Parallel Variable Selection for Effective Performance Prediction

Jonathan Wang*†, Wucherl Yoo†, Alex Sim†, Peter Nugent*†, and Kesheng Wu†
*University of California, Berkeley
†Lawrence Berkeley National Laboratory
Emails: *jonathanwang017@berkeley.edu, †{wyoo, asim, penugent, kwu}@lbl.gov

***Abstract*—Large data analysis problems often involve a large number of variables, and the corresponding analysis algorithms may examine all variable combinations to find the optimal solution. For example, to model the time required to complete a scientific workflow, we need to consider the impact of dozens of parameters. To reduce the model building time and reduce the likelihood of overfitting, we look to variable selection methods to identify the critical variables for the performance model. In this work, we create a combination of variable selection and performance prediction methods that is as effective as the exhaustive search approach when the exhaustive search could be completed in a reasonable amount of time. To handle the cases where the exhaustive search is too time consuming, we develop the parallelized variable selection algorithm. Additionally, we develop a parallel grouping mechanism that further reduces the variable selection time by 70%.**

As a case study, we exercise the variable selection technique with the performance measurement data from the Palomar Transient Factory (PTF) workflow. The application scientists have determined that about 50 variables and parameters are important to the performance of the workflows. Our tests show that the Sequential Backward Selection algorithm is able to approximate the optimal subset relatively quickly. By reducing the number of variables used to build the model from 50 to 4, we are able to maintain the prediction quality while reducing the model building time by a factor of 6. Using the parallelization and grouping techniques we developed in this work, the variable selection process was reduced from over 18 hours to 15 minutes while ending up with the same variable subset.

I. Introduction

Science, including high-energy physics, cosmology, and biology, is increasingly reliant on large complex distributed workflows to create and analyze vast amounts of data [1], [2]. As both the computer hardware and the data processing software are becoming more complex, it is more challenging to understand the performance characteristics and to anticipate the resource requirements. The key part of this challenge is that there are too many variables that affect the overall performance, which increase the time needed to build a performance model, and make it easy to overfit the model. To handle this high-dimensional data, we explore a set of variable selection techniques to find an optimal subset of variables. These techniques eliminate variables that are not essential to the prediction model, thereby reducing the time needed to establish a performance model and improving the effectiveness of the model. As opposed to dimension reduction methods such as principal component analysis (PCA), variable selection preserves the meaning of variables, which is essential in understanding the effects of various parameters.

Variable selection has been an active research topic in pattern detection, machine learning, and statistics for many decades [3]–[5]. It has been shown to be effective both in theory and in practice for improving learning efficiency, increasing predictive accuracy, and reducing the complexity of prediction models on high-dimensional data. Because the computational complexities of model construction algorithms are high-order polynomial functions of the number of variables or even exponential functions, reducing the number of variables used can significantly reduce the model construction time. The presence of a large number of variables also makes it easy to overfit the models and introduce noise to the resulting predictions; therefore, removing these noisy variables can improve the overall prediction accuracy and simplify the models.

We use the Palomar Transient Factory (PTF) application as a case study, which processes large amounts of astronomy observations through a lengthy processing pipeline [6]. The PTF is a comprehensive transient detection system, which collects observations from survey cameras and reduces the data in real time through the National Energy Research Scientific Computing Center (NERSC) machines. This work was originally motivated by scientists' need to understand the occasional slowdowns in the data processing pipeline. Additionally, the scientists are also interested in understanding future resource requirements as the cameras produce higher resolution images at a faster rate. The scientists have instrumented their workflow to record the execution time of each stage of the workflow along with dozens of variables about the data objects being processed. Our prediction task is to use these variables plus the execution time of the first few steps of the pipeline to forecast the overall execution time of the entire workflow. This way we can understand the performance of the workflow without having to go through the entire pipeline. While we experimented primarily on the PTF dataset, we also applied our methods to TCP connection measurements data collected from ESNet data transfer nodes to validate the general applicability of our experiments.

Different tools and algorithms are available for our prediction task. However, the effective methods require many, if not all, possible combinations of variables involved, which makes the computational cost increase exponentially with the number of variables involved. In this work, we initially considered using a machine learning method known as Random Forest [7], [8], because it is known to be effective on large data sets and can be easily parallelized. However, building prediction models with a large number of variables can be very time consuming even when parallelized across multiple nodes.

The key goal of this work is to efficiently find a variable subset that can be used for accurate performance prediction. To better handle the large volume of data, we have experimented with several variable selection techniques to reduce execution time and remove noisy variables, as well as parallelization to improve the performance of variable selection.

Two classes of variable selection methods are generally used: filter and wrapper methods [4], [5]. Filter methods look at the inherent properties of the variables to determine important variables, while wrapper methods train models to perform selection. In this work, we choose to use Sequential Backward Selection, a wrapper approach that eliminates the least relevant variable at each iteration [5]. It is easy to understand and runs very quickly. Though there are cases where it might not find the optimal solutions, it is able to handle multi-collinearity slightly better than other sequential methods [9]. In our tests, it found the optimal variables identified by an exhaustive search through every possible variable combination.

We parallelized sequential selection methods to utilize a high performance computing platform. The implementation of parallel Sequential Backward Selection and the performance prediction model was based on our performance analysis tool (PATHA) [10] using Apache Spark™ as the back-end. By distributing computation to multiple compute nodes, many subsets were able to be selected and tested in parallel, improving the runtime of the selection process by a factor of 20 compared to the serial sequential selection. This parallelization improvement was maximized by the number of nodes utilized.

To improve the quality of the variables selected, we follow a strategy that includes variables highly correlated with the label but not inter-correlated with each other [11]. Specifically, we investigated methodologies to utilize variable correlations to handle large variable sets more efficiently. Using this idea as an inspiration, we developed a method to handle inter-correlated (potentially redundant) variables in order to improve the selection process while keeping the correlated (relevant) variables for performance prediction. Essentially, we developed a method to quickly eliminate redundant variables in parallel, significantly reducing the number of models to build as well as the size of the models. This filter method is combined with a wrapper selection technique, grouping variables by correlation and then using a prediction model to select the best variable in a group. This grouping step was parallelized by testing correlation groups simultaneously. This optimization reduced the selection time of the parallelized Sequential Backward Selection (SBS) by almost 70% compared to SBS without correlation preprocessing. While we gained significant improvement to SBS through parallelization, this preprocessing improved the serial runtime of SBS. The number of nodes selected for the parallelization maximized the runtime improvement of the correlation grouping.

