An Adaptive Semantic Filter for Blue Gene/L Failure Log Analysis

Yinglung Liang Yanyong Zhang
ECE Dept.
Rutgers University
{ylliang,yyzhang}@ece.rutgers.edu

Hui Xiong
MSIS Dept.
Rutgers University
hui@rbs.rutgers.edu

Ramendra Sahoo
IBM T. J. Watson
Research Center
rshaoo@us.ibm.com

***Abstract*— Frequent failure occurrences are becoming a serious concern to the community of high-end computing, especially when the applications and the underlying systems rapidly grow in size and complexity. In order to better understand the failure behavior of such systems and further develop effective fault-tolerant strategies, we have collected detailed event logs from IBM Blue Gene/L, which has as many as 128K processors, and is currently the fastest supercomputer in the world. Due to the scale of such machines and the granularity of the logging mechanisms, the logs can get voluminous and usually contain records which may not all be distinct. Consequently, it is crucial to filter these logs towards isolating the specific failures, which can then be useful for subsequent analysis. However, existing filtering methods either require too much domain expertise, or produce erroneous results. This paper thus fills this crucial void by designing and developing an Adaptive Semantic Filtering (ASF) method, which is accurate, light-weight, and more importantly, easy to automate. Specifically, ASF exploits the semantic correlation between two events, and dynamically adapts the correlation threshold based on the temporal gap between the events. We have validated the ASF method using the failure logs collected from Blue Gene/L over a period of 98 days. Our experimental results show that ASF can effectively remove redundant entries in the logs, and the filtering results can serve as a good base for future failure analysis studies.**

I. Introduction

Meta-scale scientific and engineering applications have been playing a critical role in many aspects of the society, such as economies of countries, health development, and military/security. The large processing and storage demands of these applications have led to the development and deployment of IBM Blue Gene/L, the fastest supercomputer on the TOP500 supercomputer list [2]. Blue Gene/L is currently deployed at Lawrence Livermore National Laboratory (LLNL), hosting applications that span several thousand processors, in the domains such as hydrodynamics, molecular dynamics, and climate modeling.

As applications and the underlying platforms scale to this level, failure occurrences have become a norm, rather than an exception [21], [19], [20], [24], [10], [7], [17], [12], [11], [25], [18]. Failures can be broadly categorized into two classes: software failures and hardware failures. Software failures can be further categorized into application software failures such as bugs in application development, and system software failures such as bugs in the kernel domain and system configuration errors. Hardware failures are often observed in memory, storage and network subsystems, but more recently they are also found in combinational units [14], [15]. Both system software failures and hardware failures can have severe impact on the system performance and operational costs. For instance, failures can make nodes unavailable, thereby lowering system utilization. Furthermore, failures can cause applications executing on the nodes (probably having run for a long time) to abort, thus wasting the effort already expended. Additionally, failures can also greatly increase the system management costs. The system administrator may need to detect failures, diagnose the problem, and figure out the best remedial actions. On the hardware end, this may entail resetting a node, changing the motherboard/disk, etc., and on the software end it may require migrating the application, restarting the application, re-initializing/rejuvenating [23] a software module, etc. Indeed, the resulting personnel involvement will increase the Total Cost of Operation (TCO), which is becoming a serious concern in numerous production environments [8], [1].

Understanding the failure behavior in large scale parallel systems is crucial towards alleviating the above problems. This requires continual online monitoring and analysis of events/failures on these systems over long periods of time. The failure logs obtained from such analysis can be useful in several ways. First, the failure logs can be used by hardware and system software designers during early stages of machine deployment in order to get feedback about system failures and performance. It can also help system administrators for maintenance, diagnosis, and enhancing the overall health (uptime). Finally, it can be useful in fine-tuning the runtime system for checkpointing (e.g. modulating the frequency of checkpointing based on error rates), job scheduling (e.g. allocating nodes which are less failure prone), network stack optimizations (e.g. employing different protocols and routes based on error conditions), etc.

While failure logs have the above-mentioned potential uses, the raw logs cannot be directly used. Instead, redundant and/or unimportant information must be first removed. There are several reasons for doing so. First, there are many recorded warnings that do not necessarily lead to a failure. Such warnings need to be removed (note that a sequence of warnings which do lead to a failure should remain). Second, the same error could get registered multiple times in the log, or could get flagged in different ways (e.g. network busy and message delivery error). For instance, we have noticed that a single EDRAM error on Blue Gene/L is record in 20 successive entries, each with a different fatal error code. Consequently, it becomes imperative to filter this evolving data and isolate the specific events that are needed for subsequent analysis. Without such filtering, the analysis can possibly lead to wrong conclusions. For instance, the same software error can materialize on all nodes that an application is running on, and can unduely reduce the Mean Time Between Failures (MTBF) than what is really the case.

