Filtering Log Data: Finding the Needles in the Haystack

Li Yu, Ziming Zheng, Zhiling Lan
Illinois Institute of Technology
Chicago, IL 60616, USA
{lyu17, zzheng11, lan}@iit.edu

Terry Jones
Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA
trjones@ornl.gov

Jim M. Brandt, Ann C. Gentile
Sandia National Laboratories
Livermore, CA 94551, USA
{brandt, gentile}@sandia.gov

***Abstract*—Log data is an incredible asset for troubleshooting in large-scale systems. Nevertheless, due to the ever-growing system scale, the volume of such data becomes overwhelming, bringing enormous burdens on both data storage and data analysis. To address this problem, we present a 2-dimensional online filtering mechanism to remove redundant and noisy data via feature selection and instance selection. The objective of this work is two-fold: (i) to significantly reduce data volume without losing important information, and (ii) to effectively promote data analysis. We evaluate this new filtering mechanism by means of real environmental data from the production supercomputers at Oak Ridge National Laboratory and Sandia National Laboratory. Our preliminary results demonstrate that our method can reduce more than 85% disk space, thereby significantly reducing analysis time. Moreover, it also facilitates better failure prediction and diagnosis by more than 20%, as compared to the conventional predictive approach relying on RAS (Reliability, Availability, and Serviceability) events alone.**

I. Introduction

As supercomputers continue to grow in size and complexity, reliability becomes a major concern in the field of high performance computing (HPC). Because of this, modern systems are deployed with various monitoring and logging facilities to track system health and status during operations [1, 2]. For example, the environmental monitors deployed on IBM Blue Gene systems can collect data like temperatures, clock frequency, fan speeds, and voltages, from the underlying hardware devices [3]; the OVIS monitoring tool developed from Sandia National Lab can collect various state variables (e.g., temperature, CPU utilization, fan speed) and user-specified variables (e.g., aggregated memory errors over the life span of a job) on various large-scale clusters [4]. Log data is a critical asset to successfully operating large-scale systems. System administrators typically examine these data to spot faults or errors, and numerous technologies are presented to utilize log data for fault prediction and root cause analysis [5, 6, 7, 8, 9].

Nevertheless, because of the scale of supercomputers and the fine granularity of logging tools, an overwhelming amount of data are often collected in a very short period of time and enormous storage capacity is typically required for archiving these log data. For example, OVIS collects ∼10 GB of data per day when monitoring just the hardware of Sandia *Glory* cluster of 288 servers with 16 cores each (4600 cores total) and data taken on a 10 second interval [1]. For a larger system like the new Japanese K machine containing 705K cores (150+ times larger than Glory), it could collect ∼1.5TB/day of information. Future machines are projected to have even more components, thus archiving the raw environmental data is problematic and how to save space becomes critical.

Given the huge amount of data collected, human operators have to spend tremendous amount of time and effort to scan through these data for useful information. Such manual processing is time-consuming, error-prone, and impractical. In the past, a number of analytical algorithms have been presented which can be used to assist human to automatically diagnose system behaviors. Nevertheless, the large data volume and their complicated internal relations pose a major obstacle to many data analysis methods, because an analytical algorithm may take hours to extract important information from a large amount of data. As a result, although log data are valuable source, they often become useless files that just fill up disk space simply because we have too little time to review too much information. This raises the first key question: *how to significantly reduce the volume of log data like environmental data without losing important information*?

Due to the problems caused by the overwhelming amount of data, many systems turn off their monitoring facilities or only collect and archive critical events (e.g., RAS events or console events). These events are often generated when abnormal readings are encountered by environmental monitors, therefore they are much smaller in size [3]. This leads to a common practice, where most existing failure prediction and diagnosis studies rely on critical events alone for analyzing system behaviors. For example, the commonly used prediction methods (e.g., association rules, statistical rules, or combinations of various basic learners) typically examine causal correlations among fatal and non-fatal events to discover failure patterns [7, 8, 10, 11, 9]. There are at least two limitations of these studies. First, despite the great effort to improve prediction accuracy, the best accuracy achieved by these approaches is typically lower than 80%. Second, critical events generally do not contain sufficient information regarding root causes of failures. Together, these raise the second key question: *can we improve data analysis like prediction accuracy and diagnosis accuracy by utilizing environmental data*?

In this paper, we intend to answer the above two questions by presenting a 2-dimensional online data filtering mechanism to remove noisy and redundant data *horizontally* (via feature selection) as well as *vertically* (via instance selection). Figure 1 gives an overview of our filtering design. Our design comprises three major components. First is *data integration*, where various data streams from different sources are synchronized and fed to a buffer in memory. Once the buffer is filled, the data is moved out for further operations and the buffer is vacated to receive new data. Second is *feature selection*, where data filtering is conducted *horizontally*, meaning representative features are selected and the rest is dropped. Third is *instance selection*, where data filtering is performed *vertically* along the time axis, meaning representative instances are selected and the rest is dropped.

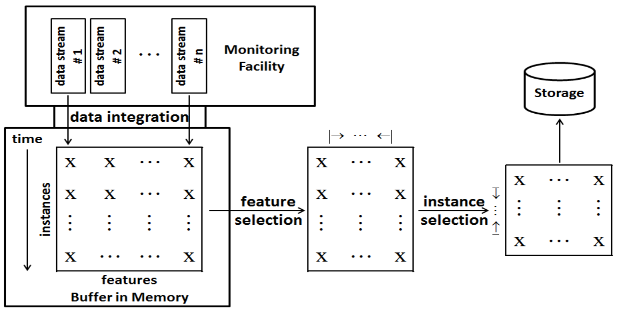


Fig. 1. An overview of our 2-dimensional online filtering mechanism. The texts in red represent the major components of our design.

To truly make this filtering mechanism useful in realistic environments, our design intends to provide several key features. First, our design provides online filtering of log data, inspired by the idea of stream processing. Second, the design is general and flexible, where a variety of selection techniques can be integrated in this framework. Third, the design can be used with data compression technologies for further data reduction.

To demonstrate the effectiveness of our filtering design, we conduct case studies on two real environmental data collected from the production supercomputers (an environmental log from a Blue Gene/P system at Oak Ridge National Lab and an OVIS log from a cluster at Sandia National Lab). We examine the amount of disk storage that can be reduced by applying our online filtering mechanism. We also compare the effects of our filtering mechanism as against random filtering. Further, we study whether the proposed filtering mechanism brings a positive or negative impact on failure prediction and diagnosis. The use of multiple data is to ensure the presented mechanism is not biased to any specific system or log and thus is general for providing filtering service for a variety of log data.

For the Blue Gene/P environmental data, our method can reduce $\sim 85.6\%$ disk space without losing prediction accuracy and root cause information. For the OVIS log, we can achieve $\sim 99.7\%$ disk space saving without losing both prediction accuracy and root cause information. To the best of our knowledge, we are among the first to explore 2-dimensional filtering (i.e., horizontally as well as vertically) for reducing system logs like environmental data and utilize the filtered data for better failure prediction and diagnosis on large-scale systems.

While we compare prediction and diagnosis accuracy on the filtered and the raw data in our experiments, the focus of this work is not to compare different predictive or diagnostic methods. Instead, our goal is to provide a general online filtering framework for large-scale systems. We believe many predictive and diagnostic methods can benefit from this work.