An important quality measure of a variable selection procedure is the consistency of the subset selected. In other words, the selected variable subsets are expected to be similar across repeated experiments using the same data. This implies that variables are selected by their importance in the data, rather than variance in the selection process. High variance from the performance prediction model can result in inconsistent subset selection as our selection process is tightly coupled with the prediction model. To improve the consistency of the variable selection procedure, we needed to select a prediction model that maintains a consistent view of the importance of the variables. Among the models that we tested, we selected Gradient Boosting [12] because it has the lowest prediction variance with the PTF workflow measurement data.

In summary, the key contributions of our work are as follows:

- Studying the interaction of variable selection methods and prediction methods to select a good combination for the performance prediction task;
- Developing parallel Sequential Backward Selection method and improving its performance with correlation-based grouping;
- Integrating the variable selection mechanism into the performance analysis tool, PATHA [10];
- Evaluating our method on the the performance data from the PTF workflow [6].

The rest of the paper is organized as follows: Sec. II presents related work. Sec. III demonstrates the design and implementation of our variable selection mechanism. Sec. IV presents experimental evaluations and Sec. V presents discussion. The conclusion and future work are in Sec. VI.

II. Related Works

Our overall goal of simplifying performance models to reduce modeling overhead is similar to that of Snavely et al. [13] and Susukita et al. [14]. Susukita et al. [14] presented a performance prediction method combining performance modeling with macro-level simulations to reduce computation time. Similarly, Snavely et al. [13] presented a framework for performance modeling that reduces the number of parameters contributing to the model. This method allows an analysis of relevant parameters and reduces the computation time of creating the model. Our work also seeks to efficiently create performance models. However, unlike the work of Susukita et al. [14], our method is purely based on prediction modeling without simulations. In addition, we utilized a generalized method to determine relevant variables, unlike the work of Snavely et al. [13], which selected factors specifically for modeling large HPC systems. We used variable selection methods in conjunction with parallelization techniques to achieve the same goal of more efficient performance modeling.

Several researchers have discussed ideas about parallelized variable selection and correlation-based variable selection. Yu and Liu [5] proposed a filter based selection method of identifying relevant features based on symmetric uncertainty and selecting the predominant (non-redundant) features. Our method is similar in that it breaks the selection process into two steps to handle relevance and redundancy separately. However, we combine filter based preprocessing with a wrapper method, resulting in a variable set that is more tuned to the prediction model. We select features by training prediction models rather than based on the inherent properties of the variables. While this is more computationally expensive, the runtime can be offset by our parallelization approach. We also use a grouping approach similar to Lo et al. [3] and Song et al. [15] by grouping and selecting from correlated variables. However, unlike Song et al., we use Breadth First Search to group variables rather than a Minimum Spanning Tree [15]. In addition, we integrate this correlation-based grouping approach with Sequential Backward Selection to further reduce the variable set. Sequential Backward Selection was determined to be a relatively fast variable selection method, which was able to identify the key variables in the PTF data. As a result, we decided to use this method to handle the variables that were left after correlation grouping. Also, unlike Lo et al., we use Sequential Backward Selection with (non-linear) Gradient Boosting rather than linear regression [3] to process the variables after the correlation grouping step. Moreover, their focus was on modeling thermal error while our focus is on predicting the execution times of the PTF pipeline from the selected variables.

The idea of parallelized variable selection methods has also been explored by several researchers. The works of Zhao et al. [16] and Lopez et al. [17] introduced improved methods that utilize parallelization within the variable selection framework. Zhao et al. [16] explored a closed form solution to Sequential Forward Selection that could be solved iteratively on multiple worker nodes. We used Sequential Backward Selection as our selection method and selected variables empirically by training and testing prediction models using machine learning mechanisms. Lopez et al. [17] proposed using parallelized Scatter Search to select variables. On the other hand, our method implements parallelization in an existing selection method. Our method is similar to that of Zhou et al. [18] in the sense that we parallelize testing variable subsets. However, their approach did not define a subset scoring function, which we have defined as the Root Mean Squared Error (RMSE) of the prediction model.

Our work is inspired by ideas from previous research and combines them into a selection framework that shows good results for performance modeling using data from the PTF application pipeline. In future work, we will adopt more optimized methods presented in some of the related works.

III. Methods

The goal of our performance model is to predict the execution time of the Palomar Transient Factory (PTF) analysis pipeline steps. By using variable selection, we find a reduced set of variables from the original variable set that contains most if not all of the information of the full variable set. The advantages of using the selected subset for performance analysis are:

- Building simpler and more interpretable performance models from the fewer variables.
- Reducing computation and execution time for building performance models.
- Removing noisy variables having high variance or irrelevant information, which can improve the accuracy of the performance models.

In order to build a variable selection framework, we explored possible methods that can be applicable to the scientific applications. We used data from the PTF application as a case study. It is a wide-field automated survey that records images of transient objects in the sky [6]. Images from these cameras are sent to the supercomputing center for processing through an image subtraction data analysis pipeline. Each job consists of 10 tasks, and each task consists of 38 pipeline processing steps. Their execution times are measured and stored. We divided the 38 steps of the pipeline into two groups: 1) the first 16 pipeline steps, that are quickly executed, for conducting initialization and pre-processing and 2) the remaining 22 pipeline steps. The execution of the first group provides useful information, such as sky conditions and image conditions, which are suspected to affect the performance of the second group. These features, along with the execution time of the first group of steps, make up the 50 variables with which we build our prediction model. We then predict the execution time of the second group, which takes up most of the overall execution and computation time.

In our initial exploration of the PTF workflow measurement data, we found out that many variables are inter-correlated. By plotting the correlation matrix of the variables in Figure 1, we confirmed that many variables are highly correlated, if not exactly correlated (either positively or negatively). Based on this information, we considered methods to most effectively handle the redundant data.

We tested a couple of machine learning methods to achieve the most consistent variable selection. Initially, we used Random Forest for the performance prediction. However, after observing high variability in the prediction results, we considered an alternative machine learning method, Gradient Boosting Regression. To compare these two methods, we built multiple models using the same data and examined the variance in the predictions for each method. We used implementations of these methods from the Python scikit-learn library [19]. For Random Forest, we used 100 trees, and for Gradient Boosting, we had 300 iterations. These parameters were points where the model stopped improving significantly. The parameters were also bounded by reasonable computation time. Based on these consistency results, we selected Gradient Boosting as our prediction model since we require consistent variable selection results and the lower prediction error could be achieved after the variables are selected.

