2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC) | 978-1-6654-1016-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/HIPC53243.2021.00056

An In-Depth I/O Pattern Analysis in HPC Systems

Jiwoo Bang*, Chungyong Kim*, Kesheng Wu†, Alex Sim†, Suren Byna†, Hanul Sung◇ and Hyeonsang Eom*
*Department of Computer Science and Engineering, Seoul National University, Seoul, Korea
†Computational Research Division, Lawrence Berkeley Nat'l Laboratory, Berkeley, CA, USA
◇Department of Game Design and Development, Sangmyung University, Seoul, Korea

***Abstract*—High-performance computing (HPC) systems consist of thousands of compute nodes, storage systems and high-speed networks, providing multiple layers of I/O stack with high complexity. By adjusting the diverse configuration settings that HPC systems provide, the I/O performance of applications can be improved. However, it is challenging to identify the optimal configuration settings without a thorough knowledge of the system, as each of the different I/O characteristics of applications can be an important factor for parameter decision. In this paper, we use multiple machine learning approaches to perform an in-depth analysis on I/O behaviors of HPC applications and to search for the optimal configuration settings for jobs sharing similar I/O characteristics. Improved by maximum 0.07 R-squared score, our results in overall show that jobs run on the HPC systems can obtain the predicted I/O performance for different configuration parameters with a high accuracy, using the proposed machine learning-based prediction models.**

***Index Terms*—I/O characteristic, Unsupervised learning, Feature selection, Clustering, Prediction model, High performance computing**

I. Introduction

As the era of exascale computing systems approaches, modern supercomputer systems provide HPC users with large amounts of computational resources and parallel storage systems interconnected via a high-speed network interface. Scientific HPC workloads perform large-scale data analysis using parallel processing power of computational nodes and conducting large number of I/O operations on back-end storage systems. However, HPC applications often experience poor I/O performance because of several reasons. As the massive amount of output and checkpoint files are read or written to the storage system through complex I/O software stacks, the bandwidth can be limited by the contention among multiple layers of software and hardware. In addition to bottleneck in the I/O stack, shared resource contention can adversely affect job execution time. Significant fluctuations in performance can occur occasionally depending on the number of jobs and their computational or I/O loads that are executed in parallel. The periodic software upgrade or malfunctioning hardware can also attribute to the I/O performance degradation [1].

One of the solutions for mitigating the problem is to make use of system configuration settings when the users submit their jobs. The HPC systems provide multiple tunable parameters that users can adjust via the resource scheduler. The parameters include the amount of computation and storage resources that can change the degree of parallelism in application execution. Unfortunately, the users often suffer from lack of knowledge in figuring out the optimal configuration setting, since each parameter has extremely large range of values. The growing diversity of the HPC workloads makes it more difficult to search for the configuration setting that can be highly complementary with the I/O characteristics of the workloads. The proper guidance with which to help the HPC users decide the optimal configuration setting can improve both application performance and resource utilization in the supercomputer systems.

There have been numerous works that make use of configurability in the HPC environment, in order to get the maximum performance of the applications. [2], [3] find the optimal configuration settings by using supervised, semi-supervised or unsupervised learning models in HPC environments. [4], [5] focus on characterizing the I/O behaviors of the HPC workloads in order to provide better insights on the performance. Specifically, [6]–[8] predict the I/O performance of petascale file systems using various machine learning techniques. Similar to that of previous works, our goal is to get a better insight into understanding the I/O characteristics of the HPC applications and to figure out the configurations that can result in the improved performance. We leverage the unsupervised learning model to reveal complex I/O patterns and relationships that the HPC applications have on performance. By building prediction models based on the clustering results, the predicted I/O performance with various configuration settings on different jobs can be obtained with improved prediction accuracy. The key contributions of this paper are summarized as follows:

- We collect the I/O related information from the real-world log dataset and group the jobs having similar I/O characteristics with low computational cost and improved efficiency (Section III).
- We construct prediction models based on the clustered datasets, which enables improved prediction performance (Section IV).
- We present our observation on searching for the optimal system configuration settings, using the constructed prediction models (Section V)

II. Motivation

The Cori Supercomputer system at the National Energy Research Scientific Computing Center (NERSC) is one of the most powerful HPC systems in the world. HPC users can

