HPC Workload Characterization Using Feature Selection and Clustering

Jiwoo Bang
Dept. of Comp. Science & Engineering
Seoul National University
Seoul, Korea
jwbang@dcslab.snu.ac.kr

Chunyong Kim
Dept. of Comp. Science & Engineering
Seoul National University
Seoul, Korea
cykim@dcslab.snu.ac.kr

Kesheng Wu
Computational Research Division
Lawrence Berkeley Nat'l Laboratory
Berkeley, CA, USA
kwu@lbl.gov

Alex Sim
Computational Research Division
Lawrence Berkeley Nat'l Laboratory
Berkeley, CA, USA
asim@lbl.gov

Suren Byna
Computational Research Division
Lawrence Berkeley Nat'l Laboratory
Berkeley, CA, USA
SByna@lbl.gov

Sunggon Kim
Dept. of Comp. Science & Engineering
Seoul National University
Seoul, Korea
skim@dcslab.snu.ac.kr

Hyeonsang Eom
Dept. of Comp. Science & Engineering
Seoul National University
Seoul, Korea
hseom@cse.snu.kr

ABSTRACT

Large high-performance computers (HPC) are expensive tools responsible for supporting thousands of scientific applications. However, it is not easy to determine the best set of configurations for workloads to best utilize the storage and I/O systems. Users typically use the default configurations provided by the system administrators, which typically results in poor performance. In an effort to identify application characteristics more important to I/O performance, we applied several machine learning techniques to characterize these applications. To identify the features that are most relevant to the I/O performance, we evaluate a number of different feature selection methods, e.g., Mutual information regression and F regression, and develop a novel feature selection method based on Min-max mutual information. These feature selection methods allow us to sift through a large set of the real-world workloads collected from NERSC's Cori supercomputer system, and identify the most important features. We employ a number of different clustering algorithms, including KMeans, Gaussian Mixture Model (GMM) and Ward linkage, and measure the cluster quality with Davies Boulder Index (DBI), Silhouette and a new Combined Score developed for this work. The cluster evaluation result shows that the test dataset could be best divided into three clusters, where cluster 1 contains mostly small jobs with operations on standard I/O units, cluster 2 consists of middle size parallel jobs dominated by read operations, and cluster 3 include large parallel jobs with heavy write operations. The cluster characteristics suggest that using parallel I/O library MPI IO and a large number of parallel cores are important to achieve high I/O throughput.

SNTA '20, June 23, 2020, Stockholm, Sweden
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7980-9/20/06…$15.00
https://doi.org/10.1145/3391812.3396270

CCS CONCEPTS

• **Information systems → Clustering and classification**; • **Computing methodologies → Feature selection**; *Machine learning*; Classification and regression trees.

KEYWORDS

High performance computing; Supercomputer; Feature selection; Clustering; Workload characterization

ACM Reference Format:
Jiwoo Bang, Chunyong Kim, Kesheng Wu, Alex Sim, Suren Byna, Sunggon Kim, and Hyeonsang Eom. 2020. HPC Workload Characterization Using Feature Selection and Clustering. In *3rd International Workshop on Systems and Network Telemetry and Analytics (SNTA '20), June 23, 2020, Stockholm, Sweden.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3391812.3396270

1 INTRODUCTION

High performance computing (HPC) systems are widely used for applications that consume or produce massive amounts of data. As the number and diversity of HPC applications grow rapidly, I/O demands vary according to applications. In order to ease the problem, the users are provided with several tunable parameters, such as the number of compute nodes, the number of storage nodes, and file system striping settings. By configuring these parameters, users can utilize HPC system efficiently and optimize I/O performance with their applications. However, as most of the users are not familiar with these tunable parameters, they use default configurations the system provides. Since there is only one default configuration in each system, some of the HPC applications do not meet the I/O demands and often obtain low performance without knowing a proper reason. Hence, it is necessary for system administrators to simplify the use of system configurations, providing appropriate system environment for different workloads. In order to understand different I/O demands of HPC applications, it is necessary to be characterized with a metric. Commonly used metrics to categorize I/O behaviors in HPC applications include duration of job execution, number of CPUs used, and memory usage [18]. Further, I/O throughput, number of IOPS, the I/O library used, and metadata operations, etc. can classify I/O workloads in detail. However it is challenging to figure out which of these metrics to use and on what criteria to classify applications with. As applications run in increasingly complicated process, trivial metrics may not capture characteristics of applications well [11]. Moreover, it is difficult to set certain criteria for each of the metrics and to manually cluster the workloads according to them.

In this paper, we use an empirical approach to cluster HPC workloads, based on their I/O behaviors. HPC workload dataset is collected from 4-month real-world application logs run on Cori system, Cray X40 supercomputer used in NERSC. Since hundreds of applications run on the supercomputer every day, it is indispensable to use machine learning methods to get meaningful information from big data and cluster them. Therefore, after proper pre-processing step is done on the dataset, several feature selection methods are used to reduce dimensions of big data. Selected features from feature selection algorithms can best represent the dataset, and can be used in next phase, the clustering methods. DBI, Silhouette and Combined Score cluster validity metrics are used to evaluate the impact selected features have on clustering results. Accordingly, the results that can best represent the dataset and the best clustering method are selected. Thereafter, clustering result of each cluster methods are characterized by determining features having significant values on each of the clusters. From our result, we can categorize the HPC workloads into three clusters, each having cluster characteristics of small size read requests, large size I/O operations and large number of total requests including metadata operations. The system administrators can get a hint from our clustering result when suggesting users the proper system configuration setting. The overall diagram of our work is depicted in Figure 1, including preprocessing, clustering and characterization steps.