After data exploration and model selection, we established a testing baseline using exhaustive variable selection, which identifies the optimal subset, and tested several standard

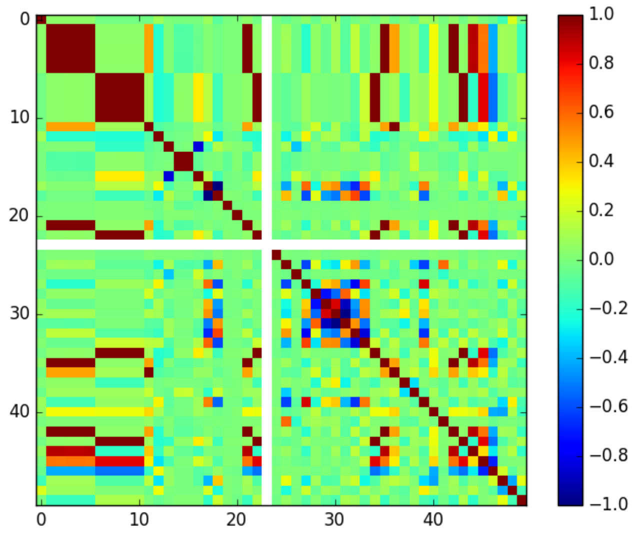

Fig. 1. Correlations between variables

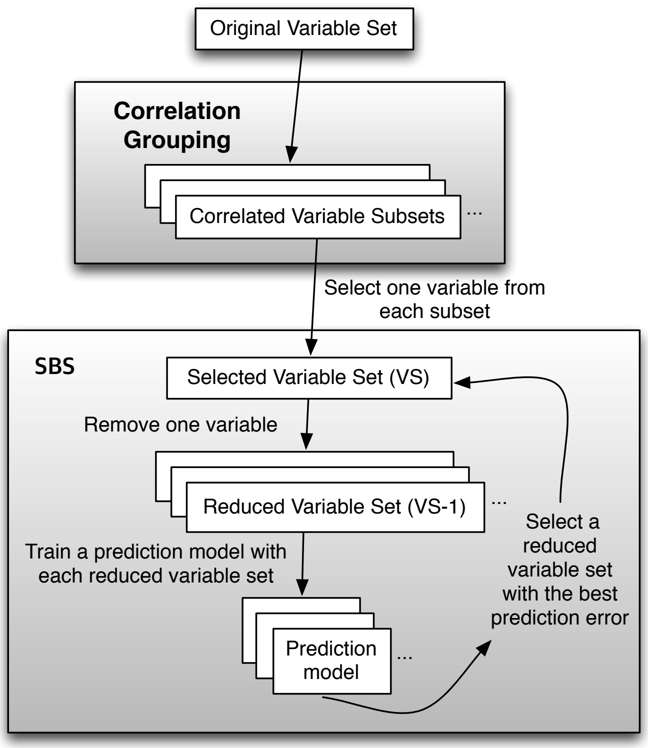
Original Variable Set
Correlation Grouping
Correlated Variable Subsets
...
Select one variable from each subset
SBS
Selected Variable Set (VS)
Remove one variable
Reduced Variable Set (VS-1)
...
Train a prediction model with each reduced variable set
Prediction model
...
Select a reduced variable set with the best prediction error

Fig. 2. The overview of variable selection process

statistical and machine learning variable selection methods, including Recursive Feature Elimination, Univariate F-Test, and Gini Feature Importance. We were only able to run exhaustive selection on smaller variable sets or for smaller subsets for the exponential computation time of the selection to finish within reasonable time. Even with parallelization, we were unable to run the full exhaustive selection on 50 variables in reasonable time.

We ran our tests using two data sets, a training set to build models and a test set to perform variable selection. These data sets were fixed to maximize consistency across tests. We used the performance data collected from Mar. 1, 2016 to Apr. 31, 2016 (PDT) as our training set and the measurements from May 1, 2016 to May 31, 2016 (PDT) as the test set.

As a greedy selection approach, we implemented Sequential Backward Selection (SBS), which starts with the full set of variables and removes one variable at each iteration. For each variable in the variable set, we created a reduced set without that variable, and trained Gradient Boosting prediction models on the training data using the variable subsets. We then used the models to predict on the test set and measured the prediction accuracy using Root Mean Squared Error (RMSE). We selected the subset with the best prediction accuracy to determine the best variable to remove. Apache Spark was used to parallelize subset testing at each iteration. We used 3 nodes on the HPC system, with each node having 24 CPU cores. This allowed us to designate one core for each variable, so we could test all subsets simultaneously, maximizing the improvement of parallelization.

As shown in Figure 2, we combined multiple methods to efficiently select variables that can accurately predict performance. We wanted to take advantage of multiple correlated variables to further improve the performance of the standard SBS. Since redundant variables do not contribute to the prediction accuracy, we integrated correlation-based grouping into the SBS to eliminate them quickly. We grouped variables that were correlated with each other beyond a certain threshold by searching through variables using Breadth First Search, so any two variables with correlation coefficient above the correlation threshold would be grouped together. This correlation threshold was determined experimentally to be 0.8, based on the maximum improvement in runtime in our experiments. Once the variables were grouped, we trained a prediction model for each group and selected the single best variable from the group, based on the Gini Importance of each variable in the model, which is derived from improvement in RMSE. This correlation grouping step was also parallelized across the correlation groups. For this parallelization, the number of cores utilized was equal to the number of correlation groups, which was bounded by the number of variables in the set, so we also used 3 nodes of 24 cores each. Each core processed one correlation group, selecting the best variable in the group. The variables selected from each group were then reduced using the SBS. This method handled redundant variables quickly in parallel and improved the serial runtime of SBS by reducing the number of iterations of selection and reducing the size of models built at each iteration. Figures 3 and 4 break down the parallelization processes for correlation grouping and SBS respectively.