The organization of this paper is as follows. Section II describes the system logs used in our case studies. Section III presents the details of our 2-dimensional filtering mechanism. Section IV presents our case studies using the 2-dimensional filtering mechanism. Section V discusses related work. Finally, Section VI summarizes this paper.

II. System logs

As mentioned earlier, we will evaluate our filtering design through two case studies. In this section we provide a background description of these logs used in our case studies. Table I gives a summarization.

System	Period	Log	Size
Eugene (2-rack BlueGene/P)	3 months	RAS log	0.8 GB
		Environmental log	6 GB
Glory (288 nodes, 4068-core cluster)	2 weeks	Console log	186 KB
		OVIS log	11 GB

TABLE I
A summarization of logs used in our case studies.

A. Eugene and Its Environmental Data

Our first case study is based on a three-month environmental log and its corresponding RAS log from the production Blue Gene/P system named *Eugene* at Oak Ridge National Laboratory. *Eugene* is a 2-rack Blue Gene/P system with the standard Blue Gene/P configuration, in which the 2 racks are laid in 2 rows (i.e., R0 to R1). The system consists of 8,192 compute nodes with a total number of 32,768 cores, offering a peak performance of 27.9 TFlops [12]. On Blue Gene/P, the environmental monitor reads status information from the cards and stores these information in the environmental database. The environmental monitor also generates RAS events when abnormal readings are encountered [3].

In our study, the environmental data are categorized and stored in 11 tables, each of which represents one of following components in *Eugene*: BULKPOWER, FAN, CLOCKCARD, LINKCARD, LINKCARDPOWER, LINKCHIP, NODECARD, NODECARDPOWER, NODE, SERVICECARD and SERVICECARDPOWER. Each table has different number of features such as voltage, current, temperature, etc., and the total number of features is $3,214$. The RAS events are used to label the system status that has four levels: "INFO","WARN","ERROR" and "FATAL", with increasing severity. Table II gives an example of the environmental data for a component and Table III gives an example of the RAS events.

Time	Location	MaxTemp1	MinTemp1	...
1.5856e8	R00-M0-N5	32	29	...
1.7356e8	R00-M1-N1	30	27	...
1.7356e8	R01-M1-N7	30	26	...

TABLE II
ENVIRONMENTAL DATA FOR A COMPONENT FROM *Eugene*

Time	Location	Severity	...
1.4856e8	R00-B-P2	WARN	...
1.5356e8	R00-M0-A9	ERROR	...
1.5356e8	R01-M0-N1-J06	FATAL	...

TABLE III
RAS EVENTS FROM *Eugene*

B. Glory and Its OVIS Data

Our second case study is based on the OVIS data collected from *Glory* at Sandia National Laboratories [4]. *Glory* is a 288-node, 4068-core Opteron cluster with an Infiniband interconnect. Similar to the environmental data in BlueGene/P, the fundamental system data collected by OVIS are stored separately into 51 tables, each of which represents a combination of component type and metric. For example, the table name "MetricCnCPUTempValues" indicates the table stores CPUTemp values of the component with type "cn". In this study, the 51 type-metric combinations are collected per node based on a one-minute sampling rate and the total number of features is 14,688. Also, the console records during the same period are used to label the system status. Table IV gives an example of the OVIS data for a type-metric combination and Table V gives an example of the console records.

TableKey	CompId	Value	Time
1	106	315536	2009-02-11 16:05:27
2	173	402619	2009-02-12 11:21:14
3	227	246136	2009-02-18 11:59:38

TABLE IV
OVIS DATA FOR A TYPE-METRIC COMBINATION FROM *Glory*

console.glory131:2009-02-11 15:42:38 Out of memory: ...
console.glory165:2009-02-27 02:04:03 APP4 invoked oom-killer: ...
console.glory181:2009-02-11 15:36:35 APP26 invoked oom-killer: ...

TABLE V
CONSOLE RECORDS FROM *Glory*

III. METHODOLOGY

As shown in Figure 1, our online filtering mechanism consists of three major components: data integration, feature selection and instance selection. The main reason of applying feature selection before instance selection is to reduce runtime overhead. We have tested different orderings and the effects on data reduction and accuracy are trivial.

A. Data Integration

The function of data integration is to synchronize data streams from various sources (sensors). Typically, monitoring tools adopt a fixed sampling rate to collect environmental data in the system. However, although collected at the same time point, data from different sources are usually reported with asynchronous timestamps and some of them are even missed accidentally. In order to facilitate online filtering, data integration works as a preprocess to guarantee a complete snapshot at each sampling timestamp.

In this study, we use a simple method to synchronize data streams from different sources. The time axis is divided into consecutive intervals with equal length which is the same as the sampling rate of the system. For example, if the monitoring tool on the Blue Gene/P system collects environmental data every five minutes, the length of the time interval is set to five minutes as well. In each interval, if complete data are obtained, they are used to form an instance and then added to the buffer; otherwise, they are considered corrupted and then dropped. In our case studies, the amount of incomplete instances is very limited: less than 0.5% in the BlueGene/P log and less than 1.5% in the OVIS log, hence removing them hardly affects experimental results. Once buffer is filled, the data in the buffer form a matrix, which is shown in the middle of Figure 2. Each row of the matrix is a snapshot of all data sources (sensors), while each column represents feature values from a single source.

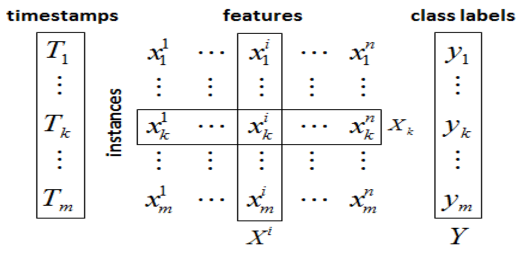


Fig. 2. An example of the matrix formed in the buffer, based on which feature selection and instance selection are performed. Each instance X_k has a timestamp T_k and a class label y_k that can be obtained from the RAS log or console log.

In our experiments, the logs were collected in an off-line manner, data integration is simulated by simply moving a time window, with the same length as the buffer, through the log data. Since all the following steps rely on the matrix given in Figure 2, we give the meanings of notations that will be used in the rest of the paper. Let the $m*n$ matrix in the figure be X, where m is number of instances and n is number of features. We use X^i $(i = 1, ...n)$ to represent a feature (state variable), where n is equal to the number of data sources (sensors) of the monitoring facilities. Similarly, we use X_k $(k = 1, ...m)$ to represent an instance in the buffer, where the value of m is determined by the size of the buffer and the sampling rate. For example, if the buffer size is set to 100 minutes and the system is based on a 5 minutes sampling rate, then the value of m is 100/5=20, which means the reduction method runs on only 20 instances each time. Additionally, each instance X_k has a timestamp T_k (indicating its collected time) and a class label y_k (indicating corresponding system status).

B. Feature selection

Given the matrix from the buffer, feature selection is used to filter out features which are less informative in depicting system status while keeping more representative ones, thus reducing the data volume. In this paper, we choose feature ranking as the selection strategy because of its simplicity and scalability. The basic idea of feature ranking is to compute a scoring function for features and rank them in a decreasing order of their scores. Features with higher scores are typically considered more important than those with lower scores and are selected.