In summary, our contributions on this paper can be described as follows:

- We design and implement new feature selection method, Min-max mutual information, in order to get meaningful information from real HPC workload data.
- We combine multiple cluster performance evaluation metrics into new metric, Combined Score, for better understanding of the cluster results.
- The improved feature selection method and cluster quality score allow us to identify meaningful clusters from the large set of application logs. For examples, one cluster contains mostly small parallel jobs using standard I/O units for input and output operations, which achieve very low I/O throughput. The larger parallel jobs are clearly separated into other groups with distinctive characteristics.

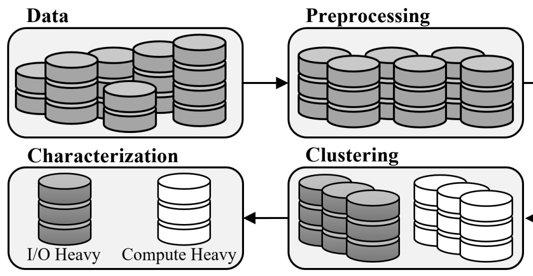

Figure 1: Total Diagram

The rest of the paper is organized as follows. The related work is surveyed in Section 2 and Section 3 introduces background of our work. Section 4 how we apply preprocessing steps to our data. Section 5 explores several feature selection methods we used and results of each of them are discussed. Section 6 presents clustering methods and the cluster validity metrics we use. The clustering performance is evaluated in Section 7 and the best clustering result is characterized and analyzed in Section 8. Finally, we conclude the paper in Section 9.

2 RELATED WORKS

Analysis and categorization of HPC workload is a broadly studied topic in the literature [5, 16]. Researches related to our work are in three major categories: clustering and characterization of data logs, error detection and analysis, and job scheduling based on past data categorization.

A way to analyze workload logs is to categorize dataset with feature selection and clustering. Mishra et al. [19], classify Google Cloud task with following features: usage of CPU, memory, disk, and network. With total 18 classes, tasks are fit according to their quantitative characteristics. Even though our work mainly focus on deploying I/O-related features from HPC workloads, we can easily change our target into CPU or memory related ones using performance and debugging tools other than Darshan, the HPC I/O characterization tool. Rodrigo et al. [21], characterize NERSC Torque logs from 2010 to 2014, by various size, time, and diversity. K-means clustering is utilized to discover minimum number of dominant clusters in an HPC workload. Terai et al. [17], analyze K computer workload data and classify them with k-means and DBSCAN. The classification is focused on diagnosis of certain clusters with poor performance metrics. Betke and Kunkel [3], identify and cluster applications with similar I/O characteristics. Different from others, monitoring data is partitioned into smaller windows for better I/O clustering. However, most of the mentioned work specifically choose a cluster number and does not check for validity of clusters which is crucial for choosing the number of clusters. In contrast, our work carefully choose the number of clusters with credible metrics.

Another way to make use of HPC workload logs is to detect and analyze errors and anomalies [9]. Fronza et al. [7], utilize weighted SVM to classify log files, either fail or non-fail. With various features such as application complexity and number of sequences, they manage to separate failed applications and predict them for energy efficiency. Tuncer et al. [25], present a framework to automatically diagnose previously encountered performance anomalies in HPC systems. Using NAS Parallel Benchmarks, feature selection are done with random forest and ST-Lan [14]. Also, the experiments are evaluated with five different classification methods.

As demand for I/O intensive workloads increases, researches on efficient job scheduling with analyzing past data through machine learning [15]. Cunha et al. [4], suggest HPC users decide where to run jobs in HPC hybrid cloud environments. The decision relies on queue waiting time and execution time of the jobs, which are predicted using traces from past job scheduling data. For the feature selection, root mean squared error is used to choose the optimal number of features. Rodrigues et al. [22], aim to help HPC users predict their memory requirements with four different machine learning algorithms. From the results, there is no single method that produces the best predictions. Different from our study, their proposed tool leverages the predictions of all methods and select the most promising ones at a given situation.

3 BACKGROUND

Supercomputer Cori system, a Cray X40, has been delivered since 2017 at National Energy Research Scientific Computing Center (NERSC) [1]. Cori is comprised of 2,388 Intel Xeon Haswell processor nodes and 9,688 Intel Xeon Phi Knights Landing nodes. In addition, Cori also has 1.8TB Cray Data Warp Burst Buffer with a performance of 1.7TB/s, which user can use by specifying APIs. All the nodes are connected with Cray Aries high-speed inter-node network and Dragon fly topology. For efficiently handling parallel I/O, Cori uses Lustre scratch file system as its disk-based storage system. Lustre file system consists of 248 OSSs including 41 HDDs and 248 OSTs, providing total 27TB of storage with peak performance of 744GB/s. When HPC users submit their job using Slurm workload manager, they can specify the amount of resources used to run their applications. For example, they can specify the number of processors or storage nodes and whether to use Burst Buffer while running their jobs.

