System Log Pre-processing to Improve Failure Prediction

Ziming Zheng, Zhiling Lan
Illinois Institute of Technology
{zzheng11, lan}@iit.edu

Byung H. Park, Al Geist
Oak Ridge National Laboratory
{parkbh, gst}@ornl.gov

Abstract

Log preprocessing, a process applied on the raw log before applying a predictive method, is of paramount importance to failure prediction and diagnosis. While existing filtering methods have demonstrated good compression rate, they fail to preserve important failure patterns that are crucial for failure analysis. To address the problem, in this paper we present a log preprocessing method. It consists of three integrated steps: (1) event categorization to uniformly classify system events and identify fatal events; (2) event filtering to remove temporal and spatial redundant records, while also preserving necessary failure patterns for failure analysis; (3) causality-related filtering to combine correlated events for filtering through apriori association rule mining. We demonstrate the effectiveness of our preprocessing method by using real failure logs collected from the Cray XT4 at ORNL and the Blue Gene/L system at SDSC. Experiments show that our method can preserve more failure patterns for failure analysis, thereby improving failure prediction by up to 174%.

Keywords: log preprocessing, event categorization, event filtering, Cray XT4, IBM Blue Gene/L

1. Introduction

Fueled by the ever-growing scale and complexity of computer systems, failures become ongoing facts of life to be dealt with in large-scale systems. Recent studies have shown that in production systems, failure rates are as high as more than 1000 per year, and depending on root cause of the problem, the average failure repair time ranges from a couple of hours to nearly 100 hours [6].

Recognizing the dramatic impact of failures on system productivity, an increasing attention has been paid to failure prediction and a variety of predictive methods have been presented in the recent years [5, 4, 17]. System logs provide a rich source of information for failure prediction. Unfortunately, system logs cannot be directly used by various prediction technologies due to the fact that they generally contain too many redundant information and are often unstructured for data analysis. In this paper, we present *a log preprocessing methodology to improve failure prediction*. Preprocessing is a process applied on the raw log before applying a prediction method. Log preprocessing not only cleans and formalizes the training data for discovering failure patterns, but also extracts necessary events for failure forecasting. *The goal of this study* is to provide an effective preprocessing methodology to distill system events for better failure prediction in large-scale systems such as high-end supercomputers [9].

Despite the crucial role of log preprocessing, existing preprocessing techniques are often ad hoc and mainly concentrate on compression rate. Temporal and spatial filtering are commonly used to remove redundant records in system logs [2]. While these preprocessing techniques have high compression rate, e.g., up to 99.96%, *they suffer from three major drawbacks*. First, they might remove important failure patterns, namely a long stream of warnings preceding the failure. As will be shown later, such a pattern embeds invaluable information for failure analysis. Second, when an event occurs across multiple locations, spatial filtering removes this trace of events. Since spatial filtering often keeps one event, this event may contain a location which is different from the source of failure. Thus it might lead to wrong analysis. Third, they ignore the fact that a failure may be reported from multiple subsystems characterizing different aspects of the failure. While these records may have different syntax, they are causally related. Preprocessing these records independently could lead to wrong results.

To address the above issues, in this study we present a log preprocessing method which contains three tightly-coupled steps:

1. *Event Categorization.* Regular expression technology is adopted to classify various events into a hierarchical set of event categories.

2. *Event Filtering.* An improved temporal and spatial filtering method is proposed to remove redundant records. Different from existing filtering methods [1, 2], our filtering method keeps track of event start and the end times, event count, and event location. This addresses the first and the second issues listed above.

3. *Causality-related Filtering.* Apriori association rules are adopted to track causal correlations among events. Rather than performing filtering on causally-related events independently, we suggest to combine correlated events for filtering. This helps to preserve failure patterns for better data analysis, which addresses the third issue listed above.