The class label y_k is needed for feature selection. Specifically, we need to select X^is that are more informative in depicting Y. Feature selection needs a training stage that is performed offline. The class label y_k can be obtained by searching the event log (see Table III and V) using the timestamps of instances. y_k is either 1 for normal or 0 for abnormal. Two ranking functions are adopted in this study.

1) Pearson correlation based ranking: The first ranking function is the square of Pearson correlation coefficient, which is also called the coefficient of determination [13]. This scoring function ranks features according to their correlations with the class. Using the notations in Figure 2, it is defined as:

$$R_i^2 = \frac{cov(X^i, Y)^2}{var(X^i)var(Y)} \ (1 \le i \le n),$$

where cov represents the covariance and var indicates the variance. Given m instances, the estimate of R_i^2 can be computed by:

$$R_i^2 = \frac{(\sum_{k=1}^{m}(x_k^i - \overline{x_i})(y_k - \overline{y}))^2}{\sum_{k=1}^{m}(x_k^i - \overline{x_i})^2 \sum_{k=1}(y_k - \overline{y})^2}, \tag{1}$$

where $\overline{x_i} = \sum_{k=1}^{m} x_k^i \ / \ m$ and $\overline{y} = \sum_{k=1}^{m} y_i \ / \ m$.

2) Information gain based ranking: The second ranking function is information gain, which is defined as the reduction in uncertainty about the class Y when the feature X^i is known [14]. Features with higher information gain contain more information about the class Y than those with lower information gain. The information gain of a feature X^i with respect to the class Y is given as:

$$IG_Y(X^i) = H(Y) - H(Y|X^i) \tag{2}$$

Using the notations in Figure 2, $H(Y)$, $H(Y|X^i)$ can be computed by:

$$H(Y) = -\sum_{k=1}^{k=m} P(Y = y_k) log_2(P(Y = y_k))$$

$$H(Y|X^i) = -\sum_{k=1}^{k=m} P(X^i = x_k^i) H(Y|X^i = x_k^i))$$

Note the above formulas require discrete variables. In our study, as the environmental variables (e.g., voltage, current and temperature) are usually continuous, they need a discretization process before being introduced into the above formulas.

Based on the two ranking functions described above, we develop Algorithm 1 for feature selection. Instead of selecting features from the entire feature space, it picks the top N_f features where N_f is in proportion to the fraction of features in that component. The component here refers to the environmental component for BlueGene/P and the type-metric combination for OVIS (See Section II). Since this algorithm selects at least one feature from each component arbitrarily, it may not be optimal in terms of space saving, but it ensures that the filtered data will preserve information from every component which is critical for maintaining similar or higher prediction and diagnosis accuracy.

Algorithm 1 Feature Selection

```
Let $q$ be the number of features to be selected
S $\leftarrow \emptyset$
$n_i \leftarrow$ number of features in each component
$w_i \leftarrow \frac{n_i}{\sum_{i=1}^{n} n_i}$
for each component C do
    Rank features in C using Equation (1) or (2)
    S $\leftarrow$ S $\bigcup$ top $N_f$ features in C with $N_f = q \cdot w_i$
end for
```

C. Instance selection

Similar to feature selection, the goal of instance selection is to choose instances that are more informative toward analyzing system behaviors. In particular, the instances are categorized into two groups: the one representing normal system status and the one indicating anomalies. It is obvious that the abnormal status related instances are more informative than the normal ones in terms of system behavioral analysis. For example, during a period of time, 100 instances are generated by the system, all of which are related to normal status. To record system status at this period, one out of 100 instances might be sufficient. If we rule out the rest 99 instances in this period, we get a 99% storage space reduction. On the contrary, we do not want to miss any instances related to abnormal system status; otherwise, we lose important information for system behavioral analysis. In this study, we examine two methods for instance selection.

1) Statistic based selection: The basic idea of statistic based selection is derived from the assumption that instances associated with abnormal system status usually diverge from those related to normal status. This method compares a set of randomly selected features in each instance in the current buffer to their statistics from the last buffer. Figure 3 shows an illustration for the statistics of features. Here, statistics could be mean, variance, and many others (we use mean for simplicity in this study). Only those instances whose features deviate far from the statistics in the last buffer are selected. Using the notations in Figure 2 and Figure 3, the statistic based instance selection method is shown in Algorithm 2.

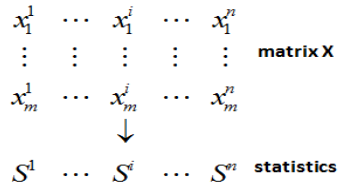


Fig. 3. An illustration for the statistics of features.

Algorithm 2 Statistic based Instance Selection

```
Let $X$ be the $m * n$ matrix in the current buffer
Let $T_{st}^1$ be the number of features for comparison
Let $T_{st}^2$ be the maximal statistical deviation
Let $T_{st}^3$ be the maximal number of deviated features
$F \leftarrow$ randomly select $T_{st}^1$ features from $X$
$S^i$ be the statistic of $F^i$ $(i = 1, ..., T_{st}^1)$ in the last buffer
A $\leftarrow \emptyset$
for each $X_k$ $(k = 1, ..., m)$ in current buffer do
    count $\leftarrow$ 0
    for each $F_k^i$ and $S^i$ $(i = 1, ..., T_{st}^1)$ do
        if $\parallel F_k^i$-$S^i \parallel \geq T_{st}^2$ then
            count = count + 1;
        end if
    end for
    if count$\geq T_{st}^3$ then
        A $\leftarrow$ A $\bigcup X_k$
    end if
end for
```

2) KD-tree based selection: A KD-tree is an index structure which has been used for fast nearest neighbor search [15]. It stems from the simple binary search tree. The KD-tree, instead of splitting 1-dimensional instances, segments instances in a multi-dimensional space, with each cut focusing on only one direction of the feature space. The root of the tree represents all the instances. Each interior node is associated with a splitting feature X^i and a splitting value V^i $(1 \leq i \leq n)$. Those instances with X^i greater than V^i are put into the left child while the others are put to the right child. To apply KD-tree for instance selection, we first set the maximal number of instances contained by each tree leaf, and then cut recursively till this predefined value is reached, finally a portion of instances in each leaf are selected based on the comparison between the statistics of current buffer and the last buffer.

Unlike the statistic based selection that compares each instance in the current buffer against the statistics of the last buffer, KD-tree based selection compares the current buffer as a whole. We use an adaptive selection rate to control the number of instances selected from each tree leaf. If the statistics of the current buffer deviate far from the previous one, the selection rate is set to a high value; otherwise, it is set to a low value. As the buffer with exceptional statistics is assumed to contain more abnormal instances, by using the adaptive selection rate, we can avoid missing informative instances effectively. Also, the statistics used may vary depending on the characteristics of log data. Using the notations in Figure 2, our KD-tree based instance selection method is shown in Algorithm 3.

Algorithm 3 KD-tree based Instance Selection

```
Let $X$ be the $m * n$ matrix in current buffer
Let $T_{kd}^1$ be the maximal leaf size
Let $T_{kd}^2$ be the selection rate
R $\leftarrow X_k$ $(k = 1, ..., m)$
A $\leftarrow \emptyset$
Build the KD-tree K using R as the root
for each leaf L of K do
    A' $\leftarrow$ randomly select $T_{kd}^2$ instances from L
    A $\leftarrow$ A $\bigcup$ A'
end for
```

IV. Experiments

A. Evaluation Metrics

The goal of our online filtering mechanism is to significantly reduce data volume without losing important information for failure prediction or diagnosis. Hence, we choose following metrics to evaluate our filtering mechanism.