We focus on applying empirical approach to machine learning for clustering HPC applications. So we use the raw data consists of real-world user data log run from October 2017 to January 2018 on Cori system. Specifically, Darshan I/O profiling tool is used to capture I/O behaviors of each of the 4-month jobs submitted and run on HPC environment. Darshan module, as a lightweight I/O characterization tool, can capture different I/O-related data from I/O stack from the beginning of the execution to the application shutdown time. Moreover, Darshan interacts with Slurm workload manager and Lustre monitoring tool to extract job information and I/O specific data on parallel file system. All the collected data from Darshan is merged at the time when the submitted job ends. Darshan log is created per each job running on the system and can be transformed into a file using the darshan-parser utility. In order to make dataset out of tens of thousands Darshan log text files to be used as machine learning training data, we use the approach developed by Kim et al. [13]. This approach implements a parser to extract meaningful information from Darshan text file. Figure 2 shows some of the features extracted from the parser. The approach from Kim et al. calculates *writeRateTotal* as total I/O size from POSIX, MPIIO and STDIO operations, divided by the largest I/O time of all process. In this paper, we use total 78 features obtained from the Darshan parser.

Feature	Description
writeRateTotal	Total write throughput
numProc	Number of processes used
numOST	Number of OSTs used
stripeSize	Lustre stripe size
totalIOReq	Total number of read/write operations
totalMetaReq	Total number of open/seek/stat operations
seqWritePct	Percentage of sequential I/O operations
totalFile	Total bytes read and written
readMore1m	Read operations with size more than 1M
ossWriteHigher4g	Number of OSSs with more than 4G write I/O

Figure 2: Features extracted from Darshan Profiler

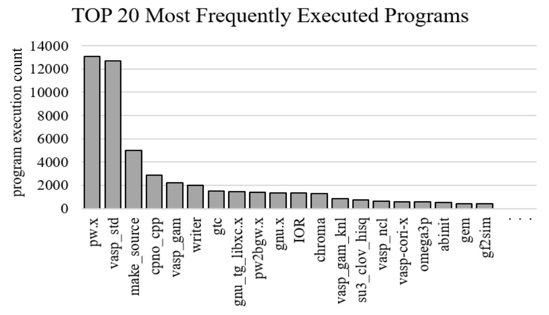

Figure 3: Top 20 mostly executed programs

4 DATA PREPROCESSING

Since using raw data to future processing steps can result in irrelevant results and low accuracy, data preprocessing is required in machine learning process. There are five steps in our preprocessing phase. First, the target variable is set to I/O throughput, which is *writeRateTotal* in our feature set. The target is chosen because our goal is to categorize HPC applications based on their I/O behaviors. The applications with less I/O operations can not precisely capture the relationship between the features extracted from Darshan module and I/O throughput. So, in order to minimize the impact the unnecessary data has on machine learning algorithms, we only use the dataset having more than 1GB I/O. Second, the data having negative values are all set to zero since they are the error values in darshan log. Third, we eliminate features with zero variance as they always have a constant value. Fourth, the features having highly correlated value with other features are eliminated using pandas library. We set the correlation value threshold to 0.8. For instance, if feature A and B have correlation value of 0.9, feature B is eliminated from the feature set. This step is included to reduce redundancy among the feature selection results. Since the eliminated features are chosen from the library, some of the representative features may not be included in the characterization result in Section 8. Finally, all the feature data is normalized to range from 0 to 1. In this way, features can have same scale and weight when calculated by feature selection methods.

Figure 3 is an histogram of top 20 mostly executed applications in our dataset. There are total 62,946 entries from 353 different applications after preprocessing step is done.

5 FEATURE SELECTION FOR DIMENSION REDUCTION

In order to understand the complex dataset and reduce computing power, selecting features that can best represent the data is necessary. We use five different feature selection methods to select features highly related to our target value. Each feature selection methods selects features with different algorithms, and has different impact on clustering models.

5.1 Feature selection Methods

5.1.1 Mutual Information Regression. Univariate feature selection runs statistical tests to find features having high relationship with target value. There are several statistical tests that can be applied to univariate method. Mutual information regression, one of the regression model in univariate feature selection, captures dependency between two features [26]. Specifically, Mutual information represents the relevance, redundancy and complementarity between the two variables. If the feature is highly related to the target and less redundant to the rest of the features, it scores high on Mutual information regression.

5.1.2 F Regression. The other statistical model that can be used in univariate feature selection is F regression. F regression method first calculates the correlation between each features and target. Then the calculated value is converted to F value, which is F regression score in our case, on the F distribution. F value can be calculated by the variance of the group means divided by the mean of the within group variances. In other words, the larger the f value, the more discriminative the feature value is. However, since F regression only considers linear dependency between the feature and target value, the result is less significant than Mutual information regression result, which not only capture linear dependency but also the dependency among the features.

5.1.3 Decision Tree. Decision tree algorithm can be used as one of the regression models in feature selection. Starting from the total dataset, Decision tree regressor splits the data by choosing a random variable. The quality of a split is measured by several rules including Mean Squared Error (MSE) and Mean Absolute Error (MAE). Since we are using MSE for split criterion, the average of MSE of each subsets is calculated for each chosen random variable. Then Decision tree regressor splits the data in a way that results in smallest MSE. The process is repeated for maximum depth of the tree or until the leaf node of the tree contains certain number of data [23][10]. In our case, the maximum depth of the tree is set to 10. The decision tree score is computed by total reduced node impurity by result, which is also known as Gini importance.

5.1.4 Extra tree. Extra tree, which refers to extremely randomized trees, is an ensemble model relying on independent decisions of the trees. Extra tree increases more randomness than Decision tree by splitting the randomly selected features. From the selected features by each trees in Extra tree model, the best split is chosen by split criterion, in this case, MSE. We use 100 trees to select random subset of features. The extra tree score is the feature importance computed as normalized total reduction of criterion, which is the same as decision tree score.