2640-0316/21/$31.00 ©2021 IEEE
DOI 10.1109/HiPC53243.2021.00056

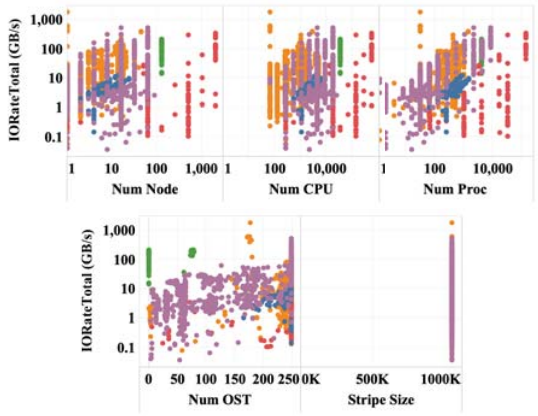

Fig. 1. The distribution of user-configurable parameters on the top five applications that have large amounts of I/O during the job execution time. The applications have different colors in the graph

run their applications on the Cori system by submitting a job script to the Slurm workload manager. The term *job* refers to a computational task that a user requests to run via the Slurm scheduler, while *application* refers to an executable program. Users can specify configurations on the job script, such as the amount of computational resource, the type of cores in the compute node to use, or the number of MPI processes with which to run the job. The limit of execution time and the storage system-related parameters can also be included in the script. Using the information that the user defined on the job script, the Slurm workload manager allocates the available resources to the users so that the jobs can be run accordingly.

We perform our analysis using the real-world user log data from Cori system from January to June, 2019 (PDT). In order to collect the HPC log data, we use the Darshan I/O profiling tool [5]. The Darshan module is a lightweight I/O instrumentation library that can capture various I/O behaviors, including application access patterns, number of I/O operations, or access sizes using multiple interfaces. Since we have to get the aggregated database of the logs to apply the machine learning models, we use the python3-based parser developed by Kim et al [9]. The parser can extract I/O-related data not only from the logs that the Darshan module provides, but also from the Slurm workload manager and Lustre Monitoring Tool. By utilizing both application-level, file system-level, and scheduler-level statistics, a total of 112 I/O related features are collected from 134,069 jobs in the analysis period.

Among the features given by the parser, there are five features that are user-configurable. The number of compute nodes(*numNode*), CPUs(*numCPU*), and processes(*numProc*) are the features that control the amount of computational resource, while the stripe count(*numOST*) and the stripe size(*stripeSize*) are the ones used to configure the file striping settings in the Lustre file system. Figure 1 shows the distribution of the user-configurable features on the jobs from the top five applications that issue the largest amount of I/O operations. The graph shows that all the jobs do not change stripe size setting in the Lustre file system, although they have a chance to improve the performance by changing the parameter. Also, only one application (marked as purple) out of five changes the stripe count dynamically, while another four use a relatively fixed number of OSTs. Another thing to notice is that 93.42% of jobs run from the other application (green) use the same configuration setting while I/O throughput varies from 14.3 to 215.6GB/s, which is approximately 15 times higher performance.

Our observations show that most of the users choose to use fixed amounts of resources to run the jobs and there are multiple factors affecting I/O performance in the HPC environment, other than the amount of resources the jobs use. When the job increases the I/O parallelism by configuring the Lustre file system to use multiple OSTs, it also increases the shared resource contention on the network and file system layer. The performance can even change due to the input parameters that are used to run the job, the information of which the profiling tools cannot capture. It is an exhaustive work to manually figure out the exact I/O related features that lead to poor performance. In order to ease the computational complexities, we use multiple machine learning models together to find the factors having a large impact on I/O performance.

III. Application of Scalable Clustering Model

A. Dataset Preprocessing

The log data from multiple profiling tools has to be refined using several data preprocessing steps before applying machine learning models. We use the jobs that issue more than 1GB I/O operations, in order to focus on the data that have enough I/O related information. Also, only the applications that have been executed more than 100 times during the analysis period are included in the dataset. To overcome the data imbalance, we sample the data to make the number of job executions of the applications to be the same by performing random over-sampling and under-sampling [10]. After the data preprocessing steps are completed, the dataset includes a total of 122,640 jobs from 84 different applications, each with the same 1,460 jobs. There are 35 features remaining in the dataset, and the target feature of clustering and prediction models is the sum of the read and write I/O throughput, also termed *IORateTotal*.