We have obtained event logs containing all the RAS (reli-

1-4244-0910-1/07/$20.00 ©2007 IEEE

ability, availability, and serviceability) data from 08/02/05 to 11/18/05 from IBM Blue Gene/L, with a total of 1,318,137 entries. Filtering raw event logs from large-scale parallel systems such as Blue Gene/L is a challenging task, mainly due to the large volume of the data, the complexity of the systems, and the nature of parallel applications. While the spatio-temporal filtering (STF, in short) method that was proposed in [12] has been shown to have the potential to filter out many redundant fatal events, it has the following obvious problems. First, it requires extensive domain knowledge in the organization of the hardware platform, as well as the logging procedure. Second, it involves a considerable amount of manual effort from these domain experts. Third, it imposes challenges in the area of software engineering because STF involves several passes, and after each pass, we need to manually manipulate the partial results for the next usage. Due to these drawbacks, STF is not suitable for online filtering, which is important for many future applications of failure data. Therefore, there is an urgent need to develop some alternative methods to meet these challenges.

To this end, we propose an **Adaptive S**emantic **F**iltering (ASF) method. The key idea behind ASF is to use semantic context of event descriptions to determine whether two events are redundant. In light of this, ASF involves three steps. Firstly, it extracts all the keywords from event descriptions and builds a keyword dictionary. Secondly, it converts every event description into a binary vector, with each element representing whether the description contains the corresponding keyword. Using these vectors, we can compute the correlation between any two events. Thirdly, it determines whether two events are redundant based on their correlation as well as the temporal gap between them. Specifically, the choice of the correlation threshold is not uniform, but varies according to the time window between two events. For example, two events that are temporally close are considered redundant even with low correlation, but two far-away events are only considered redundant when their correlation is very high. Compared to STF and other existing filtering tools, ASF considers both semantic correlation and temporal information, and therefore, the filtering results more accurately capture the real-world situations. More importantly, ASF does not require much human intervention, and can thus be easily automated. Consequently, ASF is one big step forward towards self-managing online monitoring/analysis for large-scale systems.

In this study, we have applied ASF on the failure logs from IBM Blue Gene/L. Our experimental results show that ASF is more accurate than STF in filtering fatal events due to its consideration of semantic correlation between events. Also, due to its low overhead, we can use ASF to filter non-fatal events as well. We find that ASF is quite effective for all the severity levels, with the resulting compression ratio always below 3%, and often below 1%. After merging filtering results for all severity levels, we find that events naturally form "clusters", with each clustering having non-fatal events first, and then one or more fatal events. These clusters can help visualize how events evolve with increasing severity. Indeed, this can serve as a good basis for further analysis and investigations.

Overview: The rest of this paper is organized as follows. Section II summarizes the event data logs we have collected from Blue Gene/L. Both the STF and ASF methods are presented in Section III, and the filtering results are discussed in Section IV. Following these discussion, Section V includes the related work in filtering event logs. Finally, section VI concludes with a summary of the results and identifies directions for future work.

II. An Overview of IBM Blue Gene/L RAS Event Logs

IBM Blue Gene/L has 128K PowerPC 440 700MHz processors, which are organized into 64 racks. Each rack consists of 2 midplanes, and a midplane (with 1024 processors) is the granularity of job allocation. A midplane contains 16 node cards (which houses the computing chips), 4 I/O cards (which houses the I/O chips), and 24 midplane switches (through which different midplanes connect). RAS events are logged through the Central Monitoring and Control System (CMCS), and finally stored in a DB2 database. The logging granularity is less than 1 millisecond. More detailed descriptions of the Blue Gene/L hardware and the logging mechanism can be found in our earlier paper [12].

We have been collecting RAS event logs from Blue Gene/L since August 2, 2005. Up to the date of November 18, 2005, we have totally 1,318,137 entries. These entries are records of all the RAS related events that occur within various components of the machine. Information about scheduled maintenances, reboots, and repairs is not included. Each record of the logs has a number of attributes. The relevant attributes are described as follows.

- *RECID* is the sequence number of an error entry, which is incremented upon each new entry being appended to the logs.
- *EVENT_TYPE* specifies the mechanism through which the event is recorded, with most of them being through RAS [5].
- *SEVERITY* can be one of the following levels - INFO, WARNING, SEVERE, ERROR, FATAL, or FAILURE - which also denotes the increasing order of severity.
- *FACILITY* denotes the component where the event is flagged, which is one of the following: LINKCARD, APP, KERNEL, HARDWARE, DISCOVERY, CMCS, BGLMASTER, SERV_NET or MONITOR.
- *EVENT_TIME* is the time stamp associated with that event.
- *JOB_ID* denotes the job that detects this event. This field is only valid for those events reported by computing/IO chips.
- *LOCATION* of an event (i.e., which chip/service-card/ node-card/link-card experiences the error), can be specified in two ways. It can either be specified as (i) a combination of job ID, processor, node, and block, or (ii) through a separate location field. We mainly use the latter approach (location attribute) to determine where an error takes place.
- *ENTRY_DATA* gives a description of the event.

III. Filtering Methods

In this section, we present two event filtering methods. First, we briefly introduce the **S**patio-**T**emporal **F**iltering (STF) method that was proposed in [12]. Then, we propose an **A**daptive **S**emantic **F**iltering (ASF) method, which exploits the semantic correlations between events, and can help automate the filtering process for large systems such as IBM Blue Gene/L.