1) Space Savings: To measure to what extent the data has been reduced by our method, we use the metric called *space savings* that is defined as below:

$$\begin{aligned} Space\ Savings\ (S) &= 1 - \frac{Filtered\ Size}{Raw\ Size} \\ &= 1 - ISR * FSR \end{aligned}$$

Since our method reduce data both horizontally and vertically, ISR (Instance Selection Ratio) is defined as the ratio of number of instances after instance selection to the number of instances before the selection, indicating to what extent the data has been reduced vertically. FSR (Feature Selection Ratio) is defined in the same manner for feature selection.

2) Prediction Accuracy: We evaluate whether our filtering mechanism can improve data analysis in two aspects. First, we compare prediction accuracy on the filtered data as against that on the raw data. In our experiments, failure prediction uses the following model: a predictive method analyzes the data occurring during an observation window, and aims to predict whether a failure will occur after a lead time [16]. Lead time is the time interval preceding the time of failure occurrence. To be practical, lead time is supposed to be long enough to perform a desired proactive fault prevention. *Prediction Accuracy* is defined as

$$Prediction\ Accuracy\ (A) = \frac{2TP}{2TP + FP + FN}$$

Note the prediction accuracy here is actually $F\_measure$, a commonly used metric to evaluate prediction methods [17], where $True\ Positive\ (TP)$ is the number of correctly predicted abnormal instances; $False\ Positive\ (FP)$ is the number of normal instances that are predicted as abnormal; and $False\ Negative\ (FN)$ is the number of abnormal instances that are predicted as normal.

Three predictive methods are evaluated, including Logistic Regression, MLP (Multilayer Perceptron) and SVM (Support Vector Machine) [18]. A logistic regression classifier provides conditional probability classifications of input data based on the logistic regression model; an MLP generates a nonlinear mapping between the input data and the output for decision making; a SVM model represents the data as points, which are mapped to a space so that the points belonging to separate classes are divided by a gap as wide as possible. While these classifiers differ in their classification mechanisms, they provide similar prediction results in our case studies. Hence, in the rest of the paper, we only present experimental results by using SVM.

3) Diagnosis Accuracy: We also assess our filtering method by examining whether it misses important information with regards to root cause analysis. In general, the exact failure occurrence time and location are two key information for failure diagnosis [19]. As a result, the filtering method is effective only if the filtered data preserve these information.

Since data filtering is conducted along two directions, it takes the risk of losing root cause information along both directions. First, if an instance referring to an abnormal status is ruled out during instance selection, we lose the corresponding time information. Second, if the feature reporting the failure is removed during feature selection, we lose the location information about the failure. Hence we define the metric *Diagnosis Accuracy* to evaluate whether our filtering mechanism holds important information (i.e., time and location):

$$\begin{aligned} &Diagnosis\ Accuracy \\ &\qquad = Time\ Info\ Preservation \\ &\qquad * Location\ Info\ Preservation \end{aligned}$$

Time Info Preservation is the ratio of the number of abnormal instances (i.e., instance labeled abnormal status) in the filtered data to the number in the raw data, *Location Info Preservation* is defined as the ratio of abnormal locations (i.e., sensors that report abnormal status) covered by the selected features to those covered by all features. A 100% *Diagnosis Accuracy* means the filtering method can preserve all important information. Note that abnormal locations can be obtained from the RAS log or the console log in our case studies.

B. Case Study 1: Blue Gene/P Log

We conducted four sets of experiments on the Blue Gene/P log. In the first set, we study different combinations of instance selection and feature selection methods and analyze the best achievable *Space Savings*, along with the corresponding prediction accuracy values. In the second set, we evaluate prediction accuracy with varying *Space Savings*s and compare different combinations of selection methods in terms of prediction accuracy. In the third set, we focus on diagnosis accuracy with different instance selection and feature selection ratios. In the last set, we compare our selection methods as against random selection. Together, these extensive sets of experiments intend to provide us a clear picture of the effectiveness of online filtering.

		Instance Selection		
		Kd-tree	Statistic	None
Feature Selection	Info gain	99.83% (S) 96.64% (A)	99.84% (S) 94.75% (A)	99.57% (S) 95.35% (A)
	Pearson	99.83% (S) 96.64% (A)	99.84% (S) 94.75% (A)	99.57% (S) 95.35% (A)
	None	59.88% (S) 95.91% (A)	62.27% (S) 95.35% (A)	0.0% (S) 93.24% (A)

TABLE VI
Space Savings (S) & *Prediction Accuracy* (A) ON THE BLUE GENE/P DATA. HERE, THE "NONE" COLUMN (OR ROW) LISTS THE RESULTS WITHOUT INSTANCE (OR FEATURE) SELECTION.

In the first set of experiments, we set lead time to 10 minutes and use failure prediction accuracy on the raw data as the baseline. We aim to identify the best achievable *Space Savings* of filtering when prediction accuracy is no less than the baseline. From the experimental results shown in Table VI, we can make two important observations. First, the 2-dimensional filtering can significantly reduce storage requirement (over 99% of *Space Savings*). Further, by comparing the reduction effects, for this system log, the majority of reduction is achieved by feature selection. For example, using instance selection alone, we can only achieve about 60% *Space Savings*, compared to more than 99% *Space Savings* using feature selection alone. This is caused by the nature of log data collected from Blue Gene/P. On one hand, a number of ERROR and FATAL events are reported in three months, so our filtering method needs to keep a plenty of instances to cover all anomaly related ones.

On the other hand, most ERROR and FATAL events occur in a small number of components, therefore only 0.8% of the features are sufficient to represent the abnormal status.

Next, the table clearly shows that prediction based on environmental data can achieve high accuracy (with F-measure higher than 93%). According to our literature survey [8, 7, 5] as well as previous experience [9, 20, 21], the best prediction accuracy achieved by failure prediction using RAS events is typically lower than 80%. The main reason is that RAS logs only contain limited information about the underlying system and its operating environment [6]. For example, in Blue Gene systems, RAS logs only contain limited environmental information based on the results from environmental monitor [3]. Environmental data logs contain more information about the underlying system, which significantly help failure predictor to capture a variety of fault symptoms. In addition, the filtering process can remove redundant and noisy data, thereby improving prediction accuracy further. In summary, this set of experiments indicates that the presented 2-dimensional filtering can not only significantly reduce log size (over 99% of *Space Savings*), but also substantially improve prediction accuracy (with F-measure higher than 94.75%).

In the second set of experiments, we study prediction accuracy with varying *Space Savings* and compare different combinations of selection methods in terms of prediction accuracy. In Figure 4, we plot a 3D figure to illustrate the trend of prediction accuracy with different instance selection ratios and feature selection ratios, where prediction lead time is set to 10 minutes. Due to space limit, we only present the plot using statistic based instance selection and information gain based feature selection. In Figure 5, we present prediction accuracy under different *Space Savings* (from 99.88% to 0%). The plot contains four curves, representing different combinations of instance selection and feature selection methods.