B. Feature Selection and Clustering Models

A clustering method can reveal hidden information in I/O behaviors in the HPC system and yield a better understanding of the I/O workload characteristics. The clustering algorithm calculates the similarity of the features for each data point, in order to group the jobs together. Since there are too many features that are related to I/O characteristics, it is necessary first to eliminate the irrelevant features for dimensional reduction before clustering the jobs. We seek the best feature subset that includes features having the

most dominant impact on the target feature in two steps: the Min-max mutual information feature selection, and the SBS process combined with clustering algorithm.

In order to select the features that can best represent the I/O characteristics of the data, we implement a new feature selection algorithm that chooses the features based on the correlation coefficient value. The new method, named Min-max mutual information feature selection, first selects the feature that is most strongly correlated with the *IORateTotal*, which is the target feature. Then, the second feature is selected from among the top ten least correlated features with the previously selected one, having the highest correlation value with the *IORateTotal*. In this way, not only the features that have a strong relationship with the target feature can be selected, but also the redundancy among the selected features is minimized. The process of selecting the features iterates until the desired number of features are selected.

The top ten features selected from the Min-max mutual information algorithm in our dataset are as follows: *runTime*, *OpenReqSTDIO*, *mdsOPSMin*, *writeLess1k*, *ossWriteLargestUsed*, *slowWriteTimePOSIX*, *numOST*, *totalFileSTDIO*, *writeMore1m* and *readLess1k*. Among the ten features that are selected via the Min-max mutual information algorithm, naively selecting some of the features correlated to I/O performance on clustering may not group data with similar I/O characteristics. It is also important to determine the best number of features to select for efficient clustering. To find the best feature subset that can give the highest clustering performance, we use the parallel Sequential Backward Selection (SBS) algorithm [11] , which can reduce the computational cost by searching for the best feature set in parallel.

The SBS algorithm works along with the KMeans clustering algorithm and two cluster evaluation metrics. Starting from a set of ten features, one feature at a time is removed, making ten feature subsets, each consisting of nine features. Then, the KMeans clustering algorithm is performed on each of the ten feature subsets in parallel, and the clustering result is evaluated using the Silhouette coefficient [12] and the Davies-Bouldin index (DBI) [13] metrics. The two validity metrics evaluate the cluster performance by calculating the inter-cluster variance and the inner-cluster variance. When the Silhouette coefficient score is high and the DBI score is low, the two scores indicate that the inter-cluster distance is short and the nearest-cluster distance is long, which can also be regarded as the quality of the clustering result is good. In order to combine the two validity metrics into one, we make use of the *Combined Score* validity metric, which is calculated by dividing the Silhouette score by the DBI score.

After the ten different clustering results from the ten different feature subsets are evaluated in parallel, the subset with the highest *Combined Score* is selected and becomes the starting feature set in the next iteration. The SBS process continues until three features are left in the feature set.

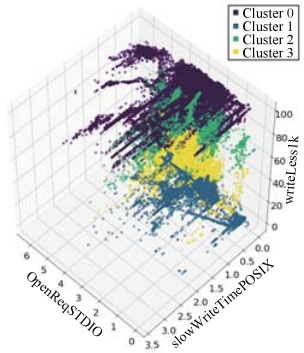

Fig. 2. Three-dimensional feature space diagram of four clusters

Since the clustering algorithm is executed and evaluated in parallel, the computation cost of searching for the optimal result with the best feature set can be reduced dramatically. The KMeans clustering algorithm runs using each feature subset and the results are evaluated on the dedicated compute nodes in parallel, reducing the computation time by approximately 82%, compared to the time taken in a serial search process.

The result shows that regardless of the number of clusters, the *Combined Score* increases as the feature set size decreases, which indicates that the SBS process eliminates the feature that is most irrelevant to I/O performance and has a negative impact on the cluster performance in every iteration. Since the objective of the clustering is to group the applications with similar I/O characteristics together and build the prediction model on each of the clustered datasets, clusters need to have a sufficient number of jobs. Considering both the *Combined Score* and the size of each clusters, we select the clustering result of four clusters by automatically calculates the cluster size variability, in the case when every cluster consists of at least 10,000 jobs.

