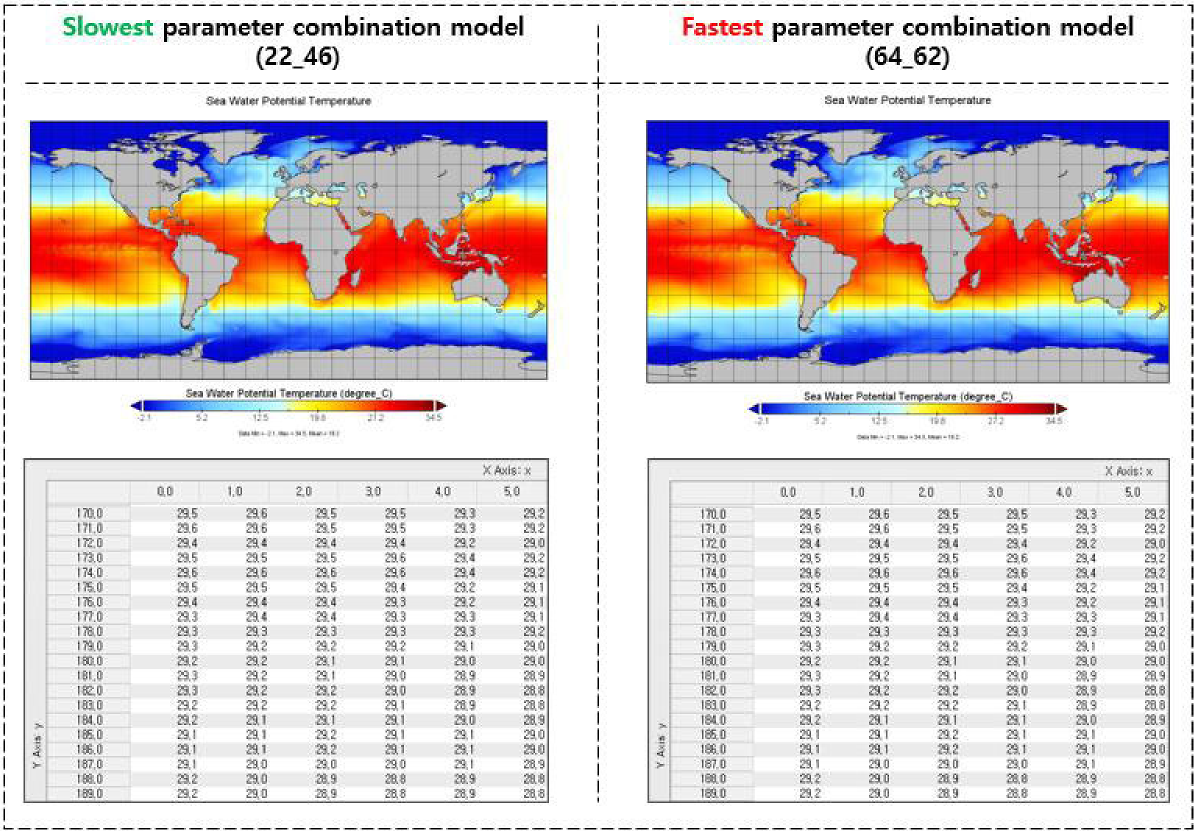


FIGURE 23. The I/O count of the model is set to the parameter value predicted by the gradient boosting.

cores, predictions are made for up to 48 cores based on the Cluster 3 hardware platform. Considering the grid size of the ATMOS and OCEAN models, the X-axis CPU core of the model can be 2, 4, 6, or 8, and the Y-axis can be 2, 4, or 6. Thus, parameter combination prediction is performed within this range. XIOS_NPROC is a parameter whose support is determined depending on the Low GloSea6 construction option. Therefore, we also set the XIOS_NPROC parameter value as input. In the case of Changwon University, the XIOS_NPROC prediction value in this study only covers 0 because the university does not support this parameter.