We demonstrate the effectiveness of our preprocessing methodology by means of system logs collected from two production systems, i.e., the Cray XT4 system at Oak Ridge National Laboratory (ORNL) and the Blue Gene/L system at San Diego Supercomputing Center (SDSC). Experiments show that it can effectively preserve failure patterns and consequently improve failure prediction by up to 174%, with a compression rate of more than 90%.

2 Background

The Cray XT4 at ORNL, named *Jaguar*, is ranked #5 on the TOP500 supercomputer list (June, 2008) [9]. It has $7,832$ XT4 compute nodes, in addition to I/O and login service nodes. Each compute node contains a quad-core 2.1 GHz AMD Opteron processor and 8GB of memory. Aggregated system performance is approximately 263 teraflops. Approximately 600 terabytes are available in the scratch file systems. Each node is connected to a Cray SeaStar router through HyperTransport, and the SeaStars are all interconnected in a 3-D-torus topology. More detailed documents of the system architecture can be found in [7]. An example of event record from the system RAS log is shown in Table 1. RAS events are collected at a granularity of one second. Each record consists of five fields. CRMS event type indicates the high-level category of the events. Both *SRC* and *SVC* are about the source of the problem. The SVC usually provides more detailed information. The entry field provides a description of the event.

The Blue Gene/L system at SDSC consists of three racks, with a total of 3,072 compute nodes and 384 I/O nodes. Each compute node consists of two 700 MHz PowerPC processors that share 512 MB of memory. The aggregated peak speed is 17.2 teraflops and the total memory is 1.5 terabytes. More details of the system architecture is available in the literature [8]. An example of event record from the system RAS log is shown in Table 2. Each record contains eight fields. *Event Type* specifies the mechanism through which the event is recorded. *Facility* indicates the services/hardware component that has experienced the event. *Event Time* is the time stamp associated with the reported event. *Job ID* identifies the job that detects the event. *Location* denotes the source of the event from which chip, node-card, service-card or link-card. *Entry Data* gives a brief description of the event. One major difference from Cray XT4 log is that Blue Gene/L log provides explicit *Severity* information. *Severity* could be INFO, WARNING, SEVER, ERROR, FATAL or FAILURE.

We have acquired two RAS logs from these systems. Table 3 summarizes the logs. As we can see from the table, the RAS log from the Cray XT4 system is substantially larger than that from the Blue Gene/L system. This is due to the fact that the machine has more number of nodes. Sample data of the logs are available at http://www.cs.iit.edu/˜zlan/sample_log/.

3 Event Categorization

This step aims at providing a standard categorization of RAS events by analyzing their syntax. If two events have the same syntax, we group them into one category for data analysis. *Regular expression* is the standard technique to analyze the syntax. It involves extracting distinct keywords and then using *concatenation*, *alternation* and *Kleene star operations* to generate the syntax for each category [15].

We adopt *a hierarchical approach* for event categorization. That is, we first divide system events into several high-level classifications, and then further group events into a number of subcategories based on manual investigation on their contents. For Cray XT, nine high-level categories are identified based on the *CRMS Event Type* field, which are further divided into 52 low-level event types; for Blue Gene/L, ten high-level categories are identified based on the *Facility* field, which are further divided into 293 low-level event types.

In addition to provide a fine-granular categorization, it is also necessary to distinguish these event categories into *fatal* or *non-fatal* groups for the purpose of data training. Non-fatal events indicate system warnings or information messages, while fatal events refer to those critical events that lead to system or application crashes. Although RAS logs in Blue Gene/L provide severity level for each event, it is not accurate as some fatal or failure events are not truly fatal at all [1]. By working with system administrators, we have identified and removed some of these events from the fatal list. Totally, there are 83 fatal events for the Blue Gene/L system. Examples include cache failure (CF), DDR register failure (DRF), interrupt failure (IF), power hardware failure (PHF) and link failure (LF).