A. A Spatio-temporal Filtering Method

In our previous work [12], we have developed a **S**patio-**T**emporal **F**iltering (STF) method for parsing Blue Gene/L event logs and filtering out redundant/unimportant event records. STF involves three steps: (1) extracting and categorizing failure events; (2) performing temporal filtering to compress events from the same chip locations; and (3) performing spatial filtering to coalesce records of the same event across different locations.

First, the raw logs have to be preprocessed, such as reformatting the entries and handling missing attributes. After the preprocessing step, the next step is to extract all the events with FATAL severity. As pointed out in Section II, an event can be associated with six levels of severity, and STF is designed to focus on filtering events with severity level FATAL because these events can terminate job executions and thus have the most severe impact on system performance. Once all the FATAL events are extracted, based on the involved hardware components, they are categorized into the following six groups: memory related failures (*mem*), network related failures (*net*), midplane switch related failures (*mps*), application I/O related failures (*aio*), node card related failures (*nc*), and unknown failures. The unknown category includes those FATAL events that do not have self-explanatory entry-data fields. In order to correctly categorize an event, we have to examine its entry data field carefully, and often a domain expert is needed. After the categorization step, a temporal filtering is conducted at every chip location, with failures that are from the same job and are close to each other coalesced into one record, and the filtering results from different locations are merged in the temporal order using the sort-merge method. The temporal filtering is a simple threshold-based scheme, and the threshold is chosen with the help of domain knowledge. Finally, due to the parallel nature of these systems and applications, an event may be reported by multiple locations at the same time. Therefore, we adopt a spatial filtering after the temporal filtering phase, which removes redundant records from different locations. Like the temporal filtering step, spatial filtering also employs a threshold, which is again determined by domain experts.

B. An Adaptive Semantic Filtering Method

While STF can effectively compress Blue Gene/L data logs, as shown in [12], it has the following drawbacks. First, it requires extensive domain knowledge, both when categorizing fatal events and when adopting suitable threshold values. Second, it requires manual operations. For example, upon the addition of a new event entry, a human operator is needed to categorize it into different types. As another example, after each step, manual operations are needed to process the partial results to enable operations in the next step. Third, STF only employs simple thresholding-techniques, which cannot handle many tricky situations, and may lead to incorrect results.

To address these challenges, in this paper, we propose an **Adaptive S**emantic **F**iltering (ASF) method, which exploits the semantic context of the event descriptions for the filtering process. ASF involves the following three steps: (1) building a dictionary containing all the keywords that appear in event descriptions, (2) translating every event description into a binary vector where each element of the vector represents whether the corresponding keyword appears in the description or not, and (3) filtering events using adaptive semantic correlation thresholds.

1) Keyword Dictionary: The keyword dictionary is the base for developing the ASF method. The keywords in the dictionary should capture the semantic context of all the events. Building the dictionary is an iterative process. In each iteration, we examine an event entry, identify its keywords, and append new keywords into the dictionary. In order to identify the keywords of an event description, we have adopted the following removal/replacement rules:

1) Remove punctuation, equal signs, single quotes, double quotes, parentheses (including the content in the parentheses).
2) Remove indefinite articles `a`, `an`, and definite article `the`.
3) Remove words such as `be`, `being`, `been`, `is`, `are`, `was`, `were`, `has`, `have`, `having`, `do` or `done`.
4) Remove prepositions such as `of`, `at`, `in`, `on`, `upon`, `as`, `such`, `after`, `with`, `from`, `to`, etc.
5) Replace an alphabetic-numeric representation of a rack, a midplane, a link card, or a node card by the keyword `LOCATION`.
6) Replace an eight-digit hexadecimal address by the keyword `8DigitHex_Addr`.
7) Replace a three-digit hexadecimal address by the keyword `3DigitHex_Addr`.
8) Replace an eight-digit hexadecimal value by the keyword `8DigitHex`.
9) Replace a two-digit hexadecimal value by the keyword `2DigitHex`.
10) Replace a numeric number by the keyword `NUMBER`.
11) Replace binary digits by the keyword `BinaryBits`.
12) Replace a register, e.g. r00, r23, etc., by the keyword `REGISTER`.
13) Replace a file directory or an image directory by the keyword `DIRECTORY`.
14) Replace uppercase letters by the corresponding lowercase letters.
15) Replace present participles and past participles of verbs by their simple forms, e.g. `Failing`, `Failed` being replaced by `Fail`.
16) Replace a plural noun by its single form, e.g. `errors` being replaced by `error`.
17) Replace week days and months by the keyword `DATE`.

After processing all 1,318,137 entries in the raw logs from August 8, 2005 to November 18, 2005, we have identified 667 keywords. One of the advantages of this method is that upon the arrival of new data, we only need to process the new entries as described above, without affecting the earlier data

in any way.

2) Correlation Computation: Following the construction of the keyword dictionary, we next convert each event description into a binary vector for the purpose of semantic correlation calculation. Suppose there are N keywords. Then the vector will have N elements, with each element corresponding to one keyword. In this way, assigning vectors to event descriptions becomes straightforward: 1 denoting the description includes the associated keyword, and 0 denoting otherwise. In order to make this step more automatic, we can choose a reasonably large value for N so that adding new logs will not require re-doing the translations for earlier logs, even when these new logs may introduce new keywords. This approach is further supported by the observation that the number of raw events may be huge, but the number of keywords stays more or less constant after it reaches a certain level.