In order to evaluate whether there exists overfitting or generalization errors in the variable selection, we split our 1-month data from May 1, 2016 to May 31, 2016 (PDT) into two equally sized, randomly selected sets, a validation set and

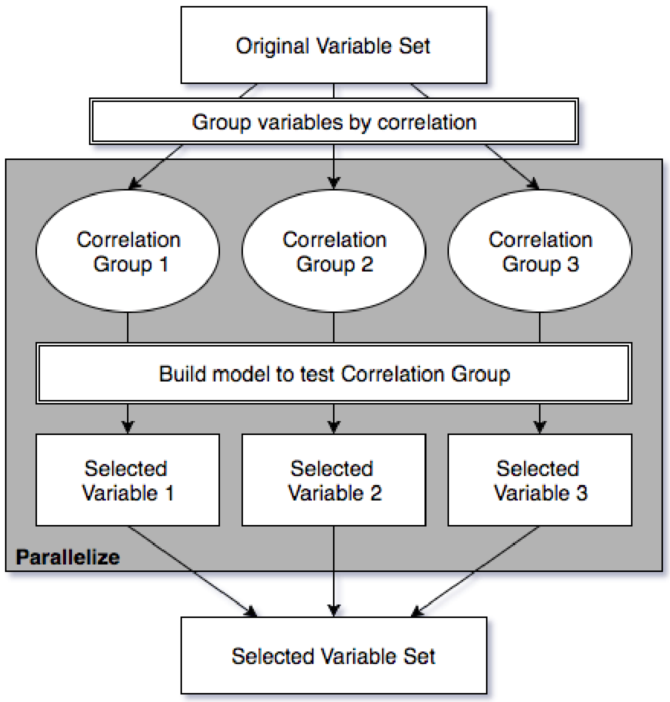

Fig. 3. Parallelization of correlation groups

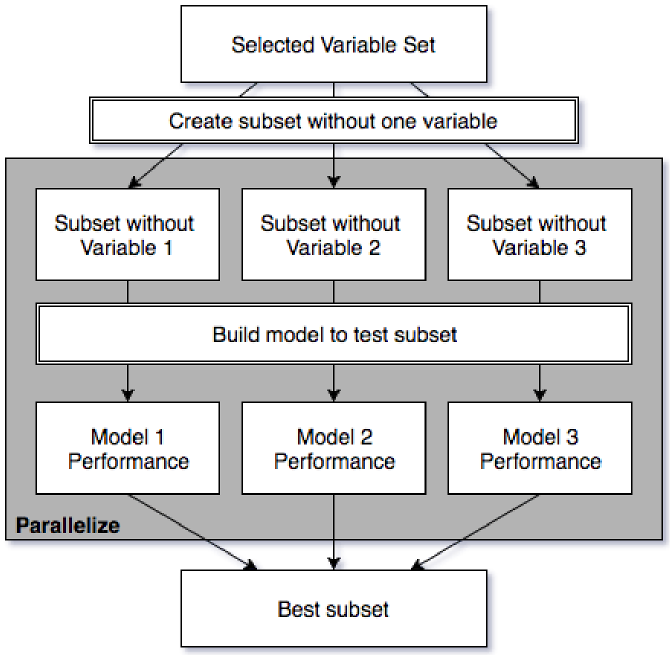
Selected Variable Set
Create subset without one variable
Subset without Variable 1
Subset without Variable 2
Subset without Variable 3
Build model to test subset
Model 1 Performance
Model 2 Performance
Model 3 Performance
Parallelize
Best subset

Fig. 4. Parallelization of one iteration of SBS

a test set. The validation set was used to test the prediction models in order to select the variable subset with the lowest prediction error and determine which variable to remove. After determining a subset of variables using variable selection, we tested those variables by building a prediction model on the test set using the selected variables and comparing the prediction results to the results of the prediction model within the variable selection mechanism.

IV. Experimental Results

A. *Experimental Setup*

The experiments were conducted on the Lawrencium HPC system at Lawrence Berkeley National Laboratory (LBNL). We used 56 nodes, and each node contains two 12-core Intel® Xeon® E5-2670 CPUs and 64GB memory. We used the measurement logs of the PTF application collected on the NERSC Cori Cray XC40 supercomputer from Mar. 1, 2016 to May. 31, 2016 (PDT). The comparison of different machine learning and variable selection methods as well as the implementation of parallel Sequential Backward Selection and the performance prediction model were based on our performance analysis tool (PATHA) [10] using Apache Spark™ as the back-end.

B. *Variable Selection*

The variable selection procedure is tightly coupled with the choice of model for performance analysis, and the selected variables can be easily applicable to general performance analysis, except for the case of using all the variables for strict evaluation. Nevertheless, it is important to evaluate whether the selected variables include representative and essential information that can produce performance analysis with quality comparable to that of the full variable set. In addition, it is important to produce consistent selection without losing key variables in the selection process as the process is iterative with multiple steps. In this sense, the selection of the performance model can impact the quality of the selected set since the error from the performance model can degrade the selection process by removing key elements due to the high variance or low accuracy from the overestimated error of the model.

In our experiment, we focused on the performance prediction of the execution time of the PTF analysis pipeline steps. As explained in Sec. III, we divided the 38 pipeline steps into two groups: 1) the first 16 steps 2) and the remaining 22 steps. We predict the execution time of the second group using 50 variables, including the execution time of the quickly executed first group. In the 3-month data, the execution time of the first group took 8.9% out of entire execution: 16.5 seconds for the first group and 158.1 seconds for the second group on average.

The first step of our analysis was to select a model for performance prediction that can make predictions with low variance and high accuracy. Figure 5 shows that Gradient Boosting is a better model than Random Forest for our purposes despite the higher bias. While Random Forest showed better accuracy that Gradient Boosting, it had significantly higher variance, which led to inconsistent subset selection. In order for useful variable selection, we require consistent improvement in prediction accuracy as variables are removed, which is not seen with the Random Forest model. On the other hand, Gradient Boosting showed consistent reduction of prediction error by removing noisy or redundant variables until selecting the final 4 variables.

