Machine Learning Based Job Status Prediction in Scientific Clusters

Wucherl Yoo, Alex Sim, Kesheng Wu
Lawrence Berkeley National Laboratory, Berkeley, CA, USA

***Abstract*—Large high-performance computing systems are built with increasing number of components with more CPU cores, more memory, and more storage space. At the same time, scientific applications have been growing in complexity. Together, they are leading to more frequent unsuccessful job statuses on HPC systems. From measured job statuses, 23.4% of CPU time was spent to the unsuccessful jobs. We set out to study whether these unsuccessful job statuses could be anticipated from known job characteristics. To explore this possibility, we have developed a job status prediction method for the execution of jobs on scientific clusters. The Random Forests algorithm was applied to extract and characterize the patterns of unsuccessful job statuses. Experimental results show that our method can predict the unsuccessful job statuses from the monitored ongoing job executions in 99.8% the cases with 83.6% recall and 94.8% precision. This prediction accuracy can be sufficiently high that it can be used to mitigation procedures of predicted failures.**

Keywords*—*Reliability, Job Log Analysis, Job Status Prediction

I. Introduction

Technical advances in scientific clusters have involved increasing scale of volumes in data accesses and movement, number of nodes involving job executions, and exploited parallelism in applications on multiple cores. These changes lead to unforeseen increased scales of interactions and communications in software executions between hardware resources and nodes in a cluster. However, technical advances have not reached the reliability of hardware components and softwares in the same scale. Therefore, the rates of unsuccessful job statuses have been dramatically increased due to the combinations of the increased scales of software and hardware executions and the lack of improvements in their reliability in the same scale.

Despite these increased unsuccessful job statuses, system administrators are overwhelmed by large-scale logs to take actions for reducing unsuccessful job statuses and their impacts such as wasted resources. It is challenging to analyze the unsuccessful job statuses on large scientific clusters due to the sizes of logs and noises in their measurements from the interactions and interferences in the job executions. Since a node can be assigned multiple tasks from different jobs, the measured executions can be noisy because of performance interferences in shared hardware resources from co-located tasks from multiple jobs. This indeterministic variances and noises due to interferences make job status prediction more challenging. In order to tackle these challenges and improve the reliability of scientific clusters and application executions, we have developed an automated job status prediction method based on machine learning classification mechanisms. We will also discuss online job status prediction by extending our method.

Scientific clusters usually have multiple types of nodes: compute nodes, storage nodes with parallel file systems, data transfer nodes for network accesses, and special purpose nodes for database or web services. Parallel programming framework such as Message Passing Interface (MPI) are generally used to implement scientific applications. To execute this MPI-type application on a scientific cluster, a job specifying application execution is submitted to the job scheduling engine. Then, the scheduling engine dispatches the executions of the job by assigning one or multiple nodes. For scheduling decisions, the scheduling engine considers the current load of a cluster and the requests of resource usages specified in the job request. The executions of dispatched jobs are stored in job logs. They contain multiple fields that can provide information about job specification such as executable name and parameters, job assignment in a cluster such as host name, resource usage related resource usages, failure code representing failures or successes, and exit code returned from the application. MPI-type jobs are usually implemented to involve parallel executions. The parallel executions of a job are divided into parts, and they are represented as tasks. A task (also known as a part of job array) is usually assigned to a node, and the parallel execution within a node is stored as an aggregated measurement in job logs. When a job is dispatched to multiple nodes, the execution of each node is stored in separate records as a task in job logs.

The resources of a cluster are reclaimed when the job finishes with success or ends with failures due to unsuccessful job statues. The failures are causes by various reasons. They can be caused by software issues such as application bugs or errors, job related issues such as insufficient allocated resources, e.g. wall clocks, CPU time, or memory, system related issues such as OS errors, file system errors, or cluster managing system errors, or transient or permanent faults caused by hardwares. We define unsuccessful job statues as the job execution patterns leading to interrupted and failed job executions. The ultimate goal of predicting unsuccessful job status is to predict future failures in an online fashion based on the historical characteristic patterns of executions from job failures compared to the current job processing statues and characteristics. Failures are costly to users and systems since they waste time and system resources. Online failure prediction can mitigate these wastes by taking early actions for those predicted failures. For instance, an ongoing task execution of a job can be predicted to be failed based on the characteristic patterns in the execution. Provided the accurate prediction, system administrators or an automated job manager can explicitly terminate the execution, fix the problem and

reschedule the job, instead of wasting time and resources by waiting until actual failure happens. If the failure is caused by an application error, the developer of the application can be notified to fix the error.

In our experiments, we used job logs from Genepool scientific cluster [1] at NERSC. The Genepool cluster includes the sufficiently large number of nodes and multiple parallel executions from the tasks that incur complexity challenges for manual log analysis. Due to the complexity and the size of the job logs, it is challenging for developers and system administrators to manually analyze and extract meaningful information from the job logs. Since there exist multiple resource usage related fields mostly correlated to each other due to the interactions in the executions, it is challenging to make a prediction model in a manual way.

To tackle these challenging problems, we apply machine learning classification mechanisms on the measured job logs. Machine learning classification mechanisms are used for job status prediction by characterizing the patterns of task executions in a job with the classes of successful and unsuccessful job statuses. We use 13 resource-usage-related fields measuring resource usages in the job logs, and feed them as features to machine learning mechanisms. The failed code and exit code are used as labels, the machine learning classification mechanism is trained to classify or predict labels from the features in each record in the job logs. We compared multiple machine learning classification mechanisms for job status prediction: the Decision tree [19], the Random forests [3], the Naive Bayes [16], the Logistic regression [13], and the Support vector machine (SVM) [6].

The contributions of this paper are:

- Applying machine learning classification for automated job status prediction.
  We empirically compare multiple machine learning classification methods for job status predictions on scientific clusters. The Random forests method is selected because of its multiple advantages , which will be shown in in Sec. IV. In addition, we provide an analysis of how to select and apply the Random forests to improve the prediction results.
- Providing empirical evidence of appropriateness in using resource usage related measurements for online job status prediction.
  We show that machine learning-baed online job status prediction based on these measurements can provide higher prediction accuracy in terms of recall and precision compared with previously proposed literatures. In addition, the prediction result is accurate for both near-term (several minutes) and long-term predictions (several days), which is an improvement from the previously proposed event-based approaches that favors to near-term predictions.
- Parallelizing log analysis to handle large volume of logs for training and prediction.