After generating a binary vector for event record, we can then compute the correlation between any two events using the ϕ correlation coefficient [16], the computation form of Pearson's Correlation Coefficient for binary variables.

		B		Row Total
		0	1	
A	0	$P_{(00)}$	$P_{(01)}$	$P_{(0+)}$
	1	$P_{(10)}$	$P_{(11)}$	$P_{(1+)}$
Column Total		$P_{(+0)}$	$P_{(+1)}$	N

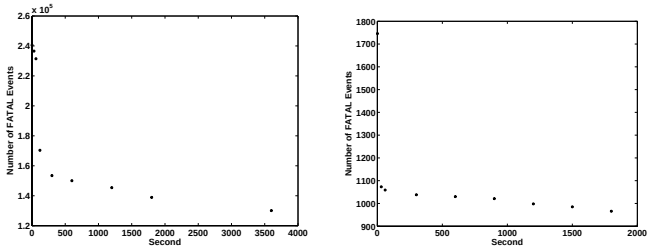
Fig. 1. A two-way table of item A and item B.

For a 2×2 two-way table as shown in Figure 1, the calculation of the ϕ correlation coefficient reduces to

$$\phi = \frac{P_{(00)}P_{(11)} - P_{(01)}P_{(10)}}{\sqrt{P_{(0+)}P_{(1+)}P_{(+0)}P_{(+1)}}}, \quad (1)$$

where $P_{(ij)}$ denotes the number of samples that are classified in the i-th row and j-th column of the table. Furthermore, we let $P_{(i+)}$ denote the total number of samples classified in the ith row, and $P_{(+j)}$ the total number of samples classified in the jth column. Thus, we have $P_{(i+)} = \sum_{j=0}^{1} P_{(ij)}$ and $P_{(+j)} = \sum_{i=0}^{1} P_{(ij)}$. In the two-way table, N is the total number of samples.

3) Adaptive Semantic Filtering: ASF tells whether a record is redundant or not based on its semantic context. An intuitive semantic correlation based approach would regard two records with a high correlation between their descriptions as redundant. A closer look at the logs, however, reveals that in addition to the correlation coefficient, the interval between two records also plays an important role in determining whether these two records are redundant. For example, if two events are very close to each other, even though their correlation may be low, they may still be triggered by the same failure. On the other hand, two records that are far away from each other, though their descriptions may be exactly the same, are more likely to report unique failures. As a result, it is insufficient to adopt a simple thresholding technique solely based on the correlation coefficients between events. Instead, we propose to adopt an adaptive semantic filtering mechanism, which takes into consideration both semantic correlation and temporal information between events to locate unique events.



(a) The number of fatal events after temporal filtering with different t_{th} values

(b) The number of fatal events after spatial filtering with different s_{th} values

Fig. 2. Choosing appropriate values of t_{th} and s_{th} for STF.

We thus believe that, by doing so, we can combine the advantage of STF and the advantage of a simple semantic filtering technique. The principle of the adaptive semantic filtering is that as the time gap between records increases, two records must have a higher correlation coefficient to be considered redundant, and after the time gap reaches a certain level, even two records with a perfect correlation coefficient will be considered unique.

The Adaptive Semantic Filtering (ASF) algorithm first sorts the log entries in time sequence and stores them into the jobID symbolic reference table as shown in Line 2 and Line 3. Then, for each anchor event in the jobID table, if the timestamp of the current anchor event is greater than the largest time threshold, the ASF algorithm deletes this anchor event from the jobID table; otherwise, in line 8, the algorithm computes the semantic correlation between the current event and the anchor event. Line 9 obtains the temporal gap between the current event and the anchor event. If either the temporal or the correlation criteria is satisfied, the current record is filtered out as indicated in Line 12. The above process is iterated for every event record.

IV. Experimental Results

In this study, we have applied the proposed adaptive semantic filtering technique to the raw RAS event logs collected from the Blue Gene/L machine. We report the detailed filtering results in this section.

A. ASF Versus STF

First, we would like to compare the effectiveness of the new semantic filter ASF, and the earlier non-semantic filter STF. Please note that STF is expected to accurately extract unique events from voluminous raw logs because it has involved extensive domain knowledge in the organization of the hardware platform, as well as the logging procedure, and because it has involved a considerable amount of manual effort from these domain experts.

Due to the space limit, instead of presenting the filtering results for all types of events, we only present the results for FATAL events here. For the sake of fairness, we first need to carefully tune the parameters of both algorithms to reach their optimal operating ranges. The main parameters involved in

STF include the threshold in the temporal filtering phase t_{th}, and the threshold in the spatial filtering phase s_{th}. Figure 2(a) plots the number of remaining FATAL events after applying different temporal values t_{th}. We take the viewpoint that a job is likely to encounter only one fatal event of the same type. Hence, a threshold that is too small will result in several fatal events from the same type for a job. On the other hand, since the log has a large portion of entries that do not have a valid JobID field, then a large threshold may filter away failures encountered by different jobs. Both factors considered, we have chosen $t_{th} = 20$ minutes, and this threshold yields 145371 fatal events. Of course, after choosing this threshold, we have validated our choice by examining both the original log and the resulting log manually.