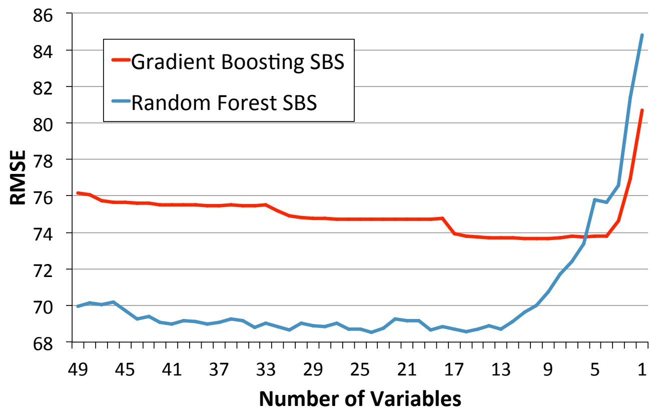

Fig. 5. Prediction error as the subset size is reduced for Gradient Boosting and Random Forest models

We investigated and compared Recursive Feature Elimination (RFE), Univariate F-Test (Univariate), and Gini Feature Importance (Importance) as methods of selecting variables. They are compared with the prediction error from the selected variables using sequential selection. Figure 6 shows the results of these tests compared to the optimal subset that was created by exhaustive search with all the possible combinations of variables. None of the existing implementations of the methods approximated exhaustive variable selection very closely, and there was no consistent improvement as the subset was reduced. However, Figure 6 also shows that both Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) perform significantly better and more consistently than the other methods. Parallelizing SBS across 50 workers resulted in a significant runtime improvement from about 18 hours (65020 seconds) to less than an hour (2727 seconds). For this experiment, we used a smaller set of 20 variables selected with the domain knowledge from the scientists, accounting for the higher level of prediction error than in the other figures.

Figure 7 examines SBS more carefully on a smaller 10-variable set where we could easily run exhaustive selection. The graph shows that for all subset sizes, SBS got the same prediction error as the optimal subset. We were unable to test exhaustive selection on the full 50-variable dataset due to the exponential time cost. The prediction error was slightly higher than that of SBS on the 50-variable set due to the random selection of the 10 variables. However, the smaller variable set tests demonstrated that the performance of SBS was similar to the exhaustive selection.

Figure 8 shows the results of testing SBS with the full variable set. Due to the exponential computation time, exhaustive selection could not be completed on the full variable set, only on the large and small subsets, where less computation is re-

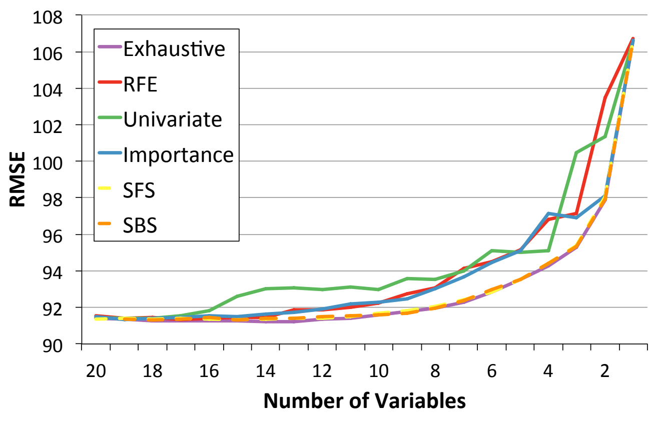

Fig. 6. Prediction error relative to subset size for standard methods (higher RMSE due to different initial variable subset)

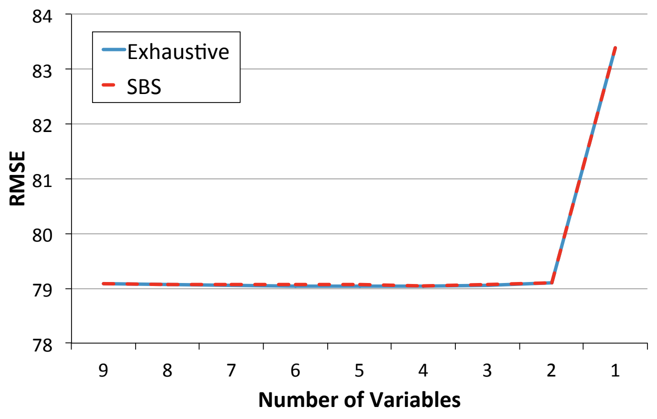

Fig. 7. Prediction error on a small variable set using exhaustive selection and SBS

quired. However, there is a very visible trend as variables were removed from the variable set. Through variable selection, noisy variables were removed from the subset, improving the prediction accuracy. Redundant variables were also removed, although they do not have a significant impact on the error.

The decreases in error in Figure 8 represent noisy variables that are removed while the flat segments represent that redundant variables are removed from the selection. We observed the optimal subset size of this dataset to be about 4, which shows the minimal error from the smallest subset. After the optimal subset was selected, the error grew rapidly due to the key variables being removed from the set, and thus there was not enough information in the data to make an accurate prediction. Despite some deviation in the sequence of selected variables, SBS converges to the same optimal subset as the exhaustive selection, as shown in Table I.

Figure 9 illustrates the rapid decrease in the training time relative to the loss in the prediction accuracy. When the size of the subset was reduced by the variable selection, little or no loss in the prediction accuracy was shown until the subset reached extremely small sizes. However, the training

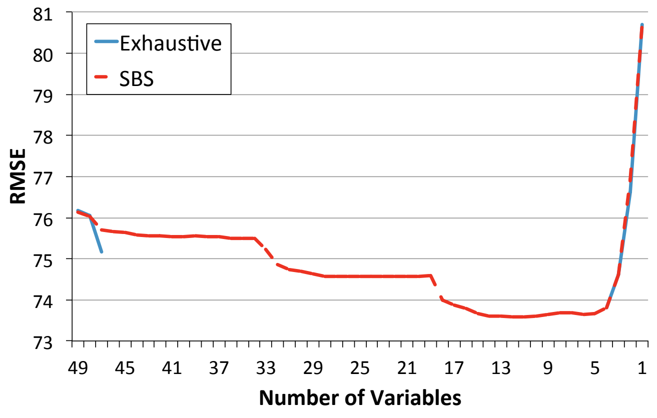

Fig. 8. Prediction error relative to subset size for SBS

TABLE I
VARIABLE SUBSETS SELECTED BY EXHAUSTIVE SELECTION AND SBS

Variables	Optimal	the SBS
1	{44}	{44}
2	{27, ***47***}	{27, ***44***}
3	{0, 27, 44}	{0, 27, 44}
4	{0, 27, 44, 47}	{0, 27, 44, 47}

time was decreased significantly as the number of variables was decreased. For the Gradient Boosting model, the training time decreased almost linearly to the subset size. While the time required to train a prediction model on the full 50 variables was about 90 seconds, the model training time with the selected 4 variables was about 15 seconds. These results show the achievement of our primary goal of reducing the model training time without losing the accuracy.