We show how we applied these mechanisms for the job status predictions in Sec. III-B. We present the experimental results showing that the Random forests method shows the best prediction results and is able to predict job statuses from the job logs of a scientific cluster in an automated manner in Sec. IV. The rest of paper is organized as follows. Sec. II presents the related work. Sec. III demonstrates the design and implementation of job status prediction. Sec. IV presents the experimental evaluation and discussion of offline and online job status prediction. The conclusion is in Sec. V.

II. Related Work

Salfner et al. [21] presented the survey of online failure prediction methods. They proposed to make distinction between the root cause analysis and the online failure prediction, and our focus in this paper is on the online failure prediction. The definition of the failure in the survey is an event that occurs when a misbehavior in the system results in an incorrect output. Our definition of failure is the failed task executions labeled by a job scheduler. It is broader than their definition so that it includes faults and errors that do not involve the misbehavior or incorrect status. In addition, most of the previous researches for the online failure detection with the event-based failure definition resulted in better near-term prediction than long-term prediction. Our online failure prediction based on the job executions shows similarly accurate prediction in both near-term and long-term predictions. This is mainly because our failure prediction uses the characterized patterns of the job executions for failures. When applications are not changed frequently as in most scientific clusters, our failure prediction method is more appropriate than those prediction methods based on events.

Schroeder et al. [22] presented a study about failures in High-performance computing (HPC) systems. Chen et al. [4] presented a study about failures in Cloud environment, using measured data from Google cluster [20]. These studies show the increasing failure rates in HPC clusters and Cloud clusters. Guan et al. [10] proposed to use the Principal Component Analysis (PCA) [15] for detecting an anomaly in Cloud environment. Detecting an anomaly is orthogonal to our work for failure prediction.

Heien et al. [12] claimed that application-specific failure models achieve better accuracy because failure events are heterogeneously correlated in space and time. Hamerly et al. [11] proposed to use Naive Bayes classification to predict hard disk failures. These proposed works cannot provide location information of failures while our work can.

Linag et al. [17] proposed to use temporal and spatial compression of failures for predicting them. Fu et al. [8] proposed to use temporal and spatial event correlation for failure prediction. Gainaru et al. [9] proposed to combine outlier detection and correlations in location and signal for failure prediction. While these event-based failure predictions provide location information of failures, their prediction is not accurate for long-term prediction while our work is accurate for both long-term and near-term prediction. In addition, these methods lack parallelization to utilize multiple cores and nodes in order to handle large-size data.

Snir et al. [23] presented the survey paper of addressing failures in exa-scale computing. This survey showed that current failure predictors achieve a precision of over 90% and a recall of below 50%. Our failure prediction method achieves 94.8% precision and 83.6% recall. Nakka et al. [18] proposed

to use the decision tree algorithm for predicting node failures in HPC after comparing multiple machine learning mechanisms. Since they use separate usage logs and node failure logs, their prediction results (81.3% recall and 73.9% precision) are less accurate than ours. In addition, the node failures cover less amount of failures than our task failures by definition.

Di et al. [7] proposed a mechanism of checkpoint and restart task execution to improve reliability in the event of failures. Chen et al. [5] proposed to use machine learning-based binning of relative changes to reduce the cost for checkpointing. These checkpoint and restart mechanisms can improve the reliability of scientific clusters, and our work is complementary to these approaches by providing predictions of failures and preparing the checkpointing in accordance with the predictions.

Previously, we have applied machine learning mechanisms to identify performance bottleneck, using fingerprints generated from micro-benchmarks [24]. Our current work extends the previous method to cluster-scale multi-nodes execution in terms of using the Random Forests to characterize the patterns of task executions leading to failures.

III. Model Development

A. Preliminaries

Machine learning classification mechanism is a supervised learning that is designed to infer class labels from the labeled trained set having input features associated with the class labels. The decision tree algorithm is a machine learning classification mechanism, where patterns of input features are analyzed to create a predictive model. A decision tree consists of non-leaf nodes representing tests of features, branches between nodes representing the outcomes of the tests, and leaf nodes holding the class labels.

Constructing the most optimal and accurate decision tree is usually NP-hard on a given training set [19]. To construct a decision tree model, most of the practical algorithms use a greedy approach using heuristics such as *information gain*. Using these algorithms, the training data is recursively partitioned into smaller subsets. When partitioning the dataset, the feature with the highest splitting criterion such as *information gain* is chosen as the splitting feature. This feature minimizes the information needed to classify the data in the resulting partitions, and reflects the least randomness in these partitions.

The Random forests method [3] consists of multiple decision trees that are constructed by randomly chosen features with a predefined number of features. The random features classify a label by voting, a plurality decision from individual decision trees. Because of the law of large numbers, the Random forests method is less prone to generalization error (overfit) as randomness are added with more trees. In addition, the generalization error converges to a limited value.

B. Job Status Prediction

The purpose of job status prediction is to predict future failures based on the characteristic patterns or fingerprints of unsuccessful job statues. As failed job executions labeled by a job scheduler when their executions are interrupted or stopped, the unsuccessful job statuses are exhibited in the measurements of the failed job executions. As a failure is labeled by a job scheduler after the task execution finishes, it is important to build an online prediction from the runtime information. Since failures are labeled systematically by a scheduler, the inference from the supervised machine learning classifier can result in better error rate than the case with manually labeled data.

We compared multiple supervised machine learning clarification mechanisms for failure prediction: the Decision tree [19], the Random forests [3], the Naive Bayes [16], the Logistic regression [13], and the Support vector machine (SVM) [6]. The Naive Bayes is based on the independence of features so that it does not fit to performance related features in job logs, as some of the features are highly correlated. The SVM constructs hyperplane from a selected kernel function to separate space of features associated with the class labels. The drawback of the SVM is in the difficulty of the correctly fitted kernel function and additional computational cost for conversion to hyperplane. The logistic regression is a special case of generalized linear model. It can probabilistically classify the features by fitting them to linear space using logistic function. The decision tree and the Random forests based on the randomly generated decision trees have two advantages compared to the aforementioned machine learning classification mechanisms. One is that they have relatively short training time and classification (prediction) time. The other is that the trained decision tree is easy to understand so that it can help analyze predicted patterns of failures.