The KMeans clustering algorithm clusters the jobs based on three features, *OpenReqSTDIO*, *writeLess1k*, and *slowWriteTimePOSIX*, which are considered to be the most relevant variables to I/O performance and can provide the best clustering performance. Figure 2 shows the three-dimensional diagram of the clustering results, with *OpenReqSTDIO* and *slowWriteTimePOSIX* being in logarithmic format. *slowWriteTimePOSIX* represents the time taken by the slowest POSIX write operation, while *writeLess1k* is the percentage of the number of write operations with an access size less than 1KB to the total number of write operations. *OpenReqSTDIO* represents the number of open requests using the STDIO interface. The diagram shows that jobs are grouped mainly based on *writeLess1k*, which represents that *writeLess1k* value highly influences the I/O performance of the job.

IV. Performance Prediction with Regression Model

A. Clustered Datasets

In the previous section, we clustered jobs based on the features highly correlated to I/O performance, from

which we obtained four clusters, each with similar I/O characteristics. Our next step is to train the prediction models using each of the clustered datasets, and evaluate the prediction performance. Since multiple prediction models are trained from each of the clusters, which model to use can be identified by the application name of the job. However, the internal I/O behaviors of two jobs from the same application can be completely different in some cases. Assuming that the parallel I/O benchmark is run on the system, depending on the command line options the users specify, one job may perform the random write operations with a 4KB block size, while another job may perform sequential read operations with a 4GB block size. Then, the profiling logs would show completely different I/O patterns, even though the two jobs are run using the same application. Therefore, the clustering algorithm would certainly place the two jobs in different clusters, considering that they have opposite I/O behaviors from each other. In this case, I/O performance cannot be predicted for the same I/O benchmark job, since it is impossible to know into which cluster the job will be grouped, and thus the appropriate prediction model cannot be distinguished.

In order to identify the proper prediction model from the application name, we restrict the clustered dataset to include applications that show similar I/O behaviors across multiple runs. For example, when 80% of the jobs from application *A* belong to Cluster 1 and the rest of the jobs belong to Cluster 2, we consider every future job from the application *A* would belong to Cluster 1. Interestingly, the clustering result shows that 66 out of 84 applications have more than 80% of the jobs grouped into the same cluster. This indicates that most of the HPC applications are run repeatedly with similar I/O characteristics, which is in line with previous works [9], [14].

B. Prediction Models

We use the KNN prediction algorithm, which is one of the most commonly used regression methods, to train the prediction models. KNN calculates the similarity between the given feature values of trained data and test data to find the K-Nearest neighbors, from which the average of the neighbors becomes the predicted value. We aim to improve the prediction performance by using the training dataset that includes jobs having common I/O characteristics. To evaluate our clustering-based prediction models, we train the models and measure the R-squared score using five different datasets: the four clusters we get in Section III, and the un-clustered dataset that has undergone only the data preprocessing step for baseline evaluation. The training set and test set are created by randomly dividing the jobs of each dataset into an 8:2 ratio, respectively. The input parameters to the model are the five user-configurable features, information of which can be obtained before the job execution time.

Table I shows the number of applications, jobs and the R-squared score, which can represent the predictive accuracy,

TABLE I
EVALUATION OF PREDICTION MODELS USING DIFFERENT DATASETS

Dataset	Number of Apps	Number of Jobs	R-squared score
Cluster 0	35	51,100	0.86
Cluster 1	22	32,120	0.91
Cluster 2	6	8,760	0.82
Cluster 3	3	4,380	0.20
Un-clustered Data	84	122,640	0.84

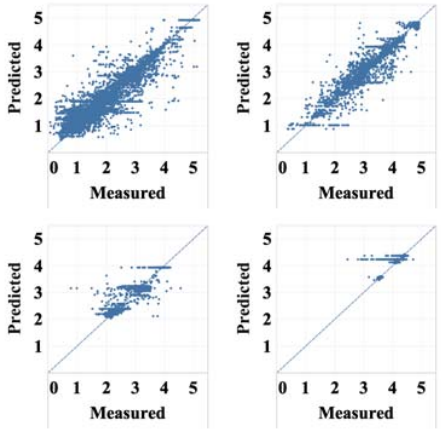

Fig. 3. Prediction plots of the measured versus predicted I/O performance (MB/s) of jobs being in logarithmic format (top left Cluster 0, top right Cluster 1, bottom left Cluster 2, bottom right Cluster 3)