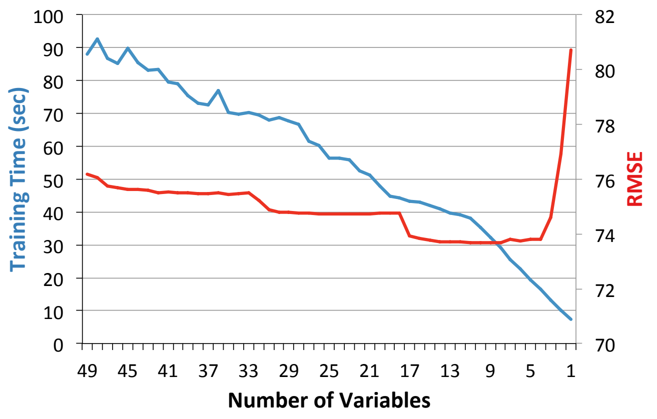

Fig. 9. Trends in prediction error and training time relative to subset size

C. Correlation Grouping

As scientific data can have inter-correlated variables, we investigated methods of using correlation to improve the variable selection process. We evaluated the correlation-based feature selection (CFS) [11] with Pearson and Spearman correlations. Figure 10 shows that CFS did not perform well on this dataset, specifically in the large spike near the middle. The poor performance of CFS can be attributed to the high level of correlation between variables, which is a detrimental factor in CFS. We tested CFS with multiple correlation metrics with similar results. In addition, CFS used an indirect heuristic which contributed to the higher prediction error.

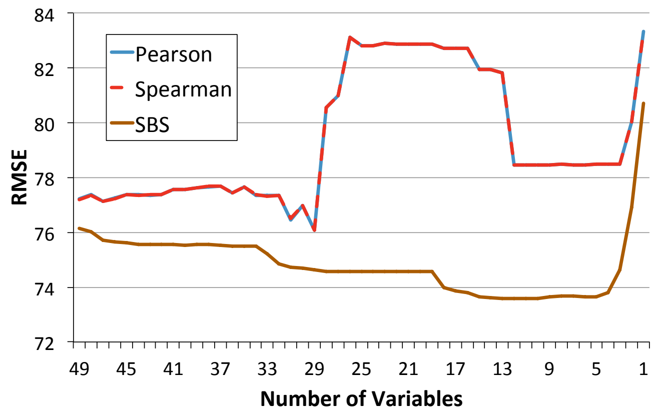

Fig. 10. Prediction error as subset size is reduced using CFS methods and SBS

From the previous experiments, we determined that SBS could achieve similar results to the optimal subset from the exhaustive selection, and be improved with parallelization. In order to improve the serial runtime of SBS, we investigated methods to use inter-correlated variables. We developed correlation-based grouping that pre-processes the variables by grouping variables correlated beyond a certain threshold and selecting a single variable from each group. Our experiments showed that this method still returned results comparable to SBS, as the variables selected using this method were the same as those selected by SBS. Figure 11 shows the selection process for both methods. The changes in error follow similar patterns, with the plot for correlation grouping being more condensed. For this dataset, correlation grouping removed roughly half of the iterations, which is significant since the earlier iterations are much more computationally expensive to train prediction models on with larger variable subsets.

To evaluate the runtime improvement of correlation grouping, we conducted the variable selection down to 1 variable using different thresholds of variable correlation for grouping. As shown in Figure 12, at the correlation threshold of 0.8, the grouped SBS took only 888 seconds to run as opposed to 2727 seconds for the regular SBS. This represented a 70% runtime improvement at no cost to the selection as the prediction accuracy from the correlation grouping was similar to that of SBS. While we achieved results comparable to the SBS even at the correlation threshold of 0.5, the time improvement was most significant at the threshold of 0.8.

As described in Sec. III, we divided our data to check for overfitting in the selected subset. The training error in Figure 13 is the RMSE of the predictions using the validation set. The test error in Figure 13 is the result of using the

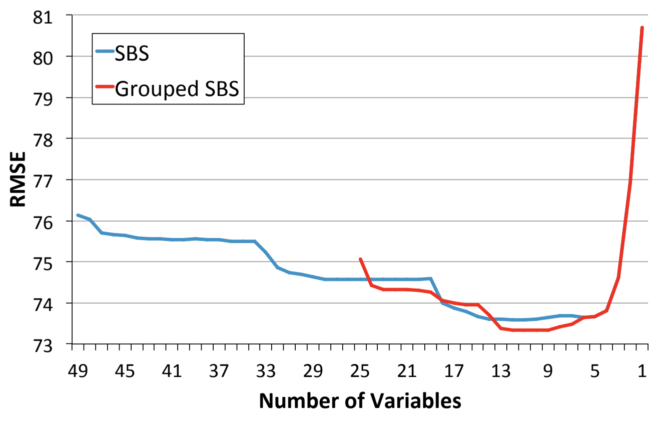

Fig. 11. Prediction error relative to subset size for correlation grouping and SBS