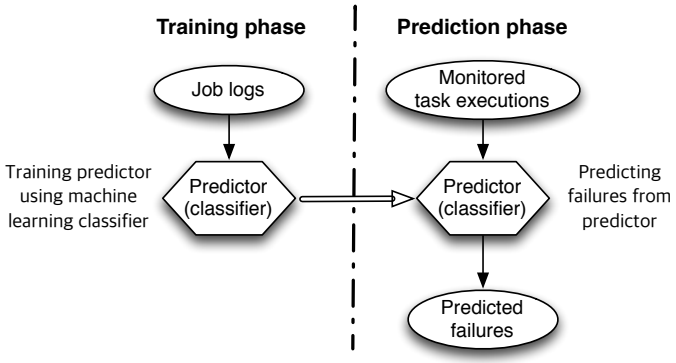

Fig. 1: The overview of failure prediction using machine learning classifier

Fig. 1 illustrates the overview of our online failure prediction. In the training phase, a machine learning classifier is trained with job logs. The job logs contain failed codes and exit codes, and they are used as labels in the training. These job logs also contain 13 resource-usage-related fields associated with failures, and they are used as features in the training. Since the number of fields is reasonably small for the experiments, we use all 13 fields instead of conducing feature selection to reduce the number of fields. However, the generated classification rules in the training of classifiers select the fields with the highest splitting criterion, which implicitly conducts the feature selection. The training of machine learning classification mechanism extract and characterize the patterns of features associated with labeled failures, and build a classifier based on these patterns so that they can classify or predict labels from the features in each record in the job logs.

While these failed code and exit code can include hints about the possible causes of failures, the identification of the causes of failures is not the focus of this paper. Instead, we focus on job status prediction to find patterns and characteristics of unsuccessful job statues.

In the prediction phase, the same classifier can be used to predict unknown class labels from the features of future measurement data. Furthermore, the trained machine learning classifier can be used for online failure prediction. For online failure prediction, the classifier needs to be trained with the progress information and to be able to predict failures from the features of ongoing execution. Normalization of features with dividing by wall clocks or CPU time can convert the originally measured features in job logs. The measured features can be converted to progress information as the form of the ratio of performance related measures by the unit of wall clocks or CPU time. Once the classifier is trained with the features of the progress ratio, it can predict failures from the features as the same form of the progress ratio. As they are normalized by time-based value, the prediction is possible regardless of how much task executions are progressed or where the progress is positioned. Our cluster monitoring tool, Procmon [14] can provide this runtime progress information for online failure prediction.

IV. Experimental Results

A. Experimental Setup

In our experiments, we have used job logs collected on Genepool cluster at NERSC, consisting of 774 nodes [1]. The logs were written by the Univa Grid Engine [2] when each job was finished. It contains 45 fields such as host name, job name, failed code, exit code, and resource usages. We selected 13 resource usage related fields: wall clock, user/system CPU time, soft/hard page faults, file block input/output, voluntary/involuntary context switches, aggregated memory usage, aggregated I/O, maximum resident set size, and maximum virtual memory. The size of the logs from Nov. 1, 2015 to Feb. 7, 2015 (PST) is 5.2 million records. The experiments were conducted on a machine with two 8-core Intel Xeon E5-2670 CPUs and 64 GB memory.

B. Job Status Prediction

When a job executes multiple tasks in multiple nodes, each task execution is recorded into a separate line in the job log. The failure codes represent different unsuccessful job exit cases involving incorrect status in task executions due to mostly software failures. For example, they include rescheduling from job scheduler or application errors, input/output errors, and being killed by signals. The failed code 25 means that a job is rescheduled without implying an error on a node. The failed code 30 means that a job is rescheduled to error state without implying an error on a node. The failed code 8 implies an error on the node that prevented job setup. The failed codes 26 and 28 are issues on the filesystems input/output. The failed code 100 means that allocated resources are exhausted such as wall clock, memory, or I/O. In addition, the exit code represents exit status from a task execution, which applications of Unix or Linux generally share meanings including signals.

In order to predict job statuses from job executions, we used machine learning mechanisms on the measured job logs. We compared multiple machine learning mechanisms that are widely used for solving the classification problem: the Decision tree [19], the Random forests [3], the Naive Bayes [16], the Logistic regression [13], and the Support vector machine (SVM) [6]. The distinction of unsuccessful job exits comes from the failed code in the jog logs, which we used failed code and exit code as a binary tuple for a multi-class label or only the status of failed or succeeded as a binary class label. In addition, we used 13 resource usage related fields for features shown in Tab. I. We trained aforementioned machine learning classification mechanisms with these labels and features in the training set. These trained classifiers can predict unsuccessful job exits from the features in the test set. Since our purpose of job status prediction is to identify unsuccessful job exits in ongoing executions of jobs, we only used resource usage related fields from the job logs. While the job logs were stored after the unsuccessful job exit occurred, the job status predictions were possible by only using the progress ratios of resource usage related fields after the normalization with diving by wall clocks or CPU time.

Cross Validation: Fig. 2 shows the results of 5-fold cross validation by the multiple machine learning mechanisms. The 6 weeks of job logs from Nov. 1, 2015 to Jan. 31, 2015 (PST) are randomly and equally divided to five sets. The number of positives (unsuccessful job exits) is 490,174 and the number of negatives (successful job finishes) is 4,304,988. Each set was used as a test set, and the rest four sets were used as training sets. This cross validation was repeated five times with different test sets. The results are the average values of predictions, as accuracy ($\frac{TP+TN}{TP+TN+FP+FN}$), recall ($\frac{TP}{TP+FN}$), and precision ($\frac{TP}{TP+FP}$), where true positives, true negatives, false positives, and false negatives are represented as TP, TN, FP, and FN. For the clarification, TP are correctly predicted unsuccessful job exits. TN are correctly predicted successful job finishes. FP are incorrectly predicted unsuccessful job exits, i.e., successful job finishes that are predicted as unsuccessful job exits. FN are incorrectly predicted successful job finishes, i.e., unsuccessful job exits that are predicted as successful job finishes.