Feature	Score
seqWritePct	1.233037
totalFile	1.138258
totalOpenReq	1.096744
totalIOReq	1.082124
numProc	1.005036
runTime	0.973189
totalReadReq	0.937343

(a) Mutual Information

Feature	Score
totalFileSTDIO	25849.56
totalFile	3074.00
numProc	1281.61
numOST	957.34
readLess1m	464.63
ossWriteMean	411.40
ossWriteHigher1g	294.13

(b) F Regression

Feature	Score
totalFileSTDIO	0.353953
runTime	0.079576
runProc	0.075793
totalFile	0.054656
totalReadReq	0.053308
seqWritePct	0.050254
writeTimePOSIXonly	0.049154

(c) Decision Tree

Feature	Score
totalFileSTDIO	0.357795
runProc	0.073166
runTime	0.069633
totalFile	0.054624
totalReadReq	0.051944
seqWritePct	0.049576
writeTimePOSIXonly	0.046418

(d) Extra Tree

Feature	Score
readMore1m	-
metaTimePOSIXonly	-
readMore1k	-
ossWriteHigher4g	-
writeLess1k	-
stripeSize	-
totalReadReq	-

(e) Min-max Mutual Information

Figure 4: The top 7 features selected from five different feature selection methods

5.1.5 Min-max mutual information. We also implement new feature selection method for our dataset. This new feature selection method selects features that can best represent the data in a different way from commonly used feature selection methods. We call this new method as Min-max mutual information. In this approach, features are selected in a way that selected features are less correlated to each other. To do so, we use data without applying one of the preprocessing steps: removing features having more than 0.8 correlation value with other features. From the dataset including all the features, the first feature which has highest correlation value with *writeRateTotal* is selected. After that, the second feature is selected from ten least correlated features with the first feature, having highest correlation value with the target among them. This process is repeated for the rest of the features.

5.2 Analysis of Feature Selection results

Figure 4 shows the top 7 scored features from the feature selection methods. Note that since Min-max mutual information is done with differently preprocessed dataset, selected features may not be included in other results. Of all the feature selection methods other than Mutual information regression, *totalFileSTDIO* scores the highest, which means *totalFileSTDIO* has the highest correlation value with target variable *writeRateTotal*. Since *totalFileSTDIO* is highly redundant to other features, it scores low on the Mutual information regression. The selected features and their scores are similar in Decision tree and Extra tree methods as they both gets their results by recursively splitting the data. *numProc* is a reasonable feature since the number of processors to run the job is highly related to parallelism, having direct impact on I/O throughput. *seqWritePct* is also a meaningful feature as sequential I/O operations improve performance by accessing storage devices sequentially.

We deploy Min-max mutual information method because of the dataset characteristic, which is that all the features are considerably correlated to target feature. Because of this attribute, the other four methods tend to select features which are highly correlated to each other. By using Min-max mutual information, we can select feature set with features less related to each other and also having strong relationships with target variable. However, it is still necessary to evaluate the feature selection results of all five methods. After the clustering results using Min-max mutual information are evaluated, we evaluate on all five feature selection methods to verify which feature set is actually the best for our dataset. The performance evaluation of clustering and feature selection methods will be discussed in the section 7.2.

6 APPLICATION OF CLUSTERING MODEL

By using the selected target-relevant features, clustering HPC applications can be done with less computation time and power. Also, reduced dimensionality can prevent overfitting clustering model. We evaluate three clustering models in this paper: KMeans from centroid-based methods, GMM from model-based clustering, and Ward linkage, one of the hierarchical clustering models. Clustering models run based on five different feature set results from feature selection methods, with the number of clusters ranges from 3 to 20. Total 15 clustering results are evaluated and discussed in Section 6.

6.1 Clustering Methods

6.1.1 KMeans Clustering. KMeans clustering algorithm repeatedly calculates distance and dissimilarity between two data to form cluster. Its main goal is to make dataset in same cluster to have highest inner similarity [8]. As KMeans algorithm is one of the centroid-based methods, it iteratively updates clusters based on centroids, which can be defined as means. This algorithm is simple, but has weakness in handling outliers, since mean value is highly sensitive to extreme values.

6.1.2 Gaussian Mixture Model. Model-based clustering makes an assumption that dataset comes from mixture of two or more underlying distributions. The Gaussian Mixture Model (GMM) is one of the probabilistic models used in model-based clustering [2][20]. GMM considers each data as mixture of multiple Gaussian distributions and automatically tries to find out which distributions the data originates from.

6.1.3 Ward Linkage Clustering. Ward linkage method is classified as agglomerative hierarchical clustering. Through the bottom-up approach, clusters each having single data value are iteratively merged together until the number of clusters reaches the specified value [6]. Similarity between two clusters is calculated by increase of Error Sum of Sqaures (ESS). If the calculated ESS is smallest among other values, two clusters are merged together since they have closest distance.

6.2 Cluster Validity Metrics

Good clustering tries to maximize the inter-cluster variance and minimize the inner-cluster variance. This can be rated by clustering validity index. In this paper, we evaluate the performance of clustering techniques using two validity metrics. First, Davies-Bouldin index (DBI) metric determines how well the clustering is done using the ratio between distribution within the cluster and distinctness among the clusters. The DBI score is decided by average of each cluster and other cluster's similarity and it ranges from 0 to 1. So if the DBI score is low, it means each cluster size is small and distance between the clusters is large, indicating better clustering. Second metric is Silhouette, which calculates Silhouette Coefficient (SC) for every data. SC is high when the inter-cluster distance is short and nearest-cluster distance is long. Since Silhouette score is mean SC of all the data, the higher the score means the better the cluster quality. Silhouette score ranges from -1 to 1.