Using the input Cluster 3 hardware platform parameters, we predicted the optimal internal parameters of Low GloSea6 based on the dependent variables of runtime, POSIX bandwidth, and STDIO bandwidth. First, the prediction results when the dependent variable is runtime are shown in Fig. 22 and Table 10. The graph's Y-axis represents the runtime of Low GloSea6 in seconds, and the X-axis represents the internal parameter settings of Low GloSea6. For example, "12_34" means "ATMOS_NPROCX = 1, ATMOS_NPROCY = 2, NEMO_NPROCX = 3, NEMO_NPROCY = 4." The blue


Slowest parameter combination model (22_46)
Fastest parameter combination model (64_62)
Sea Water Potential Temperature
Sea Water Potential Temperature (degree_C)
-2.1
5.2
12.5
19.8
27.2
34.5
X Axis: x
Y Axis: y

	0.0	1.0	2.0	3.0	4.0	5.0
170.0	29.5	29.6	29.5	29.5	29.3	29.2
171.0	29.6	29.6	29.5	29.5	29.3	29.2
172.0	29.4	29.4	29.4	29.4	29.2	29.0
173.0	29.5	29.5	29.5	29.6	29.4	29.2
174.0	29.6	29.6	29.6	29.6	29.4	29.2
175.0	29.5	29.5	29.5	29.4	29.2	29.1
176.0	29.4	29.4	29.4	29.3	29.2	29.1
177.0	29.3	29.4	29.4	29.3	29.3	29.1
178.0	29.3	29.3	29.3	29.3	29.3	29.2
179.0	29.3	29.2	29.2	29.2	29.1	29.0
180.0	29.2	29.2	29.1	29.1	29.0	29.0
181.0	29.3	29.2	29.1	29.0	28.9	28.9
182.0	29.3	29.2	29.2	29.0	28.9	28.8
183.0	29.2	29.2	29.2	29.1	28.9	28.8
184.0	29.2	29.1	29.1	29.1	29.0	28.9
185.0	29.1	29.1	29.2	29.1	29.1	29.0
186.0	29.1	29.1	29.2	29.1	29.1	29.0
187.0	29.1	29.0	29.0	29.0	29.1	28.9
188.0	29.2	29.0	28.9	28.8	28.9	28.8
189.0	29.2	29.0	28.9	28.8	28.9	28.8

	0.0	1.0	2.0	3.0	4.0	5.0
170.0	29.5	29.6	29.5	29.5	29.3	29.2
171.0	29.6	29.6	29.5	29.5	29.3	29.2
172.0	29.4	29.4	29.4	29.4	29.2	29.0
173.0	29.5	29.5	29.5	29.6	29.4	29.2
174.0	29.6	29.6	29.6	29.6	29.4	29.2
175.0	29.5	29.5	29.5	29.4	29.2	29.1
176.0	29.4	29.4	29.4	29.3	29.2	29.1
177.0	29.3	29.4	29.4	29.3	29.3	29.1
178.0	29.3	29.3	29.3	29.3	29.3	29.2
179.0	29.3	29.2	29.2	29.2	29.1	29.0
180.0	29.2	29.2	29.1	29.1	29.0	29.0
181.0	29.3	29.2	29.1	29.0	28.9	28.9
182.0	29.3	29.2	29.2	29.0	28.9	28.8
183.0	29.2	29.2	29.2	29.1	28.9	28.8
184.0	29.2	29.1	29.1	29.1	29.0	28.9
185.0	29.1	29.1	29.2	29.1	29.1	29.0
186.0	29.1	29.1	29.2	29.1	29.1	29.0
187.0	29.1	29.0	29.0	29.0	29.1	28.9
188.0	29.2	29.0	28.9	28.8	28.9	28.8
189.0	29.2	29.0	28.9	28.8	28.9	28.8

FIGURE 24. Analysis of predictive results of parameter combination models.

line represents the actual runtime of Changwon National University, while the gray and orange lines represent the predicted runtime using random forest and gradient boosting, respectively, sorted from slowest to fastest on the left side of the graph. In particular, the parameter combination with the fastest/slowest predicted runtime by the random forest model is "64_42'/'22_22," which is highlighted in bold in Table 10, and "64_62'/'22_46" are the parameter combinations with the fastest/slowest predicted runtime by the gradient boosting model. The other parameter combinations were randomly selected from the predicted values.