As shown in Fig. 2, the Decision tree and the Random forests methods show better prediction results in terms of accuracy, recall, and precision, compared to the Naive Bayes, the Logistic regression, and the SVM methods. The predictions by the Naive Bayes, the Logistic regression and the SVM are for binary class labels about the status of unsuccessful job exits and successful job finishes. The predictions by the Decision tree and the Random forests methods are both multi-class of binary tuple of failed code and exit code as well as the binary class labels. In order to interpret the strength of predictions, accuracy is the most useful measure. The high accuracy shows both high values of TP and TN, and the low values of FP and FN. The accuracy of our predictions by the Decision tree and Random forests methods were more than 99%. In addition to the accuracy, recall and precision are needed to be compared to interpret the strength of predictions. For instance, the precision of the SVM was 100%. However, the recall of the SVM was almost 0%. This was because the SVM only predicted one unsuccessful job exits correctly out of 490,174 unsuccessful

TABLE I: The description of resource usage related features.

Feature	Description
Wall clock	The duration between start and end of a task
User CPU time	The sum of spent time from CPU cores in user level
System CPU time	The sum of spent time from CPU cores in system level
CPU time	The sum of user CPU time and system CPU time
Maximum resident set size	Maximum value of utilized memory size during execution
Page reclaims	Soft page faults without involving I/O
Page faults	Hard page faults with involving I/O
Block input operations	The number of that times the file system had to perform input
Block output operations	The number of times that the file system had to perform output
Voluntary context switches	the number of times for voluntary context switches
Involuntary context switches	the number of times for involuntary context switches
Memory	The integral memory usage in Gbytes $*$ CPU time in second
IO	The amount of data transferred in input/output operations

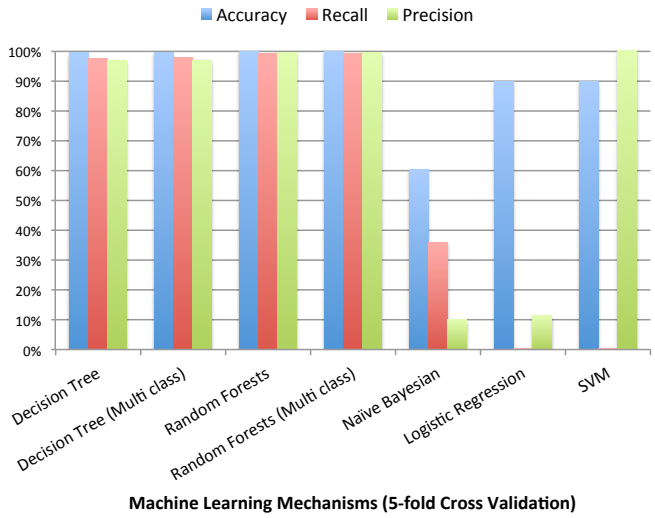

Fig. 2: Job status predictions of 5-fold cross validations by machine learning mechanisms

job exits (490,173 FN). Therefore, the strength of predictions can be compared by all these measures, accuracy, recall, and precision. The prediction results by Random forests method shows the best prediction results. In terms of (Accuracy (A), Recall (R), Precision (P)), the prediction results of results by Random forests were (99.8%, 98.8%, 99.4%) for the multi-class label and (99.8%, 98.8%, 99.4%) for the binary class label. The precision results by the Decision tree method are (99.3%, 96.4%, 96.8%) for the multi-class label and (99.4%, 97.3%, 96.8%) for the binary class label. We use this tertiary tuple (A,R,P) for showing other prediction results in this section. The prediction accuracies by the Naive Bayes, the Logistic regression and the SVM were lower than those by the Decision tree and the Random forests. We think that converting multi-class labels to binary class labels for the Naive Bayes, the Logistic regression, and the SVM made the generated classifiers too simple and too weak to characterize the different patterns of multiple job exit r. In addition, we think that the multiple randomly generated classifiers by the Random forests showed better prediction accuracy due to more robustness against generalization error (overfit) than the prediction accuracy by the decision tree.

Prediction of Future Jobs: Fig. 3 shows the prediction results of the test set of job logs from Feb. 1, 2015 to Feb. 7, 2015 (PST). The test set includes 16,902 unsuccessful job exits and 409,035 successful job finishes. The 6 weeks of job logs from Nov. 1, 2015 to Jan. 31, 2015 (PST) were used as the training set. The prediction results by the Random forests method are (98.5%, 63.9%, 97.6%) for the multi-class label and (97.7%, 44.7%, 96.9%) for the binary class label. The precision results by the Decision tree method are (96.6%, 39.3%, 61.7%) for the multi-class label and (97.0%, 42.0%, 69.0%) for the binary class label.

Compared to the results of cross validation in Fig. 2, the accuracies of the prediction results by the Decision tree and the Random forests methods are decreased. On the other hand, the accuracies of the prediction results by the Naive Bayes, the Logistic regression, and the SVM methods are slightly increased. This is because the unsuccessful job ratio of the test set (3.97%) is lower than that of training set (10.22%), which makes the number of FN smaller even when the almost all unsuccessful job exits are incorrectly predicted as successful job finishes from those three mechanisms. Obviously, these incorrect predictions are shown in the decreased precisions of those three mechanisms.

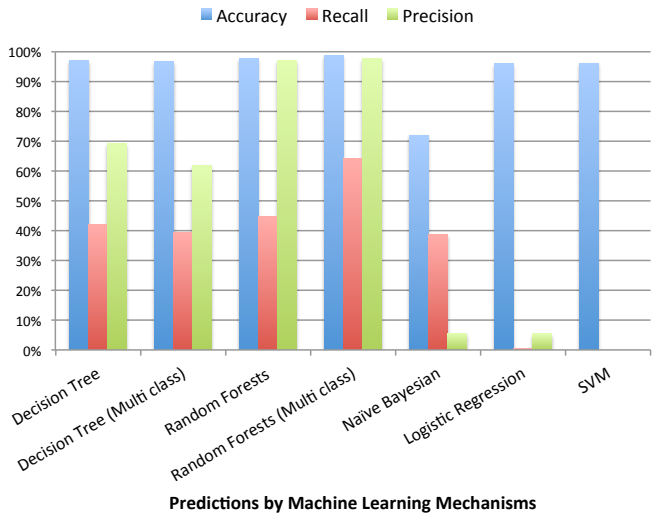

Fig. 3: Job status predictions by machine learning mechanisms

The recall and precision of the predictions of the Decision tree and the Random forests methods are decreased. These decreases are expected since the cross-validation has used the test set collected in the same duration as the training set. They presumably share the similar patterns of executions, which makes the predictions of cross-validation better than predictions of the test set for the future duration. The randomly constructed multiple decision trees in the Random forests method make their predictions less prone to generalization errors (overfit), compared to those of the (single) Decision tree. We think that this difference makes better recall and precision of the Random forests method in addition to slightly better accuracy.