Cray XT4 RAS logs do not provide such severity information. By consulting with system administrators at ORNL, we have identified ten types of fatal events. They are link failure fault (LFF), node heartbeat fault (NHF), node failed fault (NFF), service failed fault (SFF), seastar hearbeat fault (SHF), node health check fault (NHC), VERTY health check fault (VHC), RX message CRC error (RXM), RX message head CRC error (RXH) and L0 voltage fault (L0V).

Table 1. An example of event from Cray XT4 RAS log.

Event time	CRMS Event Type	SRC	SVC	Entry
2007-08-01 12:25:00	ec_mesh_link_failed	src:::c22c0s4	svc:::c2-2c0s4s0	c22c0s4s0l5=S

Table 2. An example of event from Blue Gene/L RAS log.

Rec ID	Event Type	Facility	Severity	Event Time	Job ID	Entry Data	Location
17366	RAS	KERNEL	INFO	2004-12-10-13 .52.57.333932	14	3 ddr errors(s) detected and corrected on rank 0, symbol 35, bit 3	R00-M0 -N4-C9-U11

Table 3. Summary of RAS logs from the Cray XT4 and the Blue Gene/L systems.

Log Name	Days	Start Date	End Date	Log Size (GB)	No. of Records
Cray XT4	206	2007-05-05	2007-11-27	45	160063372
Blue Gene/L	1110	2004-12-06	2007-12-12	1.84	511331

4 Event Filtering

The purpose of event filtering is to remove redundant records. An optimal filtering should not only achieve high compression rate, but also introduce low information loss. This leads to two commonly raised questions: (1) which records are redundant? and (2) what information should be kept? The first question has been discussed in [1]-[4]. Broadly speaking, there are two types of redundant records. The first type is defined from *a temporal view*. When the system detects an anomaly, it keeps producing warnings until the failure occurs. Similarly, when a failure occurs, it may re-appear multiple times in RAS logs before its root problem is solved. Temporal filtering can help to remove this redundancy by removing the same type of events being reported from the same location within $T_{temporal}$ seconds. The second type comes from *a spatial view*. Many jobs running on large-scale systems are parallel applications. When they are running on multiple nodes, any warning or failure record can be generated from multiple locations. Spatial filtering can help to remove this redundancy by removing similar events being reported at different locations within $T_{spatial}$ seconds.

Regarding the second question, i.e., what information should be kept during filtering, existing methods mainly rely on ad hoc techniques. When an event is continuously reported, existing filtering methods typically keep the first record and remove subsequent ones. This may eliminate information that are crucial for analyzing causal correlations among events. For instance, on the Cray XT4 log, by using such a filtering process, 29% of fatal events will be found without any precursor events. One example is *service failed fault (SFF)* occurred at 2007-05-05 11:29:52. The filtered log contains no event preceding this event within the past 59 minutes. However, when we checked the raw log, we found that there is a *uPacket squash fault (USF)* event occurred 51 seconds before the SFF event. This USF event was filtered out since 810 instances of the USF event occurred continuously from 2007-05-05 10:30:00 to 2007-05-05 11:29:01. Only the first record is kept and others are filtered out by using existing filtering techniques. In other words, the pattern that a long stream of warnings occur before a failure is filtered out.

To address the problem, we propose an improved filtering method. In addition to record the event, it also preserves *event start time, event end time, event count, and event locations*. For instance, suppose we set the filtering window to be 1.0 minute, then for the above SFF event, the newly proposed filtering method will not only record the USF event, but also record its start time as of 2007-05-05 10:30:00, its end time as of 2007-05-05 11:29:01 and event count as of 810. Further, our method also keeps the *location* information. For example, if USF is reported on both c9-2c2s0s0 and c9-0c2s7s3, both locations are kept for the event in the filtered log. This can assist us in studying failure propagation and identifying the failure source.

How to decide an optimal threshold for filtering is still an open question. In this study, we adopt an *iterative approach* [12, 13]. We first set the threshold to a very small number, and then gradually increase the number. The search stops when there is no significant change with respect to compression rate. Our experiments show that for the Cray XT4 log, the optimal threshold is 60 seconds, which achieves 99.97% compression rate. For the Blue Gene/L log, the optimal threshold is 300 seconds, which achieves 99.83% compression rate.