on five types of datasets. The number of applications and jobs in the un-clustered dataset is larger than the sum of those in the four clusters, because we only include applications that have similar I/O characteristics across the runs in the four clusters. The evaluation result shows that the R-squared score is higher with Clusters 0 and 1, when compared to the score of the un-clustered dataset. One of the reasons why the score of Cluster 3 is extremely poor is because only three applications are included in the cluster, which is considered to be too small to train the model. Also, since most of the jobs in Cluster 3 use a fixed value of user-configurable parameters, the prediction model cannot accurately predict I/O performance with different configurations. Figure 3 plots the measured and predicted I/O performance of the jobs in logarithmic scale in the four clusters. The horizontal dots in the bottom right graph indicate that the prediction model trained by jobs in Cluster 3 predicts the same I/O performance most of the time, due to the lack of training data. In contrast to Cluster 3, the prediction model created from Cluster 1 shows the highest R-squared score, due to the dynamic configuration settings the users configured in running the jobs. The average R-squared score with different weights for Cluster 0, 1, and 2 based on the cluster size is calculated as 0.87, which is higher than the score of un-clustered dataset. Overall, our result shows that clustering the jobs with similar I/O behaviors can help increase the R-squared score and makes it possible to predict I/O performance of the jobs with improved accuracy.

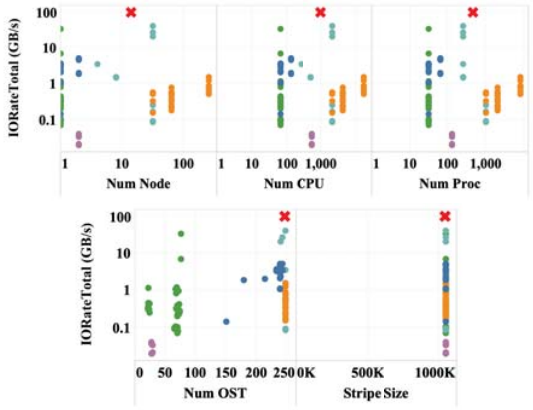

Fig. 4. The optimal user-configurable parameters obtained from the Cluster 0-based prediction model. The applications have different colors in the graph

V. Evaluation

In this section, we search for the optimal system configuration settings for HPC workloads, using the trained prediction model. We only show the Cluster 0-based trained prediction model and omit the other models due to page limitation. Other than the Cluster 3-based model, which has poor performance because of the lack of information in the training data, Cluster 0, 1 and 2-based models show the similar performance. We create the test dataset by varying the five user-configurable parameters in different ranges, which are determined based on the Cori system user log data, run from July to September 2019 (PDT). Specifically, *numNode* has the range between 1 to 3,850 and *numCPU* is determined as the product of 64 or 272 times *numNode*, which is the number of cores in the Haswell and KNL nodes respectively. Also, the product of *numNode* times 2 to the power of 0 to 8 is determined as *numProc*, referring to the log data. *numOST* ranges from 1 to 248, which is same as the number of OSSs in the Cori system, and *stripeSize* is determined to have one of 1, 2, 4, 8, 16, and 32MB values. I/O performance is predicted for every combination of the five parameters in the prediction model. Then the configuration parameters with the highest I/O performance in the model are selected.

Figure 4 shows the distribution of the user-configurable parameters of the jobs and the optimal configurations that give the highest I/O performance, using the prediction models based on Clusters 0. Among the applications included in the cluster, we plot the top five I/O-intensive applications, run from July to September 2019 (PDT), which is the time period that starts right after our training period. The red marks in the graphs represent the optimal configuration settings that are searched from the prediction models. Compared to the performance of the jobs in all five applications, the red marks show the highest I/O throughput. Although we omit the figures showing the results of the Cluster 1 and 2-based models, the optimal user-configurable parameters that are searched using the two models also give the highest I/O performance compared to the performance of the jobs that run between the test time period. By using the predicted performance for various configuration settings, not only can users get a better understanding of configuring the jobs and achieve higher performance, but also the system can efficiently schedule the limited amount of resource in the HPC environment.