Prediction of Filtered Future Jobs: We investigated the reason of prediction accuracy decreases for the future data compared to that of cross-validation. We found that there existed one majority case of unsuccessful job exits, which was represented as failed code, 25 and exit code, 99. The exit code 99 forces the job to be rescheduled with the failed code, 25. This was done by having the batch script of the job exited with the status 99. When the site administrator reschedules a job which would have a failed code 25, the exit codes would have 143 or 137 in this case (meaning signal of SIGTERM or SIGKILL respectively). Therefore, this specific case of unsuccessful job exits (25, 99) is always intended by the user choice in the execution.

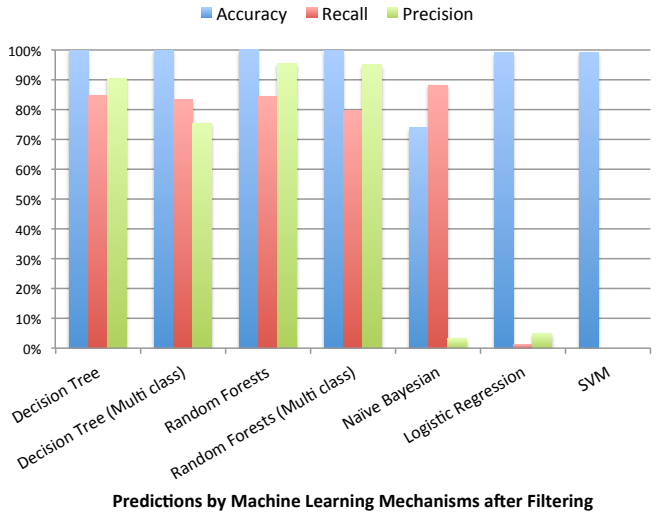

Fig. 4: Job Status predictions by machine learning mechanisms after filtering

Further investigation revealed most of the unsuccessful job exits case from a particular application. The application kept the executions after executing some tasks, and intentionally rescheduling itself in order to essentially keep the executions. Its main purpose is to monitor jobs running on an FPGA system outside of the batch system. In addition, it has other rescheduling cases waiting for a database server or web server that is busy or unavailable. Since the task executions of this application depend on the external services and behaviors, they showed the indeterministic patterns of executions that made difficult to characterize and predict from the machine learning mechanisms. This is shown that the most of FP and FN come from the specific unsuccessful job exits.

When the task of this application is rescheduled by exiting with code 99, it is labeled as an unsuccessful job exit by the scheduler due to the customized rescheduling. As it is a normal behavior from the application developer's point of view, it is better to be labeled as a successful finish instead of an unsuccessful job exit. However, regardless of labeling a successful finish or an unsuccessful job exit, this customized patterns of task executions made the predictions by the machine learning classifiers challenging, due to its peculiarity and indeterministic behaviors. Therefore, we have filtered out the unsuccessful job exit case (25, 99) to see how much the predictions improve.

Fig. 4 shows the prediction results of filtered test set of job logs from the same duration. The prediction results by the Random forests method are (99.8%, 79.6%, 94.9%) for the multi-class label and (99.9%, 84.3%, 95.2%) for the binary class label. The precision results by the Decision tree method are (99.6%, 83.2%, 75.4%) for the multi-class label and (99.8%, 84.6%, 90.3%) for the binary class label. The prediction results are improved in all measures, accuracy, recall and precision, compared to those with the unfiltered test set in Fig. 3. We think that the application is highly customized and incurred much different and indeterministic tasks executions than usual tasks executions of most other jobs. We plan to investigate further whether the highly customized job causing indeterministic patterns is discovered in the job executions in other scientific clusters. If this is the particular case in the Genepool cluster, filtering out some indeterministic tasks would not be necessary. We believe that this type of intentional rescheduling jobs are much rare.

The recalls of the multi-class labels from the Decision tree and the Random forests methods are worse than those of the binary class labels. The precisions of the multi-class labels by the Random forests method are almost the same as those of the binary class labels, while those of the multi-class labels by the Decision tree method are worse than those of the binary class labels. We think that the multiple decision trees in the Random forests method not only makes less generalization errors but also includes more characteristic patterns or fingerprints of different types of unsuccessful job exits. Therefore, the precisions by the Random Forests method are little degraded in the multi-class labels compared to the binary labels. Interestingly, the recalls of the multi-class labels by the Random forests method are better than those of the binary class labels in the unfiltered results in Fig. 3. We think that this also shows the less proneness of generalization errors by the Random forests, where they are able to predict more TP and less FN, which makes higher recall in the multi-class labels.

Parameter Selection for Random Forests: Fig. 5 shows the prediction results by the Random forests method with the different sizes of training sets. We used from 2 weeks to 8 weeks on and before Jan. 31, 2015 (PST), e.g., 2 weeks training set is from Jan. 18, 2015 to Jan. 31, 2015 (PST). The prediction results are (99.9%, 80.9%, 90.7%) for 2 weeks, (99.9%, 80.0%, 93.5%) for 4 weeks, (99.8%, 79.6%, 94.9%) for 6 weeks, and (99.8%, 78.8%, 94.0%) for 8 weeks. The accuracy is not much sensitive to the sizes of training sets. The recall is the best with

the 2 weeks training set. The precision is the best with the 6 weeks training set. As the difference of precisions between 2 weeks and 6 weeks is more significant than that of the recalls, 6 weeks was selected as the size of the training set. In addition, the precision is more important than the recall, since FP is more critical than FN in the job status prediction. It is because the application of the job status prediction such as early termination of jobs incorrectly predicted as unsuccessful job exits (FP) can be much more costly than the inaction due to the incorrectly predicted as successes (FN). This is another reason to select 6 weeks as the size of the training set.