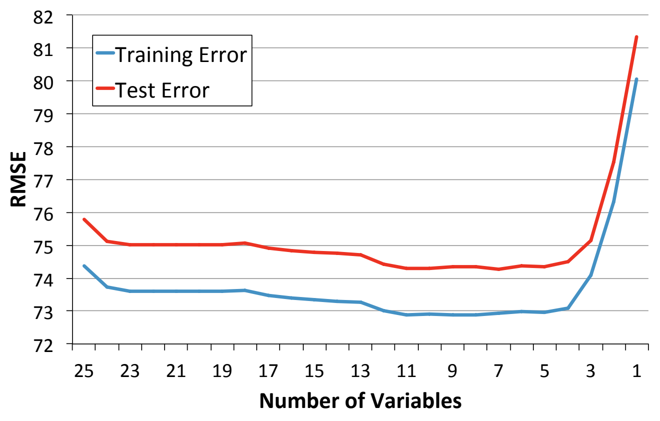

Fig. 13. Trends in training and test error relative to subset size

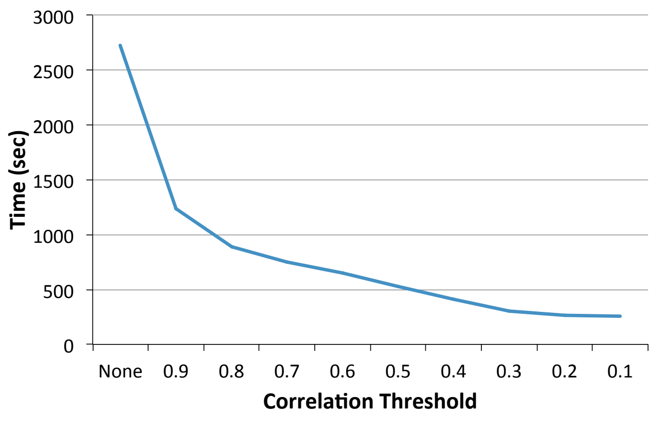

Fig. 12. Selection runtime for different correlation coefficients

selected subset on the test set. For this test, we checked the training error of the variable subset at each iteration and used the same subset with different data to check the test error by building and predicting with a separate prediction model built from the test data. While this does not check for overfitting by the prediction model, it checks for overfitting by the variable selection mechanism, which is what we are concerned about. Although the error slightly varies between the two sets due to the random sampling, Figure 13 shows that the trends of variable selection are the same. This implies that the improvement of removing a variable is consistent between the two data sets, showing that the selected variables were not overfit to the training data. The removed variable at each iteration has the same effect on the performance model regardless of the data that the model is trained on, demonstrating the generalizability of the selected subset.

V. Discussion

Based on our evaluation results, it is evident that a subset of variables can be identified to improve the prediction model by reducing the model construction time and increasing prediction accuracy. Unlike certain dimensionality reduction techniques like PCA, the variables selected by variable selection keep their original meaning, allowing us to investigate the impact of specific variables on the model. Since the variable selection process is closely associated with the model construction process, we have to explore a number of different combinations before we finally settle on a combination that can give consistent answers.

Once we identified Sequential Backward Selection (SBS) as the viable selection method, we considered options to utilize high performance computing to perform selection more quickly. Although Sequential Forward Selection (SFS) also performed relatively well compared to existing methods, it did not properly handle correlated variables compared with SBS. Since SFS builds up a variable set by adding variables instead of removing variables, it can end up selecting multiple redundant (inter-correlated) variables.

The sequential selection process is iterative, thus we could not parallelize the selection process across the subset sizes, i.e. the variable selection of subset size n is dependent on the selection of subset size $n-1$, so we cannot independently parallelize the selection of subset size n and that of $n-1$. Therefore, we parallelized the selection process across the subsets at each iteration, using CPU cores equal to the size of the initial variable set size. This decision also maximized the uniformity of each parallelized job since each job trains a model with the same number of variables. Nevertheless, there exists some imbalance in the execution time of each job, even with the same size variable subset, known as the straggler problem, i.e. the entire selection finishes when the longest job finishes. In addition, due to the removal of variables at each iteration, an increasing number of CPU cores became unused after each selection. In our experiments, with relatively small variable size (around 50), this imbalance showed little impact. We will leave the straggler problem and imbalanced resource usage for future work with experiments on larger variable sets with more limited resources.

The following is a breakdown of the runtime improvements on SBS. Building a gradient boosting model takes $O(kn)$ time where k is the number of iterations and n is the number of variables. SBS runs in $O(kn^3)$ time for n iterations and n models at each iteration. Parallelizing SBS reduces this runtime to $O(kn^2)$ since we parallelize the n models at each iteration. Correlation grouping runs in several steps - calculating the correlation matrix, grouping the correlated variables, and selecting the relevant variables. The correlation matrix can be calculated in $O(n^2)$ time or $O(n)$ time in parallel. Grouping variables takes $O(1)$ time in the best case of 1 cluster and $O(n^2)$ time in the worst case of n clusters. Lastly, selecting the relevant variables takes $O(\hat{n})$ time where $\hat{n}$ is the size of the largest correlation group. After the preprocessing, n' variables are remaining, so parallelized SBS takes $O(kn'^2)$ time. $O(kn) >> O(n^2)$ due to the number of iterations required for Gradient Boosting. In our experiments, the runtime of the preprocessing steps was insignificant compared to the actual selection process. For grouped SBS, $O(kn'^2)$ is the dominant runtime term in our PTF test case, and $n' << n$ due to the number of correlation groups, representing a significant reduction in the number of variables, so grouped SBS shows significant runtime improvement. In general, we want to balance n' and $\hat{n}$ to balance runtime and grouping accuracy.

The performance of grouped SBS is then dependent on the size of the largest cluster, preferring multiple medium sized clusters. If there are many small clusters with one large cluster, the parallelized cores would not be utilized efficiently, and the large cluster could cause a loss in accuracy by eliminating relevant variables. However, if there are only small clusters, the grouping step would not have a large impact on performance improvement. Without a fair number of clusters, the size of the preferred subset would not be able to be specified due to too few remaining variables after the grouping step. All of these trade-offs can be balanced by the selection of the correlation threshold. In our experiments, with the correlation threshold of 0.8 and the highly correlated dataset, this method was able to remove many redundant variables without eliminating any of the desired variables. Especially since training models on larger variable sets is more computationally expensive, this initial reduction offers a significant performance improvement for the entire variable selection process. In short, our method of combining correlation grouping and SBS was able to speed up SBS at little cost.

Our variable selection method can be generally applicable to datasets from other applications. SBS, which is the variable selection procedure at the core of our method, is applicable to all sorts of data, and our parallelization and pre-processing techniques are generally applicable as well although the amount of improvement will vary. Our tests to check for overfitting demonstrate that optimizing the training model within the variable selection mechanism optimizes the prediction model built using the final variable subset. Since the variables are selected based on their importance to the data, this variable selection method can be combined with various prediction models to suit the dataset being tested.

In order to validate this claim, we applied our method to a different dataset, consisting of TCP connection measurements collected from ESNet data transfer nodes. These measurements include 65 variables such as transfer size, duration, and retransmitted size. We used the throughput (transfer size divided by duration) as the target variable of the prediction model. From our preliminary results, the variable selection was able to select transfer size and duration in the selected subset. By removing these variables strongly coupled with the throughput, we were also able to select packet size that is strongly correlated with the transfer size. Compared with the PTF dataset, proportionally more variables are correlated with each other. Due to this difference, keeping only one variable from the correlated variable set made prediction result less accurate. While the initial result is promising, we plan to improve the correlation-based selection when the number of correlated variables is large.