Commonly, there is trade-off between compactness and distinctness when evaluating cluster performance. DBI and Silhouette can measure both compactness and distinctness of clusters [12], but comparing the result of both metrics can be difficult process. In order to reduce difficulty in comparing two metrics, we also use additional validity index, Combined Score, which combines two metrics together [24]. Combined Score can be defined as follows:

$$CombinedScore(x) = \begin{cases} \frac{Silhouette(x)}{DBI(x)}, & \text{if } DBI(x) \neq 0 \\ \text{undefined}, & \text{otherwise} \end{cases}$$

Since Combined Score is the result of Silhouette score divided by DBI score, the higher Combined Score means the better the clustering result is. Combined Score may not fully combine two metrics because it does not consider the different impacts two metrics have on Combined Score. However it is worth to compare two metrics and Combined Score together for better understanding of the clustering performance.

7 PERFORMANCE EVALUATION

To evaluate cluster performance, we calculate the Silhouette and DBI clustering validity scores from total 15 clustering results, which are the combinations of five feature selection methods and three clustering methods. On each combination, we change the number of clusters ranging from 3 to 20. The evaluation of the performance of clustering results is processed in two steps.

First, in order to find out the best clustering method, we compare the three clustering result from KMeans, GMM, and Ward linkage, which use the feature set of Min-max mutual information. After the best clustering method is chosen, we verify whether Min-max mutual information is actually the best feature selection method by comparing the performance score of five feature selection methods.

7.1 Selecting Best Clustering Method

Using the features from Min-max mutual information, we have to figure out the best clustering method among KMeans, GMM and Ward Linkage. Figure 5 shows the DBI, Silhouette and Combined Score of three different clustering methods. Higher Silhouette and Combined Score and lower DBI score indicate the better clustering quality. KMeans and Ward Linkage are very similar in cluster scores, while GMM shows slightly different trend compared to other two methods. For KMeans and Ward linkage, they both show the best clustering performance when the number of clusters is three, then the Silhouette score gets lower as the number of clusters increases

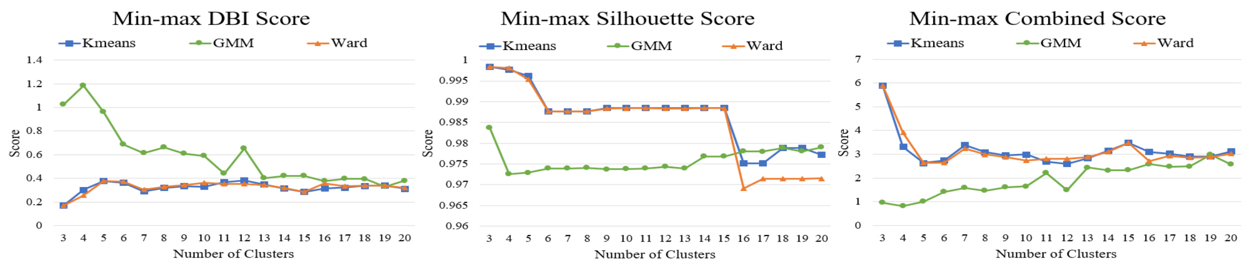

Figure 5: Cluster performance evaluation of clusters based on Min-max feature selection method

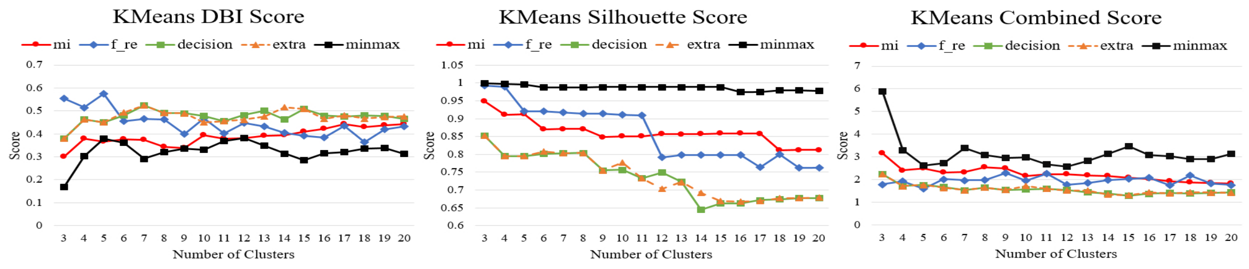

Figure 6: Cluster performance evaluation of clusters using KMeans clustering method

and the DBI score is stable regardless of the number of clusters. On the contrary, Silhouette score of GMM clustering drops at first and increase slightly as the number of clusters increases. Also, DBI score decreases steeply for GMM clustering.

In summary, as we increase the number of clusters, cluster performance of KMeans and Ward linkage tends to decrease, and performance of GMM gets better. KMeans and Ward linkage show the higher Combined Score than GMM on almost every case and it is hard to pick the best clustering method since they both show similar scores. Also, even though clustering by three scores highest, there are two elbow points of KMeans and Ward linkage from the Combined Score, which are the cases of when the number of clusters is 7 and 15. Since both clustering methods show similar trend, we can conclude that the best clustering method is KMeans clustering.