The results were analyzed from two perspectives: error rate, prediction application results, and research environment. We calculated the percentage error between the predicted model runtime using Equation (4) and the actual model runtime to analyze the error rate.

$$\text{percentage error} = \frac{|\text{real} - \text{predict}|}{\text{real}} \times 100 \tag{4}$$

Figure 22 shows a similar trend between the actual and predicted values. On average, the random forest model's predictions outperformed the other models. However, as the number of CPU cores used increased, the error rate of the gradient boosting model decreased significantly, and for the optimal parameter combinations predicted by each model, the random forest model showed an error rate of 76%, while the gradient boosting model showed an error rate of 16%. While the average error is important, this study primarily aims to obtain the optimal parameter combination; thus, we adopted the gradient boosting model as a suitable model for parameter optimization.

The analysis of the application of predictions is as follows: with reference to Fig. 22 graph, we directly applied the parameter combinations with the slowest and fastest runtime predicted by the gradient boosting model, "22_46" and "64_62," respectively, to the research environment at Changwon National University. We then compared the Darshan profile data, as shown in Fig. 23. The graph summarizes the total I/O operation count of the ATMOS and OCEAN models for each parameter combination. The Y-axis represents the number of I/O operations, while the X-axis represents the type of I/O operation. A little difference was confirmed in STDIO, but the number of reads, writes, and seek operations in POSIX decreased significantly, indicating that parameter optimization can improve the model's performance and reduce the system I/O load. Additionally, we compared the prediction results of two models using different parameter combinations. Using the same input data, we conducted a 1-day forecast and compared the final prediction outputs. The size of the final prediction output files was identical for both models. To verify the

TABLE 11. Real/prediction posix bandwidth value according to parameter combination.

Sources	Indicators	Combination of parameters ('ATMOS_NPROCXY'_'NEMO_NPROCXY')										
		22_22	24_42	64_42	42_24	22_24	24_24	64_62	22_44	64_82	22_64	22_46
Changwon (Real)	POSIX bandwidth (MiB/s)	655	834	841	856	1103	1115	1693	2157	2206	2260	**2937**
Gradient Boosting (Predict)		3557	4027	4027	3869	3869	3869	6704	4338	6704	**7016**	4337
Random Forest (Predict)		2241	2914	2914	3615	3617	3147	4913	3626	4913	**5956**	3626

The worst/best values for each model were bolded.

TABLE 12. Real/prediction stdio bandwidth value according to parameter combination.

Sources	Indicators	Combination of parameters ('ATMOS_NPROCXY'_'NEMO_NPROCXY')										
		22_22	42_24	64_62	24_24	22_24	24_42	64_42	22_64	64_82	22_44	22_46
Changwon (Real)	STDIO bandwidth (MiB/s)	2051	2051	2256	2541	2626	2930	3357	3511	3595	3688	**3758**
Gradient Boosting (Predict)		68578	7395	4829	7371	8257	7260	6653	7721	4829	**9546**	**9546**
Random Forest (Predict)		6820	6751	4879	**7042**	7009	6918	5425	6170	4879	6988	6988

The worst/best values for each model were bolded.

integrity of the data, we utilized NASA's netCDF viewer, Panoply5. The left side of Fig. 24 shows the prediction results and partial data for the "22_46" parameter combination, while the right side shows the prediction results and partial data for the "64_62" parameter combination. Upon visual inspection, no discernible difference was observed, and the output data were identical, validating that our optimization method does not compromise prediction accuracy.

The analysis from the research environment perspective is as follows: the performance of the Changwon National University research environment is generally better than Cluster 1 and lower than Cluster 2. Therefore, Low GloSea6 can be effectively optimized for other research environments with hardware platform parameter values that have not been collected through the collection and learning of performance data collected from both extreme ends.