While we show that I/O performance can be predicted with different combinations of configuration parameters, only the methodology for building the prediction model based on the clustering results is applicable to other HPC systems. We obtain the optimal configuration settings from the regression models that are trained with jobs run in the specific analysis period, January to June, 2019 (PDT). Considering that the supercomputer system changes significantly over time for multiple reasons, such as periodic maintenance, hardware replacement, and software upgrades, the results that we observe are not expected to be found in general. The objective of our work is to provide the methodology that can predict I/O performance of HPC workloads with high accuracy, using various machine learning models. By periodically searching for the optimal user-configurable parameters using our methodology, HPC users can be provided with the configuration setting guidance that can give improved I/O performance.

VI. Related Works

Many researchers and HPC users wonder about how to achieve optimal performance on HPC systems for each application they utilize [9]. In order to enhance overall I/O performance for the applications run on HPC systems, it is crucial to analyze logs from the past jobs and search for optimizable features. [15]–[17] tried to understand the correlation of various features collected by Darshan logs in HPC environment. Gauge [18] provided a web application that calculates unorganized logs of HPC jobs and showed a interactive visualization of the workloads that ran on the HPC system. However, only 61.4% of the data variances are taken into account with PCA. On the other hand, through Min-max feature selection and SBS, all the features are thoroughly considered in order to get the best clustering result in our work. Also, by utilizing these methods, most of the applications we analyzed gather into a single cluster, such that users can easily look for the cluster they need.

Also, there have been numerous works that have tried to predict the performance of applications to enhance the overall result. Lux et al. [19] analyzed a benchmark with a set of configurations in order to build a prediction model. The study suggested that the multivariate model can accurately predict I/O performance. Other works [6], [20]–[22] also predicted I/O performance for HPC-scale clusters using various methods. Our work also targets prediction of the performance of the application using characteristics of the HPC workloads. In contrast, our work is focused on analyzing the system using Darshan logs and makes the prediction based on the historical logs collected in the same

HPC environment. This enables our work to make a prediction based on I/O behaviors of HPC applications and give suggestions for input parameters for various applications.

VII. Conclusion

We used multiple machine learning techniques together to efficiently discover hidden information in the complex I/O behaviors of scientific workloads with low computational cost. By clustering the unlabeled user log data, applications having similar I/O characteristics with each other can be grouped together. Our results showed that the R-squared scores on the prediction models built on each of the clusters are higher than the score of the unclustered dataset, indicating that the clustering can help improve the prediction accuracy. Using the methodology that we introduced, I/O performance can be predicted for any system configuration setting in advance, which can help users find the best configuration settings for improved I/O performance of the jobs. Also, efficient resource scheduling in the HPC environment can be achieved by using the constructed prediction models.

Acknowledgment

This work was supported by the National Research Foundation (Grant NRF-2016M3C4A7952587, PF Class Heterogeneous High Performance Computer Development), BK21 FOUR Intelligence Computing (Dept. of Computer Science and Engineering, SNU) funded by National Research Foundation of Korea(NRF) (4199990214639), the Seoul Business Agency (Grant CY200038), the MIST(Ministry of Science, ICT), Korea, under the National Program for Excellence in SW), supervised by the IITP(Institute of Information & communications Technology Planing & Evaluation) in 2021"(2019-0-01880). This work was supported in part by Korea Institute of Science and Technology Information (Grant No.K-21-L02-C01-S01) and the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231. This work also used resources of the National Energy Research Scientific Computing Center (NERSC).

References

[1] E. Ates, Y. Zhang, B. Aksar, J. Brandt, V. J. Leung, M. Egele, and A. K. Coskun, "Hpas: An hpc performance anomaly suite for reproducing performance variations," in *Proceedings of the 48th International Conference on Parallel Processing*, ser. ICPP 2019. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3337821.3337907

[2] H. Menon, A. Bhatele, and T. Gamblin, "Auto-tuning parameter choices in hpc applications using bayesian optimization." Institute of Electrical and Electronics Engineers Inc., 5 2020, pp. 831–840.

