Critical event prediction for proactive management in large-scale computer clusters

Conference Paper · January 2003

DOI: 10.1145/956750.956799 · Source: DBLP

8 authors, including:

Ramendra Sahoo
Teradata
49 PUBLICATIONS **2,047