5 Causality-Related Filtering

A failure may be reported by multiple subsystems in different forms. While these records may have different syntax, they have the same semantics. We define it as *semantic redundancy*, which cannot be removed by existing filtering techniques or the event filtering technique presented in Section 4.

Semantic redundancy can lead to wrong analysis result. For instance, it can lead to a lower value for Mean-Time-Between-Failures. Furthermore, it might hide the

Table 4. A sequence of records from the Blue Gene/L system.

Rec ID	Event time	Category	Entry
786421	2007-08-24-05.25.24.800071	link failure	Link PGOOD error latched on link card
786422	2007-08-24-05.26.17.563519	link failure	Link PGOOD error latched on link card
...	...	...	...
786428	2007-08-24-05.48.29.446392	link failure	Link PGOOD error latched on link card
786429	2007-08-24-05.53.52.006698	power hardware failure	power module status fault detected on node card. status registers are: 0/0/1/0
786430	2007-08-24-05.56.54.804122	link failure	Link PGOOD error latched on link card
786431	2007-08-24-05.57.43.486897	link failure	Link PGOOD error latched on link card
...	...	...	...
786437	2007-08-24-06.19.37.766449	link failure	Link PGOOD error latched on link card
786438	2007-08-24-06.24.39.051317	power hardware failure	power module status fault detected on node card. status registers are: 0/0/1/0

Table 5. Exemplar transactions from Cray XT4.

Transaction ID	List of Failure Events
1	L0V, NHF,SHF,NFF
2	NHF
3	L0V, NHF,SHF,NFF
4	RXM, RXH
...	...
n	L0V, NHF,SHF,NFF

root cause of the problem. An example is shown in Table 4. As we can see, the system always reports several link failure followed by a power hardware failure. If we use existing temporal and spatial filtering methods, both the record #786429 and #786438 will be kept as independent events. Even worse, if the threshold of 900 seconds is used for event filtering, only the link failure #786421 would be kept in the log. Since the end time of the link failure is not kept, a predictive method will consider the event #786421 to be far away from #786429 and #786438. This will lead to wrong results.

To address the problem, we propose *apriori association rule mining* [11] to identify the sets of fatal events co-occurring frequently and filter them together. Suppose that $[T^f_{A,s}, T^f_{A,e}]$ and $[T^f_{B,s}, T^f_{B,e}]$ represent the start-end periods of fatal events A and B respectively. A window size of W^f is defined to measure the gap between two events. If $[T^f_{A,s} - W^f, T^f_{A,e} + W^f] \bigcap [T^f_{B,s}, T^f_{B,e}] \neq \emptyset$, then A and B are considered as a transaction. Further, the relation is transitive. That is, if $[T^f_{A,s} - W^f, T^f_{A,e} + W^f] \bigcap [T^f_{B,s}, T^f_{B,e}] \neq \emptyset$ and $[T^f_{B,s} - W^f, T^f_{B,e} + W^f] \bigcap [T^f_{C,s}, T^f_{C,e}] \neq \emptyset$, then A, B and C will be combined as one transaction. Exemplar transactions from the Cray XT4 are shown in Table 5.

Two parameters are used to measure whether events are causally-related or not. One is *confidence*, which measures whether two events are co-occurring in all the transactions:

$$\mathit{confidence}(A \rightarrow B) = \frac{P(AB)}{P(A)} \qquad (1)$$

Suppose there are n transactions, m transactions contain event A and r transactions contain both A and B, then $P(A) = m/n$ and $P(AB) = r/n$. In the other words, $\mathit{confidence}(A \rightarrow B)$ is the conditional probability of B when A occurs.