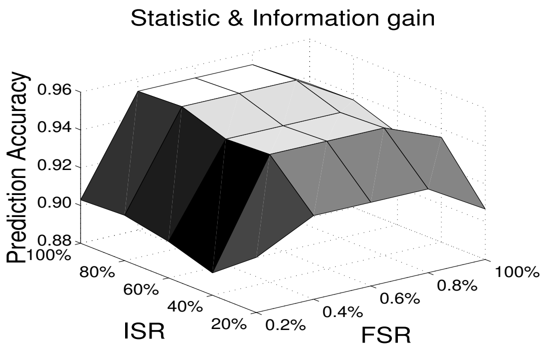


Fig. 4. Prediction accuracy on the Blue Gene/P data with different combinations of instance selection ratios (using statistic based selection) and feature selection ratios (using information gain based selection). Note the 100% ISR (FSR) indicates no filtering on the instances (features).

There are several interesting observations from Figure 4. On one hand, given the same instance selection ratio, when feature selection ratio is very small (e.g., when we remove 99.8% of the features), prediction accuracy is lower than that using

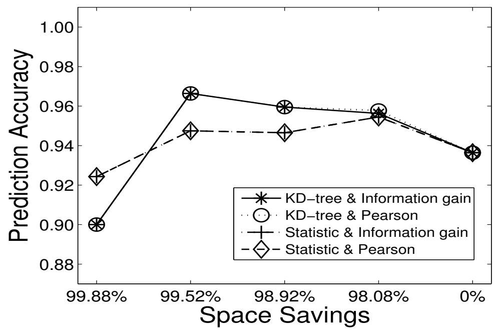


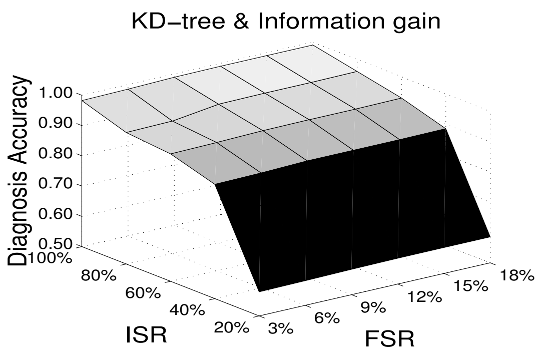
Fig. 5. Prediction accuracy with different *Space Savings* on the Blue Gene/P data. There are four curves in the plot, indicating different combinations of instance selection methods and feature selection methods. Note the 0% *Space Savings* indicates no filtering on the data.

the original data. This indicates that if we carelessly remove too many features, we may get rid of important information for failure prediction. Nevertheless, prediction accuracy can improve dramatically when we keep a little bit more features. For instance, as long as we preserve 0.4% or higher amount of the features, we can achieve better prediction accuracy as comparing to the case without applying feature selection. Also, F-measure starts to drop slowly when FSR grows beyond 0.8%. This observation implies that the filtering process can remove noisy data from the raw log, thereby improving prediction accuracy. As the number of selected features increases, more noises are included in the data and consequently deteriorate prediction accuracy. On the other hand, given the same feature selection ratio, the impact of instance selection ratio on prediction accuracy is not significant. This is because the removing of redundant instances does not substantially change the the abnormal patterns.

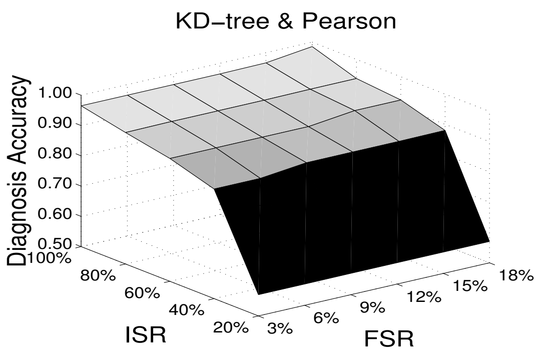
Further, as shown in Figure 5, two feature selection methods perform similarly, whereas the two instance selection methods do not. Initially when *Space Savings* is high, *Statistic based instance selection* outperforms *KD-tree based instance selection*. As *Space Savings* decreases, *KD-tree based instance selection* exceeds *Statistic based instance selection* and finally they perform similarly when *Space Savings* is smaller than 98.08%. Nevertheless, we shall point out the difference between different selection methods is generally less than 3% in terms of prediction accuracy.

In the third set of experiments, we examine diagnosis accuracy under different instance selection ratios and features selection ratios (see Figure 6). As stated in the previous subsection, diagnosis accuracy measures whether our filtering can preserve important information like time and location for root cause analysis. A good filtering engine should achieve a high value (close to 1.0).

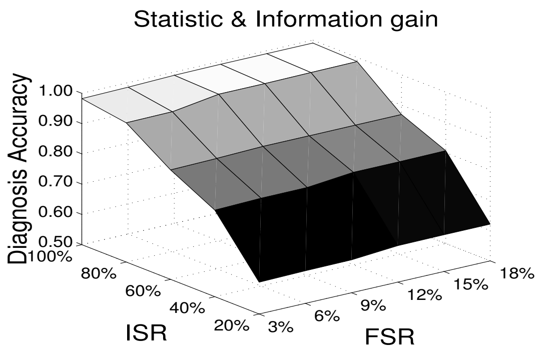
Although the two feature selection methods do not make much difference in terms of prediction accuracy, they perform differently in terms of diagnosis accuracy. The *Information gain based selection* always outperforms the *Pearson based*



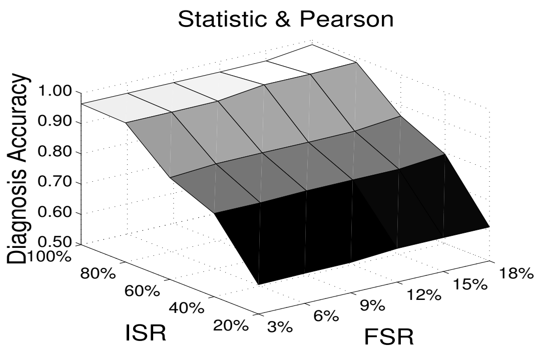
(a)



(b)



(c)



(d)

Fig. 6. Diagnosis accuracy on the Blue Gene/P data with different combinations of instance selection ratios and feature selection ratios. Note the 100% ISR indicates no filtering on the instances.

selection. Next, when comparing the two instance selection methods, we can see that statistic based method can preserve important time and location information with lower FSR. This indicates that statistic based method can remove more instances (i.e., providing higher *Space Savings*), without losing diagnosis accuracy, as compared to KD-tree based method. Moreover, when we compare the results shown here and that in Figure 4, we can see that root cause diagnosis put more restriction than failure prediction with regards to data filtering. For the Blue Gene/P data, we can achieve about 85.6% *Space Savings* without losing important information for root cause analysis, whereas we can obtain about 99.84% *Space Savings* without losing important information for failure prediction.

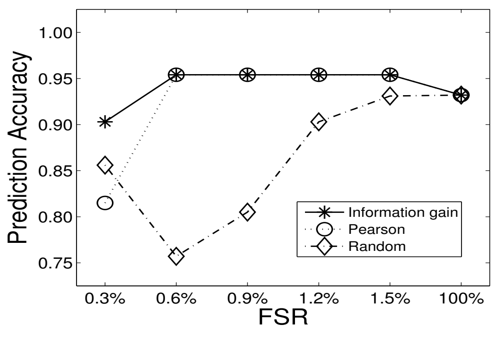
In the fourth set of experiments, we compare our selection methods as against random selection. According to previous experiments, since feature selection plays the major role in terms of data reduction (as shown in Table VI), we give the comparison of our feature selection methods and the random feature selection based on both prediction accuracy and diagnosis accuracy (Figure 7). Note that feature selection only filters out features, which only removes the location information without influencing the time information.