7.2 Feature Selection Methods Comparison

In order to figure out the impact the five feature selection methods have on KMeans clustering method, we evaluate the results of KMeans clustering which use five different feature sets. Figure 6 depicts the result of DBI, Silhouette scores and Combined Score, from the KMeans clustering result each using Mutual information, F-regression, Decision tree, Extra tree and Min-max mutual information. Except for the score of five clusters in DBI, Min-max mutual information scores best in every case on every performance validity metrics. Clearly, it can be concluded that the best feature selection method actually is Min-max mutual information.

8 CLUSTER CHARACTERIZATION

From the previous section, we have selected best feature selection method and best clustering method from the selected features using cluster performance evaluation metric. Combination of features chosen from Min-max mutual information and data clustered by KMeans shows best cluster performance in the all ranges of the number of clusters. Also, we can figure out that clustering by three scores highest from the Combined Score metric on Figure 5. Besides the best point with highest score, there are two elbow points shown in the figure, which are the cases with seven and fifteen clusters. In this section, we first characterize the clustering result of the three clusters. Each clusters is analyzed based on the data characteristics included in the cluster.

In order to figure out each cluster's tendency, we first average the feature values from all data belonging to each cluster. For instance, using clustering algorithm, our dataset is clustered into three clusters: cluster 1, 2 and 3. If we want to see how many processors the jobs included in cluster 1 are used, we can calculate it by averaging the *numProc* feature values from all data in cluster 1. In this way, average *numProc* on every cluster can be calculated, representing the *numProc* characteristics on the clusters. This averaging process is operated on every feature we used. Note that since we choose Min-max mutual information for our feature selection method, features with high correlation value with other features are not removed in the preprocessing step, which is total 46 features.

After we get the averaged values of every features, clusters can be characterized by determining the features that have dominant value on each cluster. So for every features, we select the cluster having the highest averaged feature value. Then, in order to assert that cluster has high tendency of feature, we figure out whether the cluster has more than twice as higher feature value than that of other features.

Figure 7 shows the cluster characterization results from KMeans which use the features from Min-max mutual information. The feature row with black box represents that the cluster has characteristic related to the feature. For example, since the box with

Features \ Cluster index	1	2	3
consecWritePct	▨		
ossWriteHigher4g	▨		
readLess1k	▨		
readLess1m	▨		
totalFileSTDIO	▨		
mdsOPSMin		▨	
ossReadHigher4g		▨	
ossWriteHigher1g		▨	
readMore1k		▨	
readMore1m		▨	
stripeSize		▨	
totalFile		▨	
totalFilePOSIX		▨	
writeMore1k		▨	
writeMore1m		▨	
metaTimePOSIXonly			▨
numProc			▨
readTimePOSIXonly			▨
totalFileMPIIO			▨
totalIOReq			▨
totalMetaReq			▨
totalOpenReq			▨
totalReadReq			▨
totalSeekReq			▨
totalStatReq			▨
totalWriteReq			▨
writeTimePOSIXonly			▨
consecReadPct			
mdsCPU95			
mdsCPUMean			
mdsOPS95			
mdsOPSMean			
numOST			
ossRead95			
ossReadHigher1g			
ossReadLargest			
ossReadMean			
ossWrite95			
ossWriteLargest			
ossWriteMean			
runTime			
seqReadPct			
seqWritePct			
writeLess1k			
writeLess1m			
writeRateTotal			

Figure 7: Cluster characterization result from Min-max mutual information and KMeans clustering algorithms

consecWritePct feature for `Cluster 1` is marked black, it means that the `Cluster 1` of KMeans includes jobs having high percentage of consecutive write operations. The feature with no mark, such as *consecReadPct* feature, represents that the averaged feature value of all three clusters have similar value. So in this case, we can not say which cluster has high characteristic of *consecReadPct*.

Figure 8 depicts the averaged value of three clusters on some of the features. Note that features like *writeLess1m* represent percentage of the number of operations with specific size in total number of operations. Also, the unit of *writeRateTotal* is MB/s. `Cluster 3` shows the highest write throughput, but the size of read and write operations is relatively small. From the feature values giving the information about the size and the number of open, seek or stat requests, we can assume that workloads in `Cluster 3` issue lots of metadata operations. This may harm the performance but as these workloads use large number of processors with MPI-IO API, `Cluster 3` gives the highest I/O throughput. Compared to `Cluster 3`, even though workloads in `Cluster 2` issue large size read and write operations, they show relatively slow performance since they rarely use MPI-IO for higher level of I/O parallelism and deploy relatively small number of processors. The workloads in `Cluster 1` use smallest number of processors and also issue large number of read and write operations in their runtime, resulting in lowest I/O throughput.

Overall, we can categorize the HPC applications with following characteristics when clustered by three:

(1) The workloads in `Cluster 1` tend to issue less than 1MB size read and write operations compared to the other clusters. In fact, many of the read and write operations operate on less 1KB blocks, and a significant amount of I/O operations are on stdio units, which are typically slow. Furthermore, the workloads in this cluster also use a few hundred MPI ranks, i.e., a small fraction of the overall system. Taking together, it is not surprising that the average I/O throughput for `Cluster 1` is only a few MB/s.
(2) The workloads in `Cluster 2` tend to issue more than 1MB size read and write operations. Especially, all the workloads issuing more than 1MB read operations are included in this cluster. Also, the total bytes written to the file is highest among the clusters, which also means there are lots of I/O operations during the processing time. These workloads are likely to use 8MB stripe size in average, which is 8 times more than the current default configuration, 1MB. The workloads use about 1000 MPI ranks each time and does not spend too much time in MPI IO. We suspect that using only a modest number of cores and not using MPI IO for parallel IO operations might be the reason for these workloads to achieve the highest IO throughput.
(3) Workloads in `Cluster 3` use more than 70,000 MPI ranks on average, which are the largest parallel jobs in our dataset. When compared to the other clusters, the workloads in this cluster use 62 times more processors on average. They also issue a large number of I/O requests, including metadata operations such as open, seek or stat requests. On average, each MPI rank issues nearly 500 open requests, which seems to be a high number. The block sizes of read and write operations are all less than 1MB, which is lower than those in `Cluster 2`. We postulate that the workloads in this cluster deploy lots of processors for higher parallelism to achieve the highest I/O throughput among the three clusters.