Compared to the temporal threshold, the spatial threshold is easier to set. Similarly, Figure 2(b) gives the number of remaining fatal events after applying different spatial filtering threshold values s_{th}. We have two main observations from this figure. First, applying spatial filtering is very important. Even a zero-second spatial filtering threshold can bring down the number of FATAL events from 145371 to 1746. The second observation is that, the impact of different spatial threshold values is not as pronounced as that of the temporal filtering threshold. This is because the fact that spatial filtering is adopted dominates the filtering effect. As a result, we choose 20-minute as the value for s_{th}.

Using the chosen threshold values, STF can bring down the number of FATAL records from 281462 to 998, which only constitutes 0.3546% of the raw log.

Now, let us switch our attention to the proposed adaptive semantic filter (ASF). As presented in Section III, ASF adopts different correlation coefficient threshold values according to the intervals between subsequent event records. Specifically, we take the viewpoint that two records that are temporally close to each other are likely to be correlated, and therefore, should be coalesced into one event. As a result, we adopt a lower correlation threshold for shorter intervals between subsequent records. On the other hand, two records that are far apart from each other should only be considered correlated when the semantic correlation between them is high, which suggests that we should adopt a higher threshold for events with larger intervals from their preceding events.

In order to develop suitable threshold values, we have partitioned the data sets into two halves, the first half being training data while the second half being test data. On the training data, we have applied different correlation coefficient and interval pairs, and chosen the following values which have produced similar results as those from STF:

$$C_{th} = \begin{cases} 1 & \text{if } T \in [20, \infty) \\ 0.9 & \text{else if } T \in [10, 20) \\ 0.8 & \text{else if } T \in [5, 10) \\ 0.7 & \text{else if } T \in [1, 5) \\ 0.0 & \text{else if } T \in [0.5, 1) \\ -1.0 & \text{else if } T \in [0, 0.5) \end{cases}, \quad (2)$$

where T denotes the interval between the current record and the previous record, and the time unit is a minute. Equation 2 specifies correlation coefficient threshold values for different intervals. For example, if the gap between the current record and the previous record, T, is greater than or equal to 20 minutes, then the current record will be kept in the result log if the semantic correlation between it and its previous record is less than or equal to 1.0. (Of course, since 1.0 is the maximum correlation coefficient, all the events that occur within a window longer than 20 minutes after their preceding events will be kept.) As another example, Equation 2 specifies that if T is less than 30 seconds, then the current event will be filtered out if the semantic correlation between itself and its previous event is less than -1.0. In another word, all the events that occur within 30 seconds after their previous events will be filtered out.

After we extract the parameters in Equation 2 from the training data, we have applied them to the test data to examine whether they are only specific to the training data or they can be used to the test data as well. Fortunately, we find that these values are effective for all the data after careful inspection.

Using the chosen parameters, ASF can condense the fatal failures from 281462 records to 835 records, while STF produces 998 records after filtering. Though these two numbers are rather close, we have observed three typical scenarios in which these two filters yield different results, and that among these three, two cases demonstrate ASF produces better filtering results than STF. These four scenarios are listed below (each record contains the timestamp, job ID, location, and entry data fields):

- *Advantage I: ASF can efficiently filter out semantically-correlated records whose descriptions do not exactly match each other.* Records that are semantically correlated should be filtered out (if they are reasonably close to each other), even though they do not have identical descriptions. ASF can easily achieve this because it considers semantic correlation between events. STF, on the other hand, compares event descriptions word by word, which can be problematic because many highly-correlated records do not have identical descriptions. For example, STF has produced the following records in its result:

  [ST1]2005-11-15-12.07.42.786006 - R54-M0-N4-I:J18-U11 ciod: LOGIN chdir(/xxx/xxx) failed: No such file or directory
  [ST2]2005-11-15-12.07.42.858706 - R64-M1-N8-I:J18-U11 ciod: LOGIN chdir(/xxx/xxxx/xx) failed: No such file or directory
  [ST3]2005-11-15-12.07.42.779642 - R74-M0-NC-I:J18-U11 ciod: LOGIN chdir(/xxx/xxxx/xxx) failed: No such file or directory

  In this example, all three events occur at the same time, but at different locations, and they correspond to the same fatal failure that affects all three locations. However, in the spatial filtering phase, STF only filters out records if their descriptions are the same. As a result, it has kept all three entries. This problem, however, can be avoided by ASF because ASF considers semantic correlation instead of exact word-by-word match. Hence, the result from ASF only contains one entry:

  [AS1]2005-11-15-12.07.42.786006 - R54-M0-N4-I:J18-U11 ciod: LOGIN chdir(/xxx/xxx) failed: No such file or directory

This example emphasizes the importance of semantic correlations in filtering error logs.