Figure 25 and Table 11 present the prediction results for the dependent variable of the POSIX bandwidth. The Y-axis in Fig. 25 represents I/O bandwidth (MiB/s), while the X-axis represents the internal parameter settings of Low GloSea6. The values in Table 11 overlap considerably as the ATMOS_NPROCX parameter was excluded through best subset selection for models using POSIX bandwidth as the dependent variable. For example, the "24_42" and "64_42" parameter combinations are treated identically in predicting POSIX bandwidth. Due to the significant margin of error and different trends observed in the predicted values for

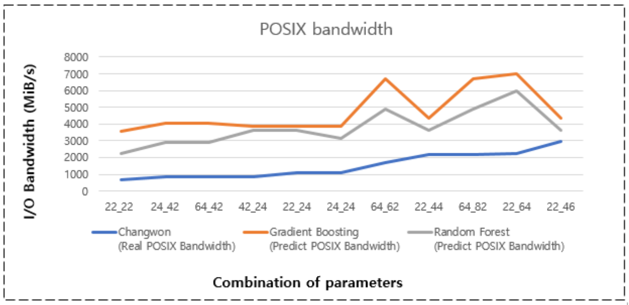


FIGURE 25. POSIX bandwidth's predicted value vs actual value.

POSIX bandwidth compared to the actual values, extracting information from the graph alone was challenging. Therefore, we present a heatmap in Fig. 26 to identify the correlation between Low GloSea6 internal parameters and POSIX bandwidth. The predicted value heatmap shows that the correlation coefficient between NEMO_NPROCX and POSIX bandwidth is above 0.8, indicating a strong correlation. Therefore, to improve POSIX bandwidth, we could predict that the NEMO_NPROCX parameter must be increased, which was validated by confirming the strong correlation between NEMO_NPROCX and POSIX bandwidth in the actual value heatmap. In summary, unlike the runtime prediction results, predicting the optimal parameter combination for the POSIX bandwidth may be challenging. However,

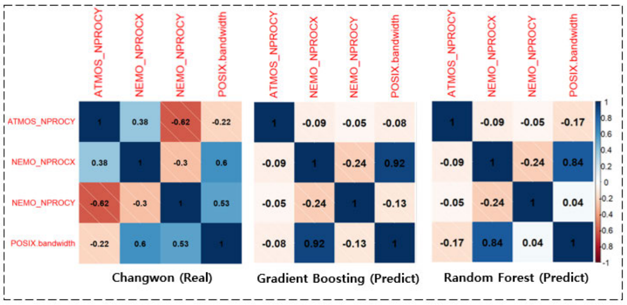


FIGURE 26. POSIX bandwidth and Low GloSea6 internal parameter correlation heatmap.

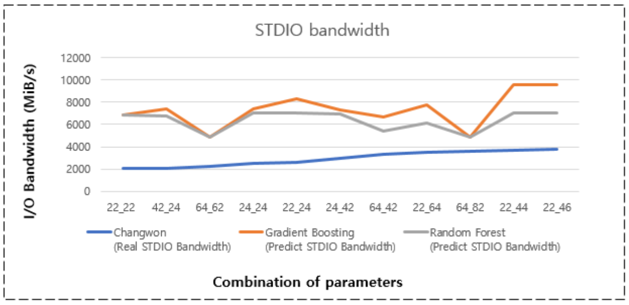


FIGURE 27. STDIO bandwidth's predicted value vs actual value.

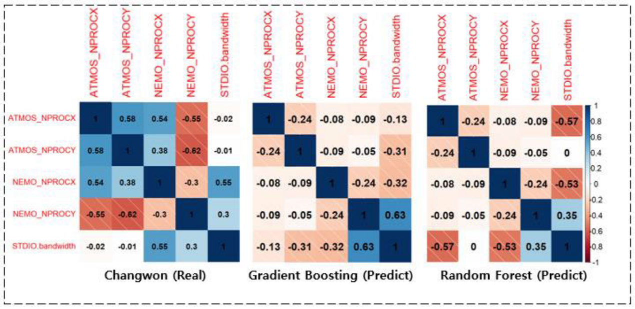


FIGURE 28. STDIO bandwidth and Low GloSea6 internal parameter correlation heatmap.

the results can provide useful reference data for determining which parameters to prioritize during optimization.

Furthermore, we present the prediction results for the dependent variable of STDIO bandwidth in Fig. 27 and Table 12. The Y-axis represents the I/O bandwidth (MiB/s), and the X-axis represents the internal parameter settings of Low GloSea6. Unlike the POSIX bandwidth, the STDIO bandwidth uses all internal parameters of Low GloSea6. However, no discernible trend was observed in the graph and heatmap presented in Fig. 27 and Fig. 28, respectively, indicating that optimization through STDIO bandwidth was impossible.