Features \ Cluster index	1	2	3
numProc	412	1140	71693
totalFileMPIIO	6.69	1.31	14.33
totalSeekReq	8037382	34852	103872479
totalStatReq	17116	1804	34260138
totalOpenReq	12778	6183	34558780
readLess1m	67.20	0	33.33
writeLess1m	49.13	0	33.33
readMore1m	0	100	0
writeMore1m	2.20	20.75	0
writeRateTotal	7844	47794	65226

Figure 8: Averaged feature values of each clusters

We also analyze the characteristics of the clusters when the number of clusters is seven and fifteen, since the Combined Score for cluster quality are at their local maximum at this cluster sizes. However, as we increase the number of clusters, the dataset is split in a way that small size clusters are divided into much smaller clusters. This leaves one big cluster including most of the dataset. Also, since there are too many clusters, the result shows that three out of seven clusters and nine out of fifteen clusters have no distinct characteristic. From our analysis, we can assume that increasing the number of clusters leads to inequality of cluster size which also results in multiple clusters with no specific aspects.

9 CONCLUSION

As the complexity of HPC cluster grows and more resources are managed in the supercomputer system, users are provided with different kinds of configuration settings. In order to simplify the use of system configuration, it is critical to categorize the HPC applications with proper standards. In this paper, we focused on categorizing four-month HPC workload data run on NERSC Cori supercomputer. After extracting several features from the log, we used several machine learning techniques, including feature selection and clustering methods. We used Mutual information regression, F-regression, Decision tree, Extra tree and Min-max mutual information to select features best representing the data. Then, using the selected features, we applied them to clustering models such as KMeans, GMM and Ward linkage to cluster the data. By evaluating the clustering performance with cluster validity metrics, we could select the best clustering method and feature selection method. Our result shows that using KMeans clustering method with features selected from the Min-max mutual information makes the best clustering result when clustered by three. Finally, we characterized each clusters by analyzing the included data. Using our observations, we can provide system designers better understanding of HPC applications running on the system, thus simplifying the configuration settings provided to users for better performance.

10 ACKNOWLEDGMENTS

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and also used resources of the National Energy Research Scientific Computing Center (NERSC). It was also supported by National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP)(NRF-2016M3C4A7952587), BK21 Plus(21A20151113068) for Pioneers in Innovative Computing (Dept. of Computer Science and Engineering, SNU), and by the Korea Institute of Science and Technology Information (Grant No.K-20-L02-C08).

REFERENCES

[1] 2019. Cori. https://www.nersc.gov/systems/cori/

[2] Jean Patrick Baudry, Adrian E. Raftery, Gilles Celeux, Kenneth Lo, and Raphaël Gottardo. 2010. Combining mixture components for clustering. *Journal of Computational and Graphical Statistics* 19, 2 (jun 2010), 332–353. https://doi.org/10.1198/jcgs.2010.08111

[3] Eugen Betke and Julian Kunkel. 2019. Footprinting Parallel I/O – Machine Learning to Classify Application's I/O Behavior. In *High Performance Computing*, Michèle Weiland, Guido Juckeland, Sadaf Alam, and Heike Jagode (Eds.). Springer International Publishing, Cham, 214–226.

[4] Renato L.F. Cunha, Eduardo R. Rodrigues, Leonardo P. Tizzei, and Marco A.S. Netto. 2017. Job placement advisor based on turnaround predictions for HPC hybrid clouds. *Future Generation Computer Systems* 67 (2017), 35 – 46. https://doi.org/10.1016/j.future.2016.08.010

[5] Christopher S. Daley, Prabhat, Sudip Dosanjh, and Nicholas J. Wright. 2017. Performance Analysis of Emerging Data Analytics and HPC Workloads. In *Proceedings of the 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS '17)*. Association for Computing Machinery, New York, NY, USA, 43–48. https://doi.org/10.1145/3149393.3149400

[6] Renato Cordeiro de Amorim. 2015. Feature Relevance in Ward's Hierarchical Clustering Using the Lp Norm. *Journal of Classification* 32, 1 (apr 2015), 46–62. https://doi.org/10.1007/s00357-015-9167-1

[7] Ilenia Fronza, Alberto Sillitti, Giancarlo Succi, Mikko Terho, and Jelena Vlasenko. 2013. Failure prediction based on log files using Random Indexing and Support Vector Machines. *Journal of Systems and Software* 86 (01 2013), 2–11. https://doi.org/10.1016/j.jss.2012.06.025

[8] Greg Hamerly and Charles Elkan. 2002. *Alternatives to the k-means algorithm that find better clusterings.* Technical Report.