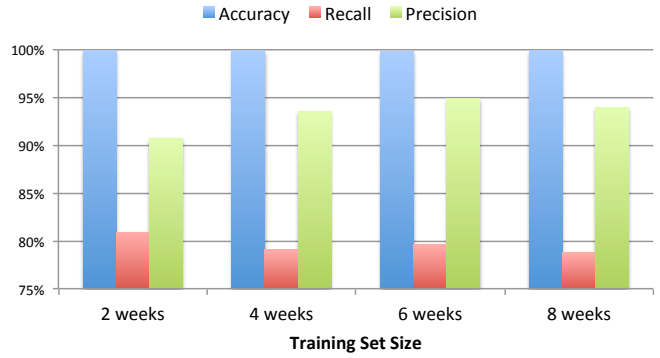

Fig. 5: Job status predictions by different training set sizes for multi-class label

Fig. 6 shows the prediction results by the Random forests method with the different depths of the training sets. The prediction results are (99.8%, 68.4%, 94.9%) for depth 5, (99.8%, 79.6%, 94.9%) for depth 10, and (99.8%, 72.2%, 92.0%) for depth 15. The recall and the precision are the best with the depth, 10, and this becomes the selected depth. We think that the prediction degradation in recall and precision with the depth, 15 is due to the generalization errors. As the more depth in Random forests method includes more patterns of the training set, it leads to unnecessary information that only fits to the training set and not needed to predict the future test set.

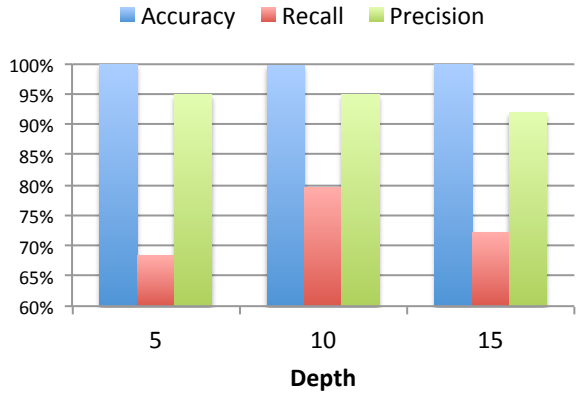

Fig. 6: Job status predictions by the Random forests with different depths for multi-class label

Fig. 7 shows the prediction results by the Random forests method with the different numbers of decision trees. The prediction results are (99.8%, 77.7%, 90.6%) for size 5, (99.9%, 78.4%, 92.6%) for size 11, (99.8%, 79.6%, 94.9%) for size 21, and (99.9%, 79.6%, 93.7%) for size 31. The recall and the precision are the best with the number of trees, 21, and this becomes the selected number. Increasing numbers of decision trees allow more randomness in the prediction that makes less generalization errors. The tradeoff is in the additional computational cost for constructing more trees in the training and traversing more trees in the prediction. The precision with number of decision trees, 31, is decreased, compared to that with 21. As increasing the number of trees does not increase the chances of the generalization errors, we think that this is the probabilistic coincidence or marginal errors from the nature of randomness in the Random forests method.

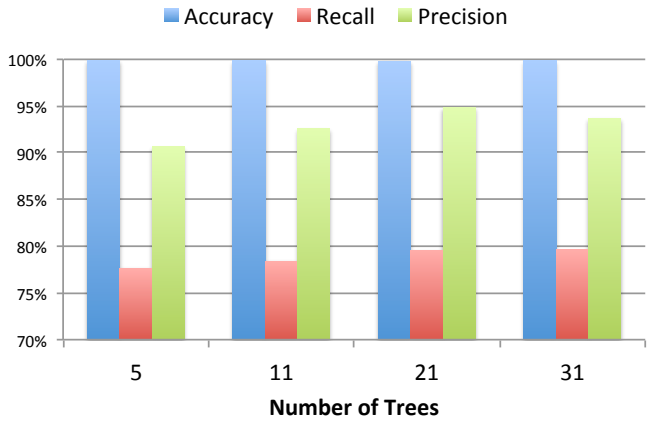

Fig. 7: Job status predictions by the Random forests with different numbers of trees for the multi-class label

Predictor Update: Fig. 8 shows the job status prediction results of test sets in different weeks. The update of predictor was done for each weekly test set in order to evaluate the effect of weekly update of prediction model. Each 6-week training set was used, and its end date is the previous date of the start date of each test set. The prediction results show similar prediction accuracy except the two weeks from 2/15/2015 to 2/28/2015. The degradations in these test sets were due to uncharacterized unsuccessful job exits (unseen pairs of failure code and exit code). The average prediction results shown in Fig. 8 were (99.3%, 75.2%, 92.4%).

Fig. 9 shows the job status prediction results of test sets in different dates in Feb. 2015. The update of predictor was done for each one-day test set in order to evaluate the effect of daily update of prediction model. Each 6-week training set was used, and its end date is the previous date of each test set. The results from daily update showed more consistently accurate and slight better than the prediction results of weekly update in Fig. 8. This is because daily update can characterize some of unseen unsuccessful job exits that were missed in weekly updates. As shown in lower recalls in Feb. 20 and Feb. 24 and lower precision in Feb. 28, prediction results were degraded when there were unseen unsuccessful job exits even after daily update. The average prediction of daily update

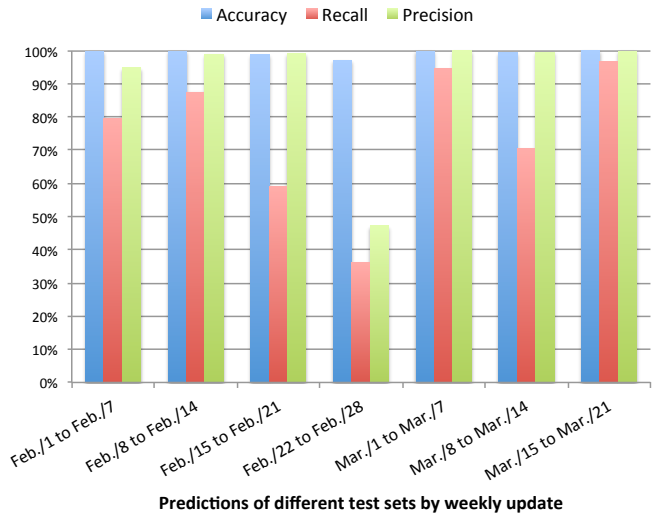

Fig. 8: Job status predictions of different test sets by the Random forests with weekly updating (training)

results shown in Fig. 9 were (99.4%, 82.7%, 95.2%), which were improved from those of weekly update. the prediction results were accurate both near-term (Fig. 9) and long-term (Fig. 8) predictions. This is because our job status prediction is based on the resource usage related measurement in task executions instead of event-based information that is dependent on the temporal correlations.

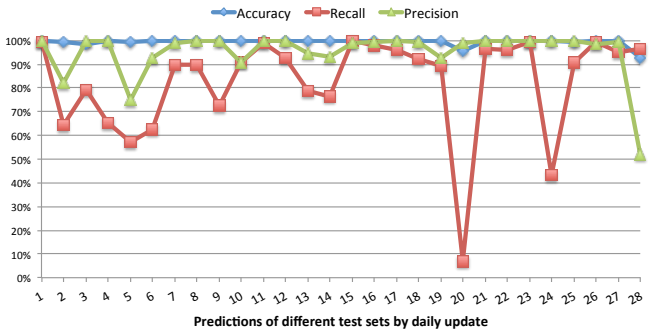

Fig. 9: Job status predictions of test sets in different dates by the Random forests with daily updating (training)

Online Job Status Prediction: Fig. 10 shows the online job status prediction results by the Random forests method with the different normalizations with wall clocks and CPU time. The prediction results with the wall clock normalization are (99.8%, 83.6%, 94.8%) for the multi-class label. The prediction results with the CPU time normalization are (99.8%, 84.1%, 92.1%) for the multi-class label. Since the job logs store the finished information of jobs regardless of the unsuccessful job exits or successful finishes, online job status prediction needs to be based on the progress information instead of post-processing of the measured information after the job is finished. The Normalization with dividing by the wall clocks or CPU time can provide progress information as the form of the ratio of resource usage related measurements by the unit of the wall clocks or CPU time.

As shown in Fig. 10, the normalization by dividing by the wall clocks is better than that of the CPU time. In addition, the multi-class prediction is better than the binary class label, and the predictions are very similar to that of the original test set without the normalization. We think that it is because the CPU time includes more crucial information for the job status prediction than the wall clocks. Furthermore, the wall clocks do not include crucial information, and the prediction is not much degraded after the normalization with the wall clocks. As shown in the predictions of one-week test set without re-training the Random forests, In short, the progress information after the normalization with the wall clocks is shown almost the same prediction quality for the online job status prediction as the prediction without the normalization for the offline job status prediction.

The 6 weeks training set (4,795,162 records normalized by dividing by wall clocks) incurs significant computational challenges for an analysis. We used Apache Spark [25] to distribute and parallelize computational loads of the online job status prediction method. The parallelized computations utilize all the 16 cores in a node in the experiments. The training time by the Random forests method with the 6 weeks training set, the depth, 10 and the number of trees, 21 took 139.4 seconds. The test time of 1 week test set (425,937 records) was 23.4 seconds, which is 55 ms per record. While we tested one-week test set, the delay of 55 ms per record will be increased with the smaller batch size of the test duration. However, the prediction results from the one-week test set showed that the trained Random forests method did not need to be frequently updated, i.e., one-week frequency was sufficient to result in 99.8% accuracy, 83.6% recall, and 94.8% precision. Since there is the tradeoff of the update frequency between computational cost (delay) and prediction accuracy, and the frequency is dependent on the characteristics of the job logs, we plan to further study this aspect.

As we used one node for the experiments, the short time of training and testing showed that the delay of the online job status prediction is not an obstacle for a realtime prediction. For the job status prediction on a larger cluster, we can simply increase the number of nodes in order to reduce the delay. While the normalized progress information from the job logs can be used for the online job status prediction, the information is not readily available in the job scheduler until the task execution is finished. In order to access the runtime progress information, we used a cluster monitoring tool, Procmon [14]. We plan to study the online job status prediction on the Procmon data, and briefly discussed in Sec. IV-C.

C. Discussion

In order to deploy the online job status predictor in a cluster, a cluster monitoring tool is necessary to gather runtime execution information from the cluster. Our cluster monitoring tool, Procmon, can provide the resource usage related measurements to feed into the trained online predictor with the Random forests. The current size of the job logs is approximately 50MB per day, and the current size of the Procmon logs is 30GB per day with the 30-second measurement interval. The difference

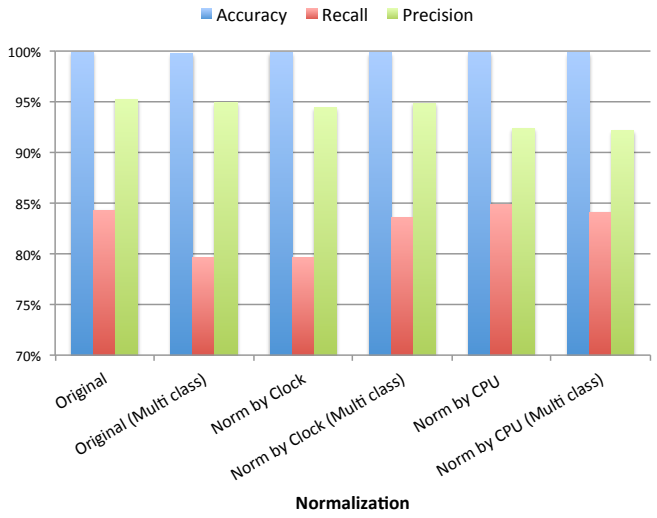

Fig. 10: Job status predictions by the Random forests with different normalizations

in data sizes between these logs results from the frequency of measurement collection. While the job logs are collected once a task finishes, the Procmon logs are collected based on the pre-defined frequency. In other words, the Procmon logs can monitor and collect information about the runtime executions with the cost of the storage I/O. The delay of the training and prediction by the online job status predictor will be significantly increased to accommodate the order of larger Procmon log sizes. In order not to increase the delay, we plan to use more nodes for parallel analysis. In addition, we will investigate filtering out unnecessary information for the online job status prediction from the Procmon logs to decrease the cost of the I/O.

There exists a possibility that the progress of a task might not be constant, i.e., resource usage related measurements from the same task and the same job might have different patterns measured in different intervals. This would degrade prediction results due to the differences in the measured patterns. We think two possible solutions for this case. We can merge multiple consecutive measurements to amortize the variances of the progress in the task executions. We think that as more measurements are merged, less variances of tasks executions will have. Thus they will be similar to the entire executions, and will result in similar prediction results. Another solution is to use the measurements of Procmon logs as the training set, instead of using resource usage related measurements of the job logs. We think this solution will improve the overall prediction results as the Procmon logs can characterize the variances in the task executions with the increased costs for computation and storage. As the former solution incurs less cost than the latter, we will compare the tradeoff between the prediction results and the overall costs.

V. Conclusions

Scientific clusters experience increasing trends of failures rates. To tackle the challenges to provide reliability against the increasing trends of number of nodes, data size, and complex interactions between software and hardware, it is crucial to provide accurate failure predictions for scientific clusters. Our job status prediction can help reduce time, resource waste, and cost against failures by enabling the preparation of checkpointing or taking actions for imminent failures.

Our automated job status prediction uses the Random forests method to characterize the patterns of performance related measurements in unsuccessful job executions. The trained Random forests can identify the characterized patterns from the runtime job executions in order to predict imminent failures resulted from the unsuccessful job statues. We provide empirical evidence that performance related measurements are appropriate data sources for the machine learning based job status prediction that can be extended to online failure prediction. A machine learning classification method, the Random Forests was applied to extract and characterize the patterns of unsuccessful job statuses. Using the trained Random Forests, the experimental results show the prediction of the unsuccessful job statuses from the monitored ongoing job executions in 99.8% the cases with 83.6% recall and 94.8% precision. This prediction accuracy may be sufficiently high so that the predictions can be used to initiate predicted failure mitigation procedures.

The future work includes applying online failure prediction on collected runtime execution information from runtime measurement tools. We plan to apply online failure prediction on other scientific clusters that have a larger number of nodes and various applications.

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. The authors would like to thank Bryce Foster and Alex Copeland at JGI; Douglas Jacobson, Jay Srinivasan, and Richard Gerber at NERSC; Arie Shoshani at LBNL; Lucy Nowell and Richard Carlson at Dept. of Energy.

References

[1] "Genepool cluster," https://www.nersc.gov/users/computational-systems/genepool, 2015.

[2] "Univa grid engine," http://www.univa.com/products/grid-engine.php, 2015.

[3] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[4] X. Chen, C.-D. Lu, and K. Pattabiraman, "Failure Analysis of Jobs in Compute Clouds: A Google Cluster Case Study," in *2014 IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)*, Nov. 2014, pp. 167–177.

[5] Z. Chen, S. W. Son, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary, "NUMARCK: Machine Learning Algorithm for Resiliency and Checkpointing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 733–744.

[6] C. Cortes and V. Vapnik, "Support-vector networks," *Mach Learn*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[7] S. Di, Y. Robert, F. Vivien, D. Kondo, C.-L. Wang, and F. Cappello, "Optimization of cloud task processing with checkpoint-restart mechanism," in *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*, Nov. 2013, pp. 1–12.

[8] S. Fu and C.-Z. Xu, "Exploring event correlation for failure prediction in coalitions of clusters," in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, 2007. SC '07*, Nov. 2007, pp. 1–12.

[9] A. Gainaru, F. Cappello, M. Snir, and W. Kramer, "Fault Prediction Under the Microscope: A Closer Look into HPC Systems," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 77:1–77:11.

[10] Q. Guan and S. Fu, "Adaptive Anomaly Identification by Exploring Metric Subspace in Cloud Computing Infrastructures," in *2013 IEEE 32nd International Symposium on Reliable Distributed Systems (SRDS)*, Sep. 2013, pp. 205–214.

[11] G. Hamerly and C. Elkan, "Bayesian Approaches to Failure Prediction for Disk Drives," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 202–209.

[12] E. Heien, D. Kondo, A. Gainaru, D. LaPine, B. Kramer, and F. Cappello, "Modeling and Tolerating Heterogeneous Failures in Large Parallel Systems," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 45:1–45:11.

[13] D. W. Hosmer Jr and S. Lemeshow, *Applied logistic regression*. John Wiley & Sons, 2004.

[14] D. Jacobsen, "Nersc procmon," http://www.osti.gov/estsc/details.jsp?rcdid=5344, 2014.

[15] I. Jolliffe, "Principal Component Analysis," in *Wiley StatsRef: Statistics Reference Online*. John Wiley & Sons, Ltd, 2014.

[16] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," in *Machine Learning: ECML-98*, ser. Lecture Notes in Computer Science, C. Ndellec and C. Rouveirol, Eds. Springer Berlin Heidelberg, 1998, no. 1398, pp. 4–15.

[17] Y. Liang, Y. Zhang, M. Jette, A. Sivasubramaniam, and R. Sahoo, "BlueGene/L Failure Analysis and Prediction Models," in *International Conference on Dependable Systems and Networks, 2006. DSN 2006*, Jun. 2006, pp. 425–434.

[18] N. Nakka, A. Agrawal, and A. Choudhary, "Predicting Node Failure in High Performance Computing Systems from Failure and Usage Logs," in *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, May 2011, pp. 1557–1566.

[19] J. R. Quinlan, "Induction of Decision Trees," *Mach Learn*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[20] C. Reiss and J. Wilkes, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, 2011.

[21] F. Salfner, M. Lenk, and M. Malek, "A Survey of Online Failure Prediction Methods," *ACM Comput. Surv.*, vol. 42, no. 3, pp. 10:1–10:42, Mar. 2010.

[22] B. Schroeder and G. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 337–350, Oct. 2010.

[23] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson *et al.*, "Addressing failures in exascale computing," *International Journal of High Performance Computing Applications*, 2014.

[24] W. Yoo, K. Larson, L. Baugh, S. Kim, and R. H. Campbell, "ADP: automated diagnosis of performance pathologies using hardware events," in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE*, vol. 40. New York, New York, USA: ACM, Jun. 2012, pp. 283–294.

[25] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster Computing with Working Sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. USENIX, 2010.