- *Advantage II: ASF can prevent non-correlated events from being filtered out.* The previous example shows that ASF can filter out semantically correlated events even when their descriptions are not identical. Similarly, ASF can also prevent non-correlated events from being blindly filtered out just because they are close to each other. This is because STF, in its temporal filtering phase, simply treats all the events that are more than 20 minutes apart as unique while all the events that are less than 20 minutes apart as redundant. Compared to STF, ASF employs a much more sophisticated mechanism, which not only exploits correlation coefficient between two events, and the threshold for the correlation coefficient also adapts to the gap between the events. As a result, if the gap between two events (from the same location) is less than 20 minutes, STF will filter out the second event, but ASF will only do so if their correlation coefficient is above a ceratin level.
  As an example, ASF has produced the following sequence:

  [AS1]2005-09-06-08.45.57.171235 - R62-M0-NC-I:J18-U11 ciod: LOGIN chdir(/home/xxx/xx/run) failed: Permission denied

  [AS2] 2005-09-06-08.49.34.442856 - R62-M0-NC-I:J18-U11 ciod: Error loading "/home/xxxxx/xxx/xxxx/xxxx": program image too big, 1663615088 $>$ 532152320

  In the above sequence, ASF chooses to keep both records because the semantic correlation between them is less than 0.0, and according to parameters in Equation 2, they are unique events. On the other hand, STF condenses the same example scenario to the only entry because the gap between these two events is 4 minutes:

  [ST1]2005-09-06-08.45.57.171235 - R62-M0-NC-I:J18-U11 ciod: LOGIN chdir(/home/xxx/xx/run) failed: Permission denied

  Comparing the results produced by both filters, we can easily tell that both entries need to be kept because each corresponds to a different problem in the system.
- *Disadvantage: ASF may filter out unique events that occur with 30 seconds from each other.* According to Equation 2, ASF filters out events when they occur within 30 seconds of each other. Though in most cases, events that are so close to each other are highly correlated, there are some rare cases where different types events may take place at the same time, e.g. a memory failure and a network failure occurring at the same time. STF can avoid such problems by categorizing failures before filtering. For example, STF has produced the following sequence of records:

  [ST1]2005-08-05-09.11.35.447278 - R33-M1-NC-C:J13-U11 machine check interrupt

  [ST2]2005-08-05-09.11.59.393092 - R54-M1-N8-I:J18-U01 ciod: Error reading message prefix after LOAD_MESSAGE on CioStream socket to xxx.xx.xx.xxx:xxxxx: Link has been severed

	Info	Warning	Severe
before filtering	1,367,531	17,121	15,749
after filtering	11,044	343	148
compression ratio	0.008	0.02	0.009
	Error	Failure	Fatal
before filtering	109,048	1,708	281,441
after filtering	146	53	1,147
compression ratio	0.001	0.03	0.004

TABLE I
THE NUMBER OF EVENTS AT ALL SEVERITY LEVELS BEFORE AND AFTER FILTERING BY ASF.

  Since these two failures, one being memory failure and the other application I/O failure, occur within 24 seconds from each other, ASF compresses them into one entry:

  [ST1]2005-08-05-09.11.35.447278 - R33-M1-NC-C:J13-U11 machine check interrupt

  Fortunately, this problem of ASF does not affect the filtering results much because the likelihood of having two failures within 30 seconds is very low. In fact, we have checked the log carefully, and found that the above example is the only case where two distinct events occur so close to each other. Even in such cases, the problem will be further alleviated by the fact that the production Blue Gene/L usually runs one job at a time, which spans all the processors of that machine. As a result, the adverse effect of compressing failures that occur at the same time is negligible because they hit the same job anyway.

B. The Blue Gene/L RAS Event Analysis

In addition to filtering fatal events, it is also equally important to filter other non-fatal events, since such information can depict a global picture about how warnings evolve into fatal failures, or about how a fatal failure is captured by different levels of logging mechanisms. STF, however, cannot be used to filter non-fatal events due to its complexity, especially when the number of non-fatal events (1,172,766 in our case) is substantially larger than that of fatal events (281,462). Fortunately, this void can be filled by the introduction of ASF, which involves much less overhead and can thus yield an automatic execution.

Table I summarizes the filtering results for events at all severity levels. These numbers show that ASF is quite effective in filtering all types of events, achieving compression ratios below 3% (many compression ratios are below 1%). After filtering events of all severity levels, we can next merge them in the temporal order, and study how lower-severity events evolve to a FAILURE or FATAL event, which can terminate job executions and cause machine reboots. We would note that, the investigation of detailed rules about what non-fatal events will lead to fatal failures, and in what fashion, is well beyond the scope of this paper. In this paper, we argue that such studies are made possible by the introduction of ASF.