The other is called *lift*, which measures the correlation between A and B in all the transactions as follows:

$$\mathit{lift}(A, B) = \frac{P(AB)}{P(A)P(B)} \qquad (2)$$

For example, if $\mathit{lift}(A, B)$ is larger than a predefined threshold, then we consider that the co-occurrence of A and B in a transaction is not by coincidence, but by their causal correlation.

Our causality-related filtering measures whether (1) $\mathit{confidence}(A \rightarrow B) = \mathit{confidence}(B \rightarrow A) = 1$, or (2) $\mathit{confidence}(A \rightarrow B) = 1$ and $\mathit{lift}(A, B) > 2$, or (3) $\mathit{confidence}(B \rightarrow A) = 1$ and $\mathit{lift}(A, B) > 2$. If any of the conditions is satisfied, the events A and B will be combined for filtering. More specifically, we keep A and B as a combined event, and apply the event filtering technique as presented in the previous section. As an example, the events #786421 and #786438 shown in Table 4 are considered causally-related.

For the Cray XT4 log, three sets are found to be correlated: (1) NHF, NFF, SHF, RXM, RXH and L0V; (2) NHC and VHC; (3) LFF and SFF. Similarly, for the Blue Gene/L log, three sets are found to be correlated: (1) cache failure, DDR address register failure, DDR Info register failure and interrupt failure; (2) data address failure, data store failure and exception syndrome failure; (3) power hardware failure and link failure.

6 Experiments

Our experiments are conducted to evaluate whether our preprocessing methods (denoted as *CFC*) can preserve useful failure patterns for better failure prediction, as against using existing spatial and temporal filtering method (denoted as *ST*) [2]. A standard 10-fold cross-validation method is used for the learning and testing. Two metrics are used to evaluate failure prediction: *precision* (i.e., proportion of correct predictions to all the predictions made) and

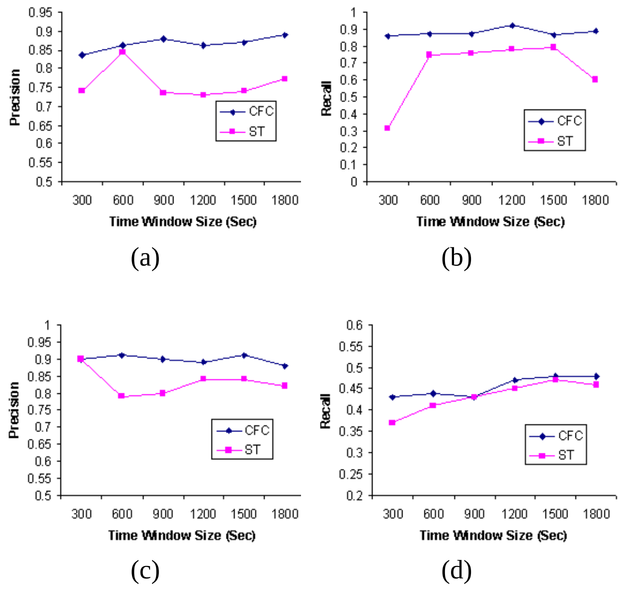

Figure 1. Impact on failure prediction: (a)-(b) from the Cray XT4; and (c)-(d) from the Blue Gene/L. The *CFC* curve represents the results on the cleaned log produced by our preprocessing method, and the *ST* curve represents the results on the cleaned log produced by existing temporal-spatial filtering method.

recall (i.e., proportion of correct predictions to the number of failures).

We have tested several prediction methods, including *decision tree*, *back propagation neural network*, and *Bayesian belief network*, on the cleaned logs produced by using our preprocessing method CFC and by using existing filtering method ST. Due to the space limit, here we only present the results with the decision tree, and the results with the other methods are very similar.

Figure 1 present prediction results by using decision trees. Each plot contains two curves, each representing the results achieved on the cleaned log produced by our preprocessing method CFC or by using existing filtering method ST. Given that a window is often used to specify how far ahead to check precursor events for failure prediction, in our experiments the window is set to 300, 600, 900, 1200, 1500, and 1800 seconds respectively.