VI. Conclusion

Many large data analysis tasks involve a large number of parameters, i.e., high-dimensional data, and variable selection methods are effective in reducing the number of variables needed for these analysis tasks. In this work, we considered how variable selection techniques can be employed to reduce the model building time in a prediction task. More specifically, we studied building a performance prediction model for an astronomy workflow from the PTF project. We showed that reducing the number of variables also reduced the model training time and even slightly reduced the prediction error. We observed that the optimal subset size of this dataset for the performance model was quite small, indicating that a handful of variables are critical to the performance prediction task, while the remaining variables are either disruptive or redundant. Sequential Backward Selection (SBS) was shown to be an effective variable selection method as it found a subset comparable to exhaustive selection in significantly shorter time. Although there were some ordering differences from the exhaustive selection due to the single direction nature of the sequential selection, this drawback did not have a significant impact in our experiment. Overall, we observe that SBS is a highly effective variable selection method that is easy to parallelize.

We have developed a framework to quickly select variables from the PTF analysis pipeline measurement data to optimize the prediction accuracy. Due to the high levels of correlation among variables in the dataset, certain variable selection methods performed poorly. However, our correlation-based grouping method was able to further improve the performance of SBS on the PTF measurement data. By clustering correlated variables, we reduce the number of iterations of the sequential selection and condense the computationally expensive models into much simpler computations. The performance of our method is more efficient with the existence of inter-correlation among variables, and was applicable to the PTF measurement data due to the high levels of correlation among the variables. In this experiment, it was able to identify the same subset as the SBS in just one-third of the computation time. Since we worked with a parallelized version of a generic variable selection method, these improvements should be applicable to other application datasets, although more improvement would be seen in correlated data.

By taking advantage of high performance computing resources and variable correlations, we were able to select a variable subset that can result in accurate performance prediction within significantly shorter computation time than that of available implemented methods. Future work includes further testing on other datasets to evaluate the applicability of our framework, better quantifying the improvement from the parallelization and optimization, and understanding the effects of variable correlations. We will also look into further optimizing both the grouping and selection steps of our framework.

In summary, with our improvement in the variable selection process for the PTF application, we are able to reduce the number of variables in the prediction model from 50 to 4, reducing model building time by a factor of 6 while maintaining the accuracy of the prediction model. Moreover, by parallelizing SBS and clustering correlated variables, we reduce the time needed to identify the optimal subset from over 18 hours to 15 minutes. Both the prediction model and variable selection improvements came at minimal cost to the model prediction quality.

Acknowledgment

This work was supported by the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231 and used resources of the National Energy Research Scientific Computing Center.

References

[1] T. Hey, S. Tansley, and K. Tolle, Eds., *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft, Oct. 2009.

[2] A. Shoshani and D. Rotem, Eds., *Scientific Data Management: Challenges, Technology, and Deployment*. Chapman & Hall/CRC Press, 2010.

[3] C.-H. Lo, J. Yuan, and J. Ni, “Optimal temperature variable selection by grouping approach for thermal error modeling and compensation,” *International Journal of Machine Tools & Manufacture*, vol. 39, pp. 1383–1396, Sep. 1999.

[4] S. Maldonado and R. Weber, “A wrapper method for feature selection using support vector machines,” *Information Sciences*, vol. 179, no. 13, pp. 2208–2217, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025509000917

[5] L. Yu and H. Liu, “Efficient feature selection via analysis of relevance and redundancy,” *Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004.

[6] N. M. Law, S. R. Kulkarni, R. G. Dekany, E. O. Ofek, R. M. Quimby, P. E. Nugent, J. Surace, C. C. Grillmair, J. S. Bloom, M. M. Kasliwal, L. Bildsten, T. Brown, S. B. Cenko, D. Ciardi, E. Croner, S. G. Djorgovski, J. v. Eyken, A. V. Filippenko, D. B. Fox, A. Gal-Yam, D. Hale, N. Hamam, G. Helou, J. Henning, D. A. Howell, J. Jacobsen, R. Laher, S. Mattingly, D. McKenna, A. Pickles, D. Poznanski, G. Rahmer, A. Rau, W. Rosing, M. Shara, R. Smith, D. Starr, M. Sullivan, V. Velur, R. Walters, and J. Zolkower, “The palomar transient factory: System overview, performance, and first results,” *Publications of the Astronomical Society of the Pacific*, vol. 121, no. 886, pp. 1395–1408, 2009.

[7] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, “Random forest: a classification and regression tool for compound classification and qsar modeling,” *Journal of chemical information and computer sciences*, vol. 43, no. 6, pp. 1947–1958, 2003.

[8] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[9] L. Xu and W.-J. Zhang, “Comparison of different methods for variable selection,” *Analytica Chimica Acta*, vol. 446, pp. 477–483, Nov. 2001.

[10] W. Yoo, M. Koo, Y. Cao, A. Sim, P. Nugent, and K. Wu, “PATHA: Performance analysis tool for hpc applications,” in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2015, pp. 1–8.

[11] M. A. Hall, “Correlation-based feature selection for machine learning,” Ph.D. dissertation, University of Waikato, Apr. 1999.

[12] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001.

[13] A. Snavely, L. Carrington, N. Wolter, J. Labarta, R. Badia, and A. Purkayastha, “A framework for performance modeling and prediction,” in *Supercomputing, ACM/IEEE 2002 Conference*. IEEE, 2002, pp. 1–17.

[14] R. Susukita, H. Ando, M. Aoyagi, H. Honda, Y. Inadomi, K. Inoue, S. Ishizuki, Y. Kimura, H. Komatsu, M. Kurokawa *et al.*, “Performance prediction of large-scale parallell system and application using macro-level simulation,” in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 20.

[15] Q. Song, J. Ni, and G. Wang, “A fast clustering-based feature subset selection algorithm for high-dimensional data,” *IEEE transactions on knowledge and data engineering*, vol. 25, no. 1, pp. 1–14, 2013.

[16] Z. Zhao, R. Zhang, J. Cox, D. Duling, and W. Sarle, “Massively parallel feature selection: an approach based on variance preservation,” *Machine Learning*, vol. 92, pp. 195–220, Jul. 2013.

[17] F. G. Lopez, M. G. Torres, B. M. Batista, J. A. M. Perez, and J. M. Moreno-Vega, “Solving feature subset selection problem by a parallel scatter search,” *European Journal of Operational Research*, vol. 169, pp. 477–489, Mar. 2006.

[18] Y. Zhou, U. Porwal, C. Zhang, H. Ngo, X. Nguyen, C. Re, and V. Govindaraju, “Parallel feature selection inspired by group testing,” *Advances in Neural Information Processing Systems*, vol. 27, pp. 3554–3562, 2014.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Oct. 2011.