[9] Olumuyiwa Ibidunmoye, Francisco Hernández-Rodriguez, and Erik Elmroth. 2015. Performance Anomaly Detection and Bottleneck Identification. *ACM Comput. Surv.* 48, 1, Article Article 4 (July 2015), 35 pages. https://doi.org/10.1145/2791120

[10] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2000. *An introduction to Statistical Learning.* Vol. 7. Springer, New York. 995–1039 pages. https://doi.org/10.1007/978-1-4614-7138-7 arXiv:arXiv:1011.1669v3

[11] Xu Ji, in Wuxi, Bin Yang, Tianyu Zhang, Xiaosong Ma, Xiupeng Zhu, Xiyang Wang, Nosayba El-Sayed, Jidong Zhai, Weiguo Liu, and Wei Xue. [n.d.]. *This paper is included in the Proceedings of the 17th USENIX Conference on File and Storage Technologies (FAST '19). Open access to the Proceedings of the 17th USENIX Conference on File and Storage Technologies (FAST '19) is sponsored by Automatic, Application-Aware I/O Forwarding Resource Allocation Automatic, Application-Aware I/O Forwarding Resource Allocation.* https://www.usenix.org/conference/fast19/presentation/ji

[12] Ling Jin, Doris Lee, Alex Sim, Sam Borgeson, Kesheng Wu, C Anna Spurlock, and Annika Todd. 2017. *Comparison of Clustering Techniques for Residential Energy Behavior using Smart Meter Data.* Technical Report. www.aaai.org

[13] Sunggon Kim, Alex Sim, Kesheng Wu, Suren Byna, Teng Wang, Yongseok Son, and Hyeonsang Eom. 2019. DCA-IO: A dynamic I/O control scheme for parallel and distributed file systems. In *Proceedings - 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2019.* Institute of Electrical and Electronics Engineers Inc., 351–360. https://doi.org/10.1109/CCGRID.2019.00049

[14] Zhiling Lan, Ziming Zheng, and Yawei Li. 2010. Toward Automated Anomaly Identification in Large-Scale Systems. *IEEE Trans. Parallel Distrib. Syst.* 21 (02 2010), 174–187. https://doi.org/10.1109/TPDS.2009.52

[15] Y. Liu, R. Gunasekaran, X. Ma, and S. S. Vazhkudai. 2016. Server-Side Log Data Analytics for I/O Workload Characterization and Coordination on Large Shared Storage Systems. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* 819–829.

[16] Huong Luu, Marianne Winslett, William Gropp, Robert Ross, Philip Carns, Kevin Harms, Mr Prabhat, Suren Byna, and Yushu Yao. 2015. A Multiplatform Study of I/O Behavior on Petascale Supercomputers. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '15).* Association for Computing Machinery, New York, NY, USA, 33–44. https://doi.org/10.1145/2749246.2749269

[17] Terai Masaaki, Kashiwaki Riku, and Shoji Fumiyoshi. 2017. *Workload Classification and Performance Analysis using Job Metrics in the K computer.* Technical Report 13. RIKEN Advanced Institute for Computational Science, Graduate School of Simulation Studies, Univeristy of Hyogo, RIKEN Advanced Institute for Computational Science.

[18] Asit K Mishra, Joseph L Hellerstein, Walfredo Cirne, and Chita R Das. [n.d.]. *Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters.* Technical Report.

[19] Asit K. Mishra, Joseph L. Hellerstein, Walfredo Cirne, and Chita R. Das. 2010. Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters. *SIGMETRICS Perform. Eval. Rev.* 37, 4 (March 2010), 34–41. https://doi.org/10.1145/1773394.1773400

[20] Kevin P. Murphy. 2012. *Machine learning : a probabilistic perspective.* MIT Press, Cambridge, Mass. 1067 pages.

[21] Gonzalo P. Rodrigo, P.-O. Östberg, Erik Elmroth, Katie Antypas, Richard Gerber, and Lavanya Ramakrishnan. 2018. Towards understanding HPC users and systems: A NERSC case study. *J. Parallel and Distrib. Comput.* 111 (2018), 206 – 221. https://doi.org/10.1016/j.jpdc.2017.09.002

[22] E. R. Rodrigues, R. L. F. Cunha, M. A. S. Netto, and M. Spriggs. 2016. Helping HPC Users Specify Job Memory Requirements via Machine Learning. In *2016 Third International Workshop on HPC User Support Tools (HUST).* 6–13.

[23] Carolin Strobl, James Malley, and Gerhard Tutz. 2009. An Introduction to Recursive Partitioning: Rationale, Application, and Characteristics of Classification and Regression Trees, Bagging, and Random Forests. *Psychological Methods* 14, 4 (dec 2009), 323–348. https://doi.org/10.1037/a0016973

[24] Wiebke Toussaint and Deshendran Moodley. 2019. *Comparison of clustering techniques for residential load profiles in South Africa.* Technical Report.

[25] Ozan Tuncer, Emre Ates, Yijia Zhang, Ata Turk, Jim Brandt, Vitus J. Leung, Manuel Egele, and Ayse K. Coskun. 2017. Diagnosing Performance Variations in HPC Applications Using Machine Learning. In *High Performance Computing*, Julian M. Kunkel, Rio Yokota, Pavan Balaji, and David Keyes (Eds.). Springer International Publishing, Cham, 355–373.

[26] Jorge R. Vergara and Pablo A. Estévez. 2014. A review of feature selection methods based on mutual information. , 175–186 pages. https://doi.org/10.1007/s00521-013-1368-0 arXiv:1509.07577