For the Cray XT, both *precision* and *recall* are always above 0.85 on the cleaned log generated by our method CFC, whereas they are typically below 0.75 on the cleaned log produced by existing method ST. The relative improvement on failure prediction is between $20\% - 174\%$. Further, we notice that prediction accuracy on the cleaned log produced by existing method ST is not consistent, with the results oscillating dramatically between $0.30 - 0.80$. By examining the logs, we find that *the significant improvement on the Cray XT comes from the event filtering step*. In the raw log, a long stream of warnings often occur before each failure event. The ST method removes this pattern from the cleaned log, especially when the prediction window is small, thereby resulting in a large portion of fatal events without any precursor events and leading to a low value on *recall*. Meanwhile, the ST method keeps some of the precursor events that are irrelevant to failures in the cleaned log. This leads to a low value on *precision*. Instead, our preprocessing method CFC can address these issues by applying an improved event filtering and a causally-related filtering.

For the Blue Gene/L, we can observe similar improvements by using our preprocessing method. The relative improvement on failure prediction is between $12\% - 27\%$. By examining the logs, we find that *the improvement on the Blue Gene/L comes from the causally-related filtering step*. The Blue Gene/L log contains a substantial amount of semantic redundancies, which cannot be removed by using the ST method.

We also observe that the *recall* values on the Blue Gene/L log is low (below 0.50), meaning that more than half of failure events cannot be predicted. The main reason is that the log contains about 24% of events without any precursor events, mostly network related. It indicates that the decision tree method alone is not sufficient for effective failure prediction. As pointed out in our previous work [4], in a large-scale system the sources of failures are many and complex, thus it is improbable for a single prediction method to capture all of them alone. Instead, a meta-learning based approach should be applied to combine the strengths of multiple predictive methods for better failure prediction.

In addition, we have also tested our preprocessing mechanism on the Cray log archived in the USENIX Computer Failure Data Repository [18]. By using the CFC preprocessing method with the same experimental parameters, we have obtained a compression rate of 97%. With respect to the improvement on failure prediction, *recall* can be dramatically boosted from 0.3 to 0.7 on the cleaned log produced by the CFC method as against the log generated by the ST method.

7 Related Work

Considerable research efforts have been conducted on system log analysis. For example, a simple event correlator is developed to recognize temporal patterns among failures for web sever logs [15]; a 22-month log is studied for identifying transient and intermittent errors in [10]; a set of log mining techniques are investigated for predicting anomalies in enterprise telephony systems [16]; and Sahoo et al. have evaluated the time-series, the rule-based classification and Bayesian network for failure prediction on an IBM 350-nodes cluster [5]. *Unlike these studies, our paper is more focused on providing an effective pre-*

processing methodology to improve log analysis in large-scale systems. We believe that our preprocessing method can be integrated with the above studies by working on the raw system log and producing a cleaned log for log analysis. Recognizing the critical role of system logs for fault management, the first WASL workshop is organized in December, 2008. The workshop contains many related papers on novel techniques for extracting useful information from existing logs and on methods to improve the information content of future logs [17].

How to effectively preparing system logs for log analysis is a challenging problem and has been neglected in most scientific papers [14]. In [12, 13], tupling methods are presented to coalesce related events for log analysis. *Event cluster* is proposed to deal with multiple redundant records of fatal events at one location [2]. In one cluster, only the first record is kept after filtering. In [3], an adaptive semantic filter (ASF) method is designed to exploit semantic correlation between the events by using temporal gap. Salfner et al. present three preprocessing algorithms to prepare log files fed into the HSMM prediction model used in a commercial telecommunication system [14]. *Compared to these studies, our preprocessing method has three unique features*. First, it not only keeps event type and start/end times, but also records event frequency. This can greatly help to preserve failure re-occurring patterns observed in the raw log. Second, it includes apriori association rule mining to find causally related events and combine them for filtering. This can assist us in achieving better data analysis by preserving casual correlations. Finally, our work is currently focused on high-end supercomputers like Cray XT and Blue Gene (both are pioneering systems in the field of high performance computing). To date, little work has been done on detailed log preprocessing techniques for these systems, especially for Cray XT systems.