After merging events with different severity levels, we observe that they form natural "clusters" consisting of mutiple non-fatal events and one or more fatal events following them. These clusters clearly show that how events evolve in their

facility	severity	timestamp	location	entry data
CMCS	INFO	2005-11-07-08.40.12.867033	-	Starting SystemController UNKNOWN_LOCATION
HARDWARE	WARNING	2005-11-07-08.40.48.975133	R63-M0	EndServiceAction is restarting the NodeCards in midplane R63-M0 as part of Service Action 541
DISCOVERY	WARNING	2005-11-07-08.42.07.610916	R63-M0-N6	Node card is not fully functional
DISCOVERY	SEVERE	2005-11-07-08.42.07.769056	R63-M0-N6	Can not get assembly information for node card
DISCOVERY	ERROR	2005-11-07-08.42.07.797900	R63-M0-N6	Node card status: no ALERTs are active. Clock Mode is Low. Clock Select is Midplane. Phy JTAG Reset is asserted. ASIC JTAG Reset is asserted. Temperature Mask is not active. No temperature error. Temperature Limit Error Latch is clear. PGOOD IS NOT ASSERTED. PGOOD ERROR LATCH IS ACTIVE. MPGOOD IS NOT OK. MPGOOD ERROR LATCH IS ACTIVE. The 2.5 volt rail is OK. The 1.5 volt rail is OK.
HARDWARE	SEVERE	2005-11-07-12.28.05.800333	R63-M0-L2	LinkCard power module U58 is not accessible
MONITOR	FAILURE	2005-11-07-14.11.44.893548	R63-M0-L2	No power module U58 found found on link card
HARDWARE	SEVERE	2005-11-07-14.38.38.623219	R63-M0-L2	LinkCard power module U58 is not accessible

TABLE II

AN EXAMPLE EVENT SEQUENCE THAT REVEALS HOW INFO EVENTS EVOLVE INTO FAILURE EVENTS.

severity. An example cluster is shown in Table II. This sequence starts with an INFO event that informs the system controller was starting. Thirty seconds later, all the node cards on midplane R63-M0 were restarted, as suggested in the following WARNING event, and another two minutes later, another WARNING message points out that one of the node cards on that midplane, R63-M0-N6, was not fully functional. At almost the same time, a SEVERE event and an ERROR event were recorded, which give more detailed information about the node card malfunction. The SEVERE event reports that the assembly information could not be obtained for the same node card, and the ERROR event reports several major status parameters of the node card, such as "PGOOD is not asserted", "MPGOOD is not OK", etc. About 4 hours later, a SEVERE event reports that one of the link cards' power module U58 was not accessible from the same midplane, and about 2 hours later, the power module U58 was reported totally un-found by a FAILURE event. After the FAILURE event, the midplane needs to be repaired by a system administrator before it can be used again.

The ability to locate such sequences is important for studying failure behavior and predicting failures. This was impossible without a good filtering tool. In our example above, there are only 8 records, but they correspond to a much longer sequence in the raw logs, with 572 records. We would note that, it is very difficult, if not at all impossible, to keep track of event occurrences from a 572-entry sequence.

V. RELATED WORK

Collection and filtering of failure logs has been examined in the context of small-scale systems. For example, Lin and Siewiorek [13] found that error logs usually consist of more than one failure process, making it imperative to collect these logs over extended periods of time. In [3], Buckley et al. made recommendations about how to monitor events and create logs by using one of the largest data sets at that time, comprising 2.35 million events from a VAX/VMS system with 193 processors. They pointed out that data sets with poor quality are not very helpful, and can lead to wrong conclusions. Their findings reiterated several important issues in this area, namely, lack of information in the logs (e.g. power outages), errors in the monitoring system (e.g. in the timestamps), and the difficulty of parsing and collecting useful patterns. It has also been recognized [4], [6], [9], [22] that it is critical to coalesce related events since faults propagate in the time and error detection domain. The tupling concept developed by Tsao [22] groups closely related events, and is a method of organizing the information in the log into a hierarchical structure to possibly compress failure logs [4].

Filtering raw logs becomes more important for larger parallel and distributed systems. In our previous study [17], we studied the failure behavior for a large-scale heterogeneous AIX cluster involving 400 nodes over a 1.5 year period. In that study, we used a simple thresholding technique to filter out redundant entries, and the threshold was 5 minutes. In another previous study [12], we developed a spatio-temporal tool (STF) to filter logs collected from a Blue Gene/L prototype consisting of 8192 processors. STF was the first filtering tool that could deal with large failure data sets, and is used as the baseline technique in this exercise.

VI. CONCLUSIONS AND FUTURE WORK

Parallel system event/failure logging in production environments has widespread applicability. It can be used to obtain valuable information from the field on hardware and software failures, which can help designers make hardware and software revisions. It can be used by system administrators for diagnosing problems in the machine, scheduling maintenance and down-times. Finally, it can be used to enhance fault resilience and tolerance abilities of the runtime system for tuning checkpointing frequencies and locations, parallel job scheduling, etc. With fine-grain event logging, the volume of data that is accumulated can become unwieldy over extended periods of time (months/years), and across thousands of nodes. Further, the idiosyncracies of logging mechanisms can lead to multiple records of the same events, and these need to be cleaned up in order to be accurate for subsequent analysis.

In this paper, we have presented an Adaptive Semantic Filtering (ASF) method, which exploits the semantic correlation as well as the temporal information between events to determine whether they are redundant. The ASF method involves three steps: first building a keyword dictionary, then computing the correlation between events, and finally choosing appropriate correlation thresholds based on the temporal gap between events. Compared to existing filtering tools, the proposed filter (1) produces more accurate results, (2) incurs

less overhead, and (3) avoids frequent human intervention. We have validated the design of the filter using the failure logs collected from Blue Gene/L, which consists of 128K processors, and is the fastest supercomputer on the Top 500 Supercomputers List, over a period of 98 days.