In summary, using "I/O performance estimate" metrics, such as POSIX bandwidth and STDIO bandwidth, as performance indicators for I/O performance optimization may help determine the importance of parameters in the initial optimization or as validation metrics after optimization. However, because the I/O bandwidth is an estimated value, it is inappropriate for performance optimization and prediction. Conversely, using runtime as a performance metric for overall performance optimization led to improved performance not only in runtime but also in the I/O aspect. The prediction error rate for runtime based on the optimal parameter combination was 16%, which is a significant result.

VI. CONCLUSION AND FUTURE WORK

A machine learning-based approach for optimizing hardware/software parameters of scientific applications was demonstrated in this study. The weather forecast scientific application Low GloSea6 was used as a target, and a dataset containing the application's internal parameters and hardware platform parameters and performance data based on the combination of these two parameters was constructed. Before applying the machine-learning model, the dataset was verified, and the validity of the regression model trained with insufficient data was ensured through the LOOCV technique. The optimal hardware platform parameters and corresponding Low GloSea6 internal parameters were found using the trained machine-learning model in a new research environment and these values agreed with the actual parameter combinations. In particular, the predicted execution time based on the parameter combination showed a 16% error rate compared to the actual execution time, demonstrating a meaningful result in predicting execution time. The proposed optimization method can be applied to improve the performance of other HPC scientific applications. Besides weather and climate modeling, to name a few, there are computational fluid dynamics (CFD) simulations, molecular dynamics (MD) simulations, and quantum chemistry calculations. Frequently, scientists who run such HPC scientific applications used to get help from staff members at supercomputing centers to optimize their applications, and our optimization method will help this manual performance optimization process expedited.

Two directions for future research are outlined in terms of data. First, increasing the absolute amount of data is necessary. In this study, the accurate prediction of execution time was hindered owing to the omission of some hardware platform parameters. Therefore, collecting additional hardware/software parameters and I/O performance indicators would improve model performance. Second, implementing the benchmark-based cross-inference optimization method proposed in this study's initial algorithm would be beneficial. This would accelerate data collection and enable the collection of parameter values not collected in this study through alternative parameters, thereby expanding the model performance improvement and application range.

REFERENCES

[1] *Concept of a Numerical Forecast Model*. Accessed: Aug. 10, 2023. [Online]. Available: http://web.kma.go.kr/aboutkma/intro/supercom/model/model_concept.jsp
[2] P. Davis, C. Ruth, A. A. Scaife, and J. Kettleborough, "A large ensemble seasonal forecasting system: GloSea6," Dec. 2020, vol. 2020.
[3] M. Howison, Q. Koziol, D. Knaak, J. Mainzer, and J. Shalf, "Tuning HDF5 for Lustre file systems," Lawrence Berkeley Nat. Lab., Berkeley, CA, USA, Tech. Rep. LBNL-4803E, 2010.

[4] B. Behzad et al., "Taming parallel I/O complexity with auto-tuning," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2013, p. 68.

[5] B. Behzad, S. Byna, Prabhat, and M. Snir, "Optimizing I/O performance of HPC applications with autotuning," *ACM Trans. Parallel Comput.*, vol. 5, no. 4, pp. 1–27, Mar. 2019, doi: 10.1145/3309205.

[6] S. Robert, S. Zertal, and G. Goret, "Auto-tuning of IO accelerators using black-box optimization," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Jul. 2019, pp. 1022–1027, doi: 10.1109/HPCS48598.2019.9188173.

[7] A. Bağbaba, X. Wang, C. Niethammer, and J. Gracia, "Improving the I/O performance of applications with predictive modeling based auto-tuning," in *Proc. Int. Conf. Eng. Emerg. Technol. (ICEET)*, Oct. 2021, pp. 1–6, doi: 10.1109/ICEET53442.2021.9659711.

[8] S. Valcke and R. Redler, "The OASIS coupler," in *Earth System Modelling*, vol. 3. Berlin, Germany: Springer, 2012, pp. 23–32, doi: 10.1007/978-3-642-23360-9_4.