As shown in Figure 7, the two feature selection methods outperform the random feature selection in terms of both prediction accuracy and diagnosis accuracy. Using prediction accuracy as the metric, neither our feature selection methods nor the random feature selection performs well when the number of selected features is small (see Figure 7(a)). This is because too much information is missed. As the number of selected features increases, the two feature selection methods outperform the random one significantly (e.g., with 0.6% FSR or higher). It indicates that our feature selection methods can effectively select features that are strongly correlated with the system symptoms. Based on diagnosis accuracy, the disparity between our features selection methods and the random selection is more distinct. As FSR increases from 0.3% to 1.5%, the randomly selected features can hardly capture any abnormal locations (see Figure 7(b)). On the contrary, the two feature selection methods can cover more than 90% abnormal locations with only 0.3% FSR, suggesting their high diagnosis accuracy. Comparing Figure 7(a) with 7(b), we can see diagnosis accuracy is more sensitive to feature selection strategies than prediction accuracy.

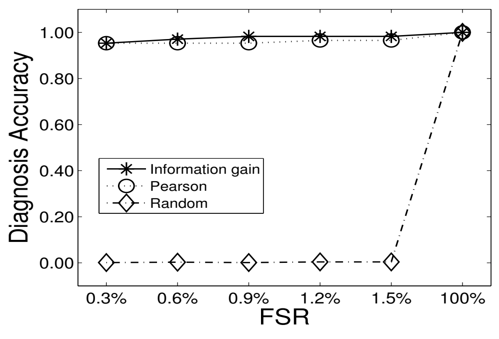
C. Case Study 2: OVIS Log

In the second case study, we apply our filtering mechanism to the OVIS data and examine whether it can obtain similar benefits observed on the Blue Gene/P log. We also conduct four sets of experiments, similar to those conducted on the Blue Gene/P data.

In the first set of experiments, we also set lead time to 10 minutes and use prediction accuracy on the raw data as the baseline. Table VII shows the best achievable *Space Savings* using different combinations of selection methods.



(a)



(b)

Fig. 7. Comparison of feature selection methods with random selection on the Blue Gene/P data. Note the 100% FSR indicates no filtering on the features.

By comparing Table VII with Table VI, we have a key observation. That is, unlike the Blue Gene/P data where feature selection plays a major role in terms of *Space Savings*, both instance selection and feature selection can greatly reduce data volume on the OVIS data. For example, on the Blue Gene/P data, instance selection can achieve about 62.27% of *Space Savings* without losing prediction accuracy; on the OVIS data, removing 97.32% of the instances (i.e., 2.68% of ISR) can still achieve even higher prediction accuracy. We guess this is because the number of abnormal instances in the OVIS data is limited and these instances occur in a burst manner. Thus a substantial amount of instances related to normal system status are filtered out by instance selection.

		Instance Selection		
		Kd-tree	Statistic	None
Feature Selection	Info gain	99.95% (S)	99.81% (S)	93.88% (S)
		100% (A)	96.43% (A)	95.08% (A)
	Pearson	99.95% (S)	99.51% (S)	87.76% (S)
		100% (A)	95.08% (A)	95.08% (A)
	None	97.32% (S)	91.12% (S)	0.0% (S)
		100% (A)	95.08% (A)	95.08% (A)

TABLE VII
Space Savings (S) & *Prediction Accuracy* (A) ON THE OVIS DATA. HERE, THE "NONE" COLUMN (OR ROW) LISTS THE RESULTS WITHOUT INSTANCE (OR FEATURE) SELECTION.

In the second set of experiments, we examine prediction accuracy with varying selection ratios. Figure 8 shows the results, using a 10 minutes lead time. By comparing it with Figure 4, we find that feature selection reduces significantly less ratio of data on the OVIS log than on the Blue Gene/P log (i.e., 4% FSR versus 0.4% FSR) without losing prediction accuracy. This is partially caused by the weighted feature selection mechanism described in Algorithm 1, according to which a larger portion of features is selected from the OVIS data than from the Blue Gene/P data. Note that since the different combinations of selection methods show similar trends in terms of prediction accuracy as observed on the Blue Gene/P data, we do not show the results here due to the space limit. Overall speaking, our filtering mechanism can obtain more than 99.7% *Space Savings* without losing both prediction and diagnosis accuracy on the OVIS data (using Statistic based instance selection & Information gain based feature selection).

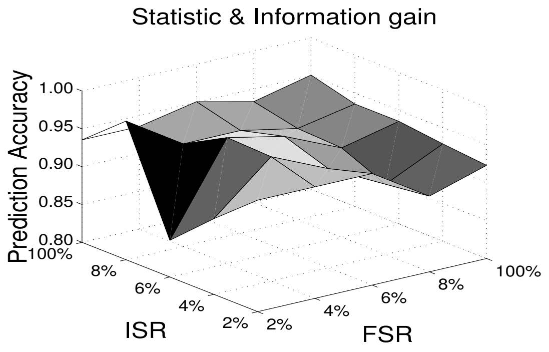


Fig. 8. Prediction accuracy on the OVIS data with different combinations of instance selection ratios (using statistic based selection) and feature selection ratios (using information gain based selection). Note the 100% ISR (FSR) indicates no filtering on the instances (features).

In the third set of experiments, diagnosis accuracy under different selection ratios is examined. Due to space limit, we only show one combination of instance selection method and feature selection method in Figure 9. An interesting observation can be made by comparing Figure 9 with Figure 6(c). On the OVIS data, 6% ISR and 4% FSR can achieve 100% diagnosis accuracy, as compared to 80% ISR and 18% FSR on the Blue Gene/P data. A possible explanation is that the OVIS log contains much less noise than the Blue Gene/P log, hence our filtering mechanism can obtain better coverage of faulty locations.

In the fourth set of experiments, the comparison of our selection methods with random selection shows similar results as obtained on the Blue Gene/P data. Due to limited space, we only show the comparison of our instance selection methods with the random instance selection in terms of diagnosis accuracy in Figure 10. Again, our instance selection methods have proven to be much more effective than the random

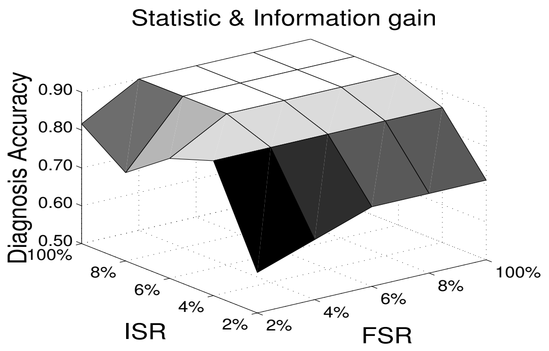


Fig. 9. Diagnosis accuracy on the OVIS data with different combinations of instance selection ratios (using statistic based selection) and feature selection ratios (using information gain based selection). Note the 100% ISR (FSR) indicates no filtering on the instances (features).

selection. Both methods can achieve 100% diagnosis accuracy with 6% ISR.