Fault-tolerance for large-scale systems requires long-term efforts from the entire community, and this study only serves as a starting point towards this goal. There are several interesting possibilities for future work, and we are particularly interested in investigating online statistical analysis of this data for predictability. Also, we are planning to use this information for enhancing the runtime fault-tolerance mechanisms such as checkpointing and job scheduling.

REFERENCES

[1] The UC Berkeley/Stanford Recovery-Oriented Computing (ROC) Project. `http://roc.cs.berkeley.edu/`.

[2] TOP500 List 11/2004. http://www.top500.org/lists/2005/11/basic.

[3] M. F. Buckley and D. P. Siewiorek. Vax/vms event monitoring and analysis. In *FTCS-25, Computing Digest of Papers*, pages 414–423, June 1995.

[4] M. F. Buckley and D. P. Siewiorek. Comparative analysis of event tupling schemes. In *FTCS-26, Computing Digest of Papers*, pages 294–303, June 1996.

[5] M. L. Fair, C. R. Conklin, S. B. Swaney, P. J. Meaney, W. J. Clarke, L. C. Alves, I. N. Modi, F. Freier, W. Fischer, and N. E. Weber. Reliability, Availability, and Serviceability (RAS) of the IBM eServer z990. *IBM Journal of Research and Development*, 48(3/4), 2004.

[6] J. Hansen. *Trend Analysis and Modeling of Uni/Multi-Processor Event Logs*. PhD thesis, Dept. of Computer Science, Carnegie-Mellon University, 1988.

[7] T. Heath, R. P. Martin, and T. D. Nguyen. Improving cluster availability using workstation validation. In *Proceedings of the ACM SIGMETRICS 2002 Conference on Measurement and Modeling of Computer Systems*, pages 217–227, 2002.

[8] IBM. Autonomic computing initiative, 2002. http://www.research.ibm.com/autonomic/index_nf.html.

[9] R. Iyer, L. T. Young, and V. Sridhar. Recognition of error symptons in large systems. In *Proceedings of the Fall Joint Computer Conference*, 1986.

[10] M. Kalyanakrishnam and Z. Kalbarczyk. Failure data analysis of a lan of windows nt based computers. In *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, 1999.

[11] Y. Liang, Y. Zhang, M. Jette, A. Sivasubramaniam, and R. Sahoo. Bluegene/l failure analysis and prediction models. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2006. To Appear.

[12] Y. Liang, Y. Zhang, A. Sivasubramaniam, R. Sahoo, J. Moreira, and M. Gupta. Filtering failure logs for a bluegene/l prototype. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2005.

[13] T. Y. Lin and D. P. Siewiorek. Error log analysis: Statistical modelling and heuristic trend analysis. *IEEE Trans. on Reliability*, 39(4):419–432, October 1990.

[14] S.S. Mukherjee, C. Weaver, J. Emer, S.K. Reinhardt, and T. Austin. A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*, pages 29–40, 2003.

[15] A. Parashar, S. Gurumurthi, and A. Sivasubramaniam. A Complexity-Effective Approach to ALU Bandwidth Enhancement for Instruction-Level Temporal Redundancy. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2004.

[16] H. T. Reynolds. *The Analysis of Cross-classifications*. The Free Press, New York, 1977.

[17] R. Sahoo, A. Sivasubramaniam, M. Squillante, and Y. Zhang. Failure Data Analysis of a Large-Scale Heterogeneous Server Environment. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, pages 389–398, 2004.

[18] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam. Critical Event Prediction for Proactive Management in Large-scale Computer Clusters . In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2003.

[19] D. Tang and R. K. Iyer. Impact of correlated failures on dependability in a VAXcluster system. In *Proceedings of the IFIP Working Conference on Dependable Computing for Critical Applications*, 1991.

[20] D. Tang and R. K. Iyer. Analysis and modeling of correlated failures in multicomputer systems. *IEEE Transactions on Computers*, 41(5):567–577, 1992.

[21] D. Tang, R. K. Iyer, and S. S. Subramani. Failure analysis and modelling of a VAXcluster system. In *Proceedings International Symposium on Fault-tolerant Computing*, pages 244–251, 1990.

[22] M. M. Tsao. *Trend Analysis and Fault Prediction*. PhD thesis, Dept. of Computer Science, Carnegie-Mellon University, 1983.

[23] K. Vaidyanathan, R. E. Harper, S. W. Hunter, and K. S. Trivedi. Analysis and Implementation of Software Rejuvenation in Cluster Systems. In *Proceedings of the ACM SIGMETRICS 2001 Conference on Measurement and Modeling of Computer Systems*, pages 62–71, June 2001.

[24] J. Xu, Z. Kallbarczyk, and R. K. Iyer. Networked Windows NT System Field Failure Data Analysis. *Technical Report CRHC 9808 University of Illinois at Urbana- Champaign*, 1999.

[25] Y. Zhang, M. Squillante, A. Sivasubramaniam, and R. Sahoo. Performance Implications of Failures in Large-Scale Cluster scheduling. In *Proceedings of the 10th Workshop on Job Scheduling Strategies for Parallel Processing*, June 2004.