8 Conclusions

In this paper, we have presented a log preprocessing method containing three interrelated steps (*event categorization*, *event filtering* and *causality-related filtering*) for large-scale systems. We have evaluated it on the failure logs collected from the Cray XT4 at ORNL and the Blue Gene/L at SDSC, as well as the Cray failure log shared on a public domain [18]. Experimental results have shown that it can not only keep failure patterns in the raw logs for better log analysis, but also provide a satisfactory compression rate.

Acknowledgment

Z. Lan is supported by US NSF grants CNS-0834514, CNS-0720549, and CCF-0702737. The work of B.H. Park and A. Geist is supported by US DOE, Office of Science, Advanced Computing Science Research Division. We would like to thank John White and Eva Hocks at SDSC for helping us on the Blue Gene/L log. This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. The authors like to thank Jeffrey Becklehimer for his help on the Cray events categorization. Many thanks are due to our paper shepherd, Andrea Bondavalli, for his time and guidance.

References

[1] A. Oliner and J. Stearly, "What Supercomputers Say: A Study of Five System Logs," *Proc. of DSN*, 2007.

[2] Y. Liang, Y. Zhang, A. Sivasubramanium, R. Sahoo, J. Moreia, and M. Gupta, "Filtering Failure Logs for a Blue Gene/L Prototype," *Proc. of DSN*, 2005.

[3] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "An Adaptive Semantic Filter for Blue Gene/L Failure Log Analysis," *Workshop on SMTPS*, 2007.

[4] P. Gujrati, Y. Li, Z. Lan, R. Thakur, and J. White, "A Meta-Learning Failure Predictor for Blue Gene/L Systems," *Proc. of ICPP*, 2007.

[5] R. Sahoo, A. Oliner, I. Rish, M. Gupta, J. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam, "Critical Event Prediction for Proactive Management in Large-Scale Computer Clusters," *Proc. of SIGKDD*, 2003.

[6] B. Schroeder and G. Gibson , "A Large-Scale Study of Failures in High Performance Computing Systems," *Proc. of DSN*, 2006.

[7] Cray XT series documents. *http://www.cray.com/products/XT.aspx*,2008.

[8] A. Gara, M. Blumrich et al., "Overview of the Blue Gene/L System Architecture," *IBM J. Res. & Dev.*, vol. 49(2/3), 2005.

[9] The TOP500 supercomputing site. *http://top500.org/*, June 2008.

[10] T. Lin and D. Siewiorek, "Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis," *IEEE Transactions on Reliability*, vol. 39(4), 1990.

[11] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Database," *Proc. of SIGMOD*, 1993.

[12] J. Hansen and D. Siewiorek, "Models for Time Coalescence in Event Logs," *Proc. of FTCS*, 1992.

[13] M. Buckley and D. Siewiorek, "Comparative Analysis of Event Tupling Schemes," *Proc. of FTCS*, 1996.

[14] F. Salfner and S. Tschirpke, "Error Log Processing for Accurate Failure Prediction," *Proc. of WASL'08, in conjunction with OSDI*, 2008.

[15] J. Rouillard, "Real-time Log File Analysis Using the Simple Event Correlator (SEC)," *Proc. of LISA*, 2004.

[16] C. Lim, N. Singh and S. Yajnik, "A Log Mining Approach to Failure Analysis of Enterprise Telephony Systems," *Proc. of DSN*, 2008.

[17] The First USENIX Workshop on the Analysis of System Logs. *http://www.usenix.org/events/wasl08/*, Dec., 2008.

[18] USENIX Computer Failure Data Repository. *http://cfdr.usenix.org/data.html#cray*.