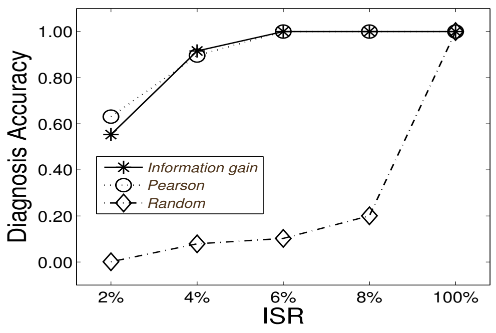


Fig. 10. Comparison of instance selection methods with random selection on the OVIS data. Note the 100% ISR indicates no filtering on the instances.

D. Analysis Time Reduction

To further demonstrate the benefits brought by our filtering mechanism, in this additional set of experiments, we run four commonly used analytic algorithms on the filtered data as well as on the raw data. The goal is to assess time savings that we can achieve by filtering data. SVM and Logistic are two predictive methods that have been widely used to find failure patterns in the dataset. They have been used in our case studies to measure prediction accuracy (described in Section IV-A as "SVM" and "Logistic Regression"). Subset evaluation is usually used to select useful features among the dataset [14]. Unlike the feature selection methods used in our case studies (i.e., Pearson based and Information gain based feature selection), it evaluates the worth of a subset of features by considering the individual predictive ability of each feature along with the degree of redundancy between them. K-Means is a commonly used method for data clustering, through which similar features in the dataset can be grouped for better diagnosis [18]. All the results given in this set of experiments are based on a platform that installs a Intel Quad 2.83GHz processor with 8GB memory, running windows 7 professional 64-bit operating system.

System Logs		Analytic algorithms			
		SVM	Logistic	Subset	K-Means
BG/P	T_f	158.6	40.1	20.5	18.9
	T_u	675.5	171.2	189.8	335.7
	$(T_u$-$T_f)/T_u$	**76.5%**	**76.6%**	**89.2%**	**94.4%**
OVIS	T_f	0.6	3.2	2.3	2.6
	T_u	4.8	34.6	105.7	243.8
	$(T_u$-$T_f)/T_u$	**87.5%**	**90.8%**	**97.8%**	**98.9%**

TABLE VIII
THE RUNNING TIME (IN SECONDS) OF ANALYTIC ALGORITHMS ON THE FILTERED DATA (T_f) AND ON THE UNFILTERED DATA (T_u) AND THE RELATIVE TIME REDUCTION. THE FILTERED BLUE GENE/P DATA HAS 85.6% *Space Savings* AND THE FILTERED OVIS DATA HAS 99.7% *Space Savings*.

Table VIII gives the comparison of running time (in seconds) and the relative time reduction. The *Space Savings* of the filtered data are 85.6% (Blue Gene/P) and 99.7% (OVIS) respectively, meaning the highest achievable value without losing both prediction and diagnosis accuracy. The results show that our filter mechanism makes big gain in terms of running time of analytic algorithms. An interesting observation is that all algorithms have significantly shorter running time on the OVIS data than on the Blue Gene/P data, although the OVIS data is larger in size.

All of these four algorithms are based on iterative refinement technologies, hence the time reduction on the filtered data comes from two aspects. First, the time for each iteration shortens as the data is reduced in size. Second, the number of iterations decreases because of the removing of noisy and redundant information, meaning a faster convergence of the algorithm. This also explains why these algorithms run faster on the OVIS data although it is larger in size.

We do not include algorithms whose running time rely only on the size of data in this set of experiments because their time reductions are easy to estimate. For example, the time reduction of an O(n) algorithm will have a linear relation with the *Space Savings* of the filtered data.

E. Result Summary

How Much Runtime Overhead Is Introduced? The overhead mainly comes from training, which is offline. After constructing selection rules, the actual runtime selection process is trivial: feature selection is instantaneous with almost no delay; instance selection is dependent on the buffer that typically contains dozens of instances, and thus the cost is normally within a couple of seconds.

Feature Selection or Instance Selection? While instance selection and feature selection may have different impact on different log data, we believe both methods are necessary for removing redundant and noisy data.

KD-tree or Statistic? The former usually results in better prediction accuracy, while the later is preferred for time information preservation.

Information gain or Pearson? They are comparable in terms of prediction accuracy, but the former works better for location information preservation.

V. Related Work

Data reduction. Data reduction technologies have been widely used for system performance monitoring and analysis. Existing studies can be broadly classified as instance based methods and feature based methods. On one hand, instance-based methods generally identify representative instances and discard the instances with redundant information. Considerable research on instance-based methods has been performed on system log analysis in large-scale systems. Buckley Reed et. al presented Tupling methods to coalesce related events [22]; Liang et. al adopted temporal and spatial filtering to remove redundant records of RAS logs [23]; Reed et. al used event throttling to prevent the generation of large data volumes from parallel computer systems [24]; Zheng et. al presented a causality-related filtering method to pinpoint the sets of fatal events co-occurring frequently and filter them together [25]. On the other hand, feature-based methods aim to select or extract a subset of relevant features for robust learning. Yang et. al conducted a two-stage selection strategy to identify subsets of system metrics and showed that only 20% of the metrics is sufficient to describe application behavior [26]; Zhang et. al proposed a method to predict compliance using an ensemble of Bayesian network models in a dynamic environment, in which feature selection was used to select subset of metrics most closely related to model data relations [19]; Mendes and Reed applied dynamic clustering and statistical sampling methods to select subsets of processors or metrics [27]; Lan et. al adopted PCA (principal component analysis) ICA (and independent component analysis) to extract fault related features for anomaly identification [16].

While many data reduction techniques have been presented to date, to the best of our knowledge there is no known effort on building a systematic support for effectively reducing environmental data on HPC systems for significant storage saving and improved data analysis. Furthermore, our work distinguishes from the above studies at two key aspects. First, our work aims to reduce log data through both directions feature selection and instance selection. Second, we propose a general framework for online filtering of log data on HPC systems, which can be easily extended with various feature selection and instance selection techniques.

Failure prediction and root cause diagnosis. Recognizing the importance of reliability on HPC systems, considerable studies have been conducted on failure prediction and root cause analysis. Sahoo et al. applied association rules to predict failure events in a IBM cluster [7]; Liang et al. examined several data mining and machine learning techniques for failure forecasting in a Blue Gene/L system [8]; Zhang et. al explored three different prediction methods and evaluated them on Blue Gene/L [5]; Zheng et. al explored a dynamic meta-learning prediction engine in large-scale systems [9]. Most of existing studies mainly focused on analyzing RAS events for failure prediction and/or root cause analysis.

Nevertheless, RAS logs only contain limited information about the underlying system and its operating environment, which is inadequate in understanding failures and system behaviors [6]. Unlike these studies, this work discusses online filtering of environmental logs containing data like temperatures, clock frequency, fan speeds, and voltages, from the running hardware devices. These logs contain more information about the underlying system, yet they can easily overwhelm us with data. Our proposed methods can significantly reduce the size of environmental logs without losing important information for failure prediction and root cause analysis. Hence, this work not only motivates the long-term archive of filtered environmental data, but also stimulates the research of using environmental data for improved failure prediction and root cause analysis.