[3] S. Robert, S. Zertal, and P. Couvee, "Shaman: A flexible framework for auto-tuning hpc systems," vol. 12527 LNCS. Springer Science and Business Media Deutschland GmbH, 11 2021, pp. 147–158. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-68110-4_10

[4] P. J. Pavan, J. L. Bez, M. S. Serpa, F. Z. Boito, and P. O. A. Navaux, "An unsupervised learning approach for i/o behavior characterization," in *2019 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2019, pp. 33–40.

[5] S. Snyder, P. Carns, K. Harms, R. Ross, G. K. Lockwood, and N. J. Wright, "Modular hpc i/o characterization with darshan," in *2016 5th Workshop on Extreme-Scale Programming Tools (ESPT)*, 2016, pp. 9–17.

[6] B. Xie, Y. Huang, J. S. Chase, J. Y. Choi, S. Klasky, J. Lofstead, and S. Oral, "Predicting output performance of a petascale supercomputer," in *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 181–192. [Online]. Available: https://doi.org/10.1145/3078597.3078614

[7] S. Madireddy, P. Balaprakash, P. Carns, R. Latham, R. Ross, S. Snyder, and S. Wild, *Machine Learning Based Parallel I/O Predictive Modeling: A Case Study on Lustre File Systems*, 01 2018, pp. 184–204.

[8] D. Li, Y. Wang, B. Xu, W. Li, W. Li, L. Yu, and Q. Yang, "Pipuls: Predicting i/o patterns using lstm in storage systems," in *2019 International Conference on High Performance Big Data and Intelligent Systems (HPBD IS)*, 2019, pp. 14–21.

[9] S. Kim, A. Sim, K. Wu, S. Byna, Y. Son, and H. Eom, "Towards hpc i/o performance prediction through large-scale log analysis," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 77–88. [Online]. Available: https://doi.org/10.1145/3369583.3392678

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, Jun 2002. [Online]. Available: http://dx.doi.org/10.1613/jair.953

[11] J. Wang, W. Yoo, A. Sim, P. Nugent, and K. Wu, "Parallel variable selection for effective performance prediction," 05 2017, pp. 208–217.

[12] P. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, no. 1, p. 53–65, Nov. 1987. [Online]. Available: https://doi.org/10.1016/0377-0427(87)90125-7

[13] D. Davies and D. Bouldin, "A cluster separation measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-1, pp. 224 – 227, 05 1979.

[14] R. G. Tirthak Patel, "Gift: A coupon based throttle-and-reward mechanism for fair and efficient i/o bandwidth management on parallel storage systems." *Proceedings of the 18th USENIX Conference on File and Storage Technologies, 2020.* [Online]. Available: https://par.nsf.gov/biblio/10200013

[15] G. Ren, E. Tune, T. Moseley, Y. Shi, S. Rus, and R. Hundt, "Google-wide profiling: A continuous profiling infrastructure for data centers," *IEEE Micro*, vol. 30, no. 4, pp. 65–79, 2010.

[16] Y. Lv, B. Sun, Q. Luo, J. Wang, Z. Yu, and X. Qian, "Counterminer: Mining big performance data from hardware counters," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2018, pp. 613–626.

[17] A. Yasin, "A top-down method for performance analysis and counters architecture," in *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2014, pp. 35–44.

[18] E. del Rosario, M. Currier, M. Isakov, S. Madireddy, P. Balaprakash, P. Carns, R. B. Ross, K. Harms, S. Snyder, and M. A. Kinsy, "Gauge: An interactive data-driven visualization tool for hpc application i/o performance analysis," in *2020 IEEE/ACM Fifth International Parallel Data Systems Workshop (PDSW)*, 2020, pp. 15–21.

[19] T. Lux, L. Watson, T. Chang, J. Bernard, B. Li, L. Xu, G. Back, A. Butt, K. Cameron, and Y. Hong, "Predictive modeling of i/o characteristics in high performance computing systems," 01 2018.

[20] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for large-scale advanced analytics," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, Mar. 2016, pp. 363–378. [Online]. Available: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/venkataraman

[21] A. Ali-Eldin, J. Tordsson, E. Elmroth, and M. Kihl, "Workload classification for efficient auto-scaling of cloud resources," 2013.

[22] Z. Hou, S. Zhao, C. Yin, Y. Wang, J. Gu, and X. Zhou, "Machine learning based performance analysis and prediction of jobs on a hpc cluster," in *2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, 2019, pp. 247–252.