[9] *Analysing UM Outputs*. Accessed: Feb. 14, 2023. [Online]. Available: http://climate-cms.wikis.unsw.edu.au/Analysing_UM_outputs

[10] *Unidata | NetCDF*. Accessed: Nov. 28, 2022. [Online]. Available: https://www.unidata.ucar.edu/software/netcdf/

[11] Icons8. *Free Icons, Clipart Illustrations, Photos, and Music*. Accessed: Jul. 18, 2023. [Online]. Available: https://icons8.com

[12] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, and K. Riley, "24/7 characterization of petascale I/O workloads," in *Proc. 2009 Workshop Interfaces Archit. Sci. Data Storage*, Sep. 2009, pp. 1–10.

[13] *Darshan Introduction*. Accessed: Aug. 10, 2023. [Online]. Available: https://wordpress.cels.anl.gov/darshan/wp-content/uploads/sites/54/2014/08/iiswc-2014-darshan-instrumentation.pdf

[14] R. Ross, D. Nurmi, A. Cheng, and M. Zingale, "A case study in application I/O on Linux clusters," in *Proc. ACM/IEEE Conf. Supercomput.*, New York, NY, USA, Nov. 2001, p. 11, doi: 10.1145/582034.582045.

[15] S. Herbein, D. H. Ahn, D. Lipari, T. R. W. Scogland, M. Stearman, M. Grondona, J. Garlick, B. Springmeyer, and M. Taufer, "Scalable I/O-aware job scheduling for burst buffer enabled HPC clusters," in *Proc. 25th ACM Int. Symp. High-Perform. Parallel Distrib. Comput.*, New York, NY, USA, May 2016, pp. 69–80, doi: 10.1145/2907294.2907316.

[16] *R Package*. Accessed: Apr. 26, 2023. [Online]. Available: https://cran.r-project.org/src/contrib/Archive/

[17] R. Genuer, J.-M. Poggi, and C. Tuleau, "Random forests: Some methodological insights," 2008, *arXiv:0811.3619*.

[18] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). New York, NY, USA: Springer, 2009, doi: 10.1007/978-0-387-84858-7.

[19] *Package RandomForest*. Accessed: Apr. 26, 2023. [Online]. Available: https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

[20] *Package GBM*. Accessed: Apr. 26, 2023. [Online]. Available: https://cran.r-project.org/web/packages/gbm/gbm.pdf

[21] Tekie. (Sep. 23, 2022). *SSD vs HDD—Comparing Speed, Lifespan, Reliability*. Accessed: Feb. 15, 2023. [Online]. Available: https://tekie.com/blog/hardware/ssd-vs-hdd-speed-lifespan-and-reliability/

[22] L. M. Rea and R. A. Parker, *Designing & Conducting Survey Research A Comprehensive Guide*, 3rd ed. San Francisco, CA, USA: Jossey-Bass, 2012.

[23] R. Kabacoff, *R in Action: Data Analysis and Graphics with R and Tidyverse*, 3rd ed. New York, NY, USA: Simon and Schuster, 2022.

SOOHYUCK CHOI received the bachelor's degree in science and technology computer and information communications engineering from Hongik University, Sejong, South Korea, in 2021, and the master's degree, in 2023.

Since February 2023, he has been a Researcher with the Software and Communications Engineering Department, Hongik University. His research interests include machine learning at edge and high-performance I/O for scientific applications. He is actively involved in research related to I/O optimization for weather forecasting models at the Korea Meteorological Administration.

EUN-SUNG JUNG (Senior Member, IEEE) received the bachelor's degree in electrical engineering from Seoul National University, in 1996, the master's degree in electrical engineering, in 1998, and the Ph.D. degree in computer engineering from the University of Florida, in 2010.

From 1998 to 2000, he was a Research Engineer with LG Industrial Systems, and from 2000 to 2005, he was the Team Leader of MacroImpact. From 2011 to 2012, he was a Principal Engineer with the Samsung Advanced Institute of Technology. He held a postdoctoral position with the Argonne National Laboratory, from 2013 to 2016, and during the summers of 2016, 2017, 2018, and 2019, he actively participated as a Visiting Faculty with the Faculty Research Program, Argonne National Laboratory. Since 2016, he has been an Associate Professor with Hongik University, Sejong, South Korea.

Prof. Jung is a member of IEICE. In 2011, he served as a Journal Reviewer for IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, and many other journals.