Feature and instance selection have been integrated for text classification in [28]. Our work is inspired by the Huan's work [29] and Thompson's work [30]. In [29], a KD-tree based sampling method was proposed for informative instance selection. In this paper, we modify the KD-tree algorithm to rule out instances irrelevant to abnormal states. In [30], Thompson utilized two feature extraction methods on numeric data for failure prediction in the Blue Gene/P system, which was based on the same dataset in our first case study. However, their study was only on the feature level, which was insufficient for data reduction. Meanwhile, their extraction methods could lead to information loss for root cause diagnosis.

VI. Conclusion

In this paper we have presented an online filtering framework for environmental logs containing the data like temperatures, clock frequency, fan speeds, and voltages. These logs contain abundant information about the underlying system status, yet they can easily overwhelm us with data. The goal of our work is to remove redundant and noisy data from these logs both horizontally (via feature selection) and vertically (via instance selection). Our filtering mechanism can not only significantly save disk space without losing important information for failure prediction and diagnosis, but also open doors for more analysis algorithms that are suitable for large-scale systems like the TOP500 machines [31]. Our case studies, by means of real environmental logs from the production supercomputers, have demonstrated that the proposed filtering framework can achieve more than 85% storage *Space Savings*. Moreover, it also facilitates better failure prediction and diagnosis by up to 20%, as compared to the conventional predictive approach relying on RAS events alone. We are currently working on integrating this framework with the OVIS monitoring tool [4]. We believe the development of such a filtering mechanism will motivate the long-term archive of environmental data at HPC centers and open doors for more analytic algorithms that are suitable for future exascale systems. It will further promote failure prediction and root cause analysis in the field as well.

Acknowledgment

The work at Illinois Institute of Technology was supported in part by US National Science Foundation grants CNS-0834514, CNS-0720549. The work at ORNL was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. The work at Sandia was supported by the United Stated Department of Energy, Office of Defense Programs. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

References

[1] J. Brandt, A. Gentile, J. Mayo, P. Pebay, D. Roe, D. Thompson, and M. Wong, "Resource monitoring and management with OVIS to enable hpc in cloud computing environments," in *Proc. of IPDPS*, 2009, pp. 1–8.

[2] M. Li and Y. Zhang, "Hpc cluster monitoring system architecture design and implement," in *Proc. of ICICTA*, 2009, pp. 325–327.

[3] G. Lakner, "IBM system Blue Gene solution: BlueGene/P system administration," *IBM*, 2007.

[4] J. Brandt, A. Gentile, C. Houf, J. Mayo, P. Pebay, D. Roe, D. Thompson, and M. Wong, "OVIS 3.2 user's guide," *Sandia National Laboratories*, 2010.

[5] Y. Zhang and A. Sivasubramaniam, "Failure prediction in IBM BlueGene/L event logs," in *Proc. of IPDPS*, 2008, pp. 1–5.

[6] A. Oliner and J. Stearly, "What supercomputers say: A study of five system logs," in *Proc. of DSN*, 2007, pp. 575–584.

[7] R. Sahoo, A. Oliner, I. Rish, M. Gupta, J. Moreira, and S. Ma, "Critical event prediction for proactive management in large-scale computer clusters," in *Proc. of SIGKDD*, 2003, pp. 426–435.

[8] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, and R. Sahoo, "BlueGene/L failure analysis and models," in *Proc. of DSN*, 2006, pp. 425–434.

[9] Z. Lan, J. Gu, Z. Zheng, R. Thakur, and S. Coghlan, "A study of dynamic meta-learning for failure prediction in large-scale systems," *JPDC*, vol. 70, pp. 630–643, 2010.

[10] R. Vilalta and S. Ma, "Predicting rare events in temporal domains," in *Proc. of ICDM*, 2002, p. 474.

[11] B. Schroeder and G. Gibson, "A large-scale study of failures in high performance computing systems," in *Proc. of DSN*, 2006, pp. 249–258.

[12] S. Alam, R. Barrett, M. Bast, M. Fahey, J. Kuehn, C. McCurdy, J. Rogers, P. Roth, R. Sankaran, J. Vetter, P. Worley, and W. Yu, "Early evaluation of IBM BlueGene/P," in *Proc. of SC*, 2008, pp. 1–12.

[13] D. Ozer, "Correlation and the coefficient of determination," *PSYCHOL BULL*, vol. 97, pp. 307–315, 1985.

[14] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *JMLR*, vol. 3, pp. 1157–1182, 2003.

[15] A. Moore, "An introductory tutorial on kd-trees," *Technical Report No.209, Computer Laboratory, University of Cambridge*, 1991.

[16] Z. Lan, Z. Zheng, and Y. Li, "Toward automated anomaly identification in large-scale systems," *IEEE Trans. Parallel Distrib.*, vol. 21, pp. 174–187, 2010.

[17] C. V. Rijsbergen, *Information Retrieval*. Butterworth-Heinemann, 1979.

[18] E. Alpaydin, *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2004.

[19] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons, and A. Fox, "Ensembles of models for automated diagnosis of system performance problems," in *Proc. of DSN*, 2005, pp. 644–653.

[20] Z. Zheng, Z. Lan, R. Gupta, S. Coghlan, and P. beckman, "A practical failure prediction with location and lead time for BlueGene/P," in *Proc. of DSNW*, 2010, pp. 15–22.

[21] L. Yu, Z. Zheng, Z. Lan, and S. Coghlan, "Practical online failure prediction for Blue Gene/P: Period-based vs event-driven," in *Proc. of PFARM*, 2011, pp. 259–264.

[22] M. Buckley and D. Siewiorek, "Comparative analysis of event tupling schemes," in *Proc. of FTCS*, 1996, p. 294.

[23] Y. Liang, Y. Zhang, A. Sivasubramanium, R. Sahoo, J. Moreia, and M. Gupta, "Filtering failure logs for a BlueGene/L prototype," in *Proc. of DSN*, 2005, pp. 476–485.

[24] D. Reed, R. Aydt, R. Noe, P. Roth, K. Shields, B. Schwartz, and L. Tavera, "Scalable performance analysis: The pablo performance analysis environment," in *Proc. of SPLC*, 1993, pp. 104–113.

[25] Z. Zheng, Z. Lan, B.-H. Park, and A. Geist, "System log pre-processing to improve failure prediction," in *Proc. of DSN*, 2009, pp. 572–577.

[26] L. Yang, J. Schopf, C. Dumitrescu, and I. Foster, "Statistical data reduction for efficient application performance monitoring," in *Proc. of CCGrid*, 2006, pp. 327–334.

[27] C. Mendes and D. Reed, "Monitoring larger systems via statistical sampling," *Int. J. High Perform. Comput.*, vol. 18, pp. 267–277, 2004.

[28] D. Fragoudis, D. Meretakis, and S. Likothanassis, "Integrating feature and instance selection for text classification," in *Proc. of KDD*, 2002, pp. 501–506.

[29] H. Liu, H. Motoda, and L. Yu, "Feature selection with selective sampling," in *Proc. of ICML*, 2002, pp. 395–402.

[30] J. Thompson, , D. Dreisigmeyer, T. Jones, M. Kirby, and J. Ladd, "Accurate fault prediction of BlueGene/P RAS logs via geometric reduction," in *Proc. of DSNW*, 2010, pp. 8–14.

[31] "Top500 list: List of top 500 supercomputers." [Online]. Available: http://http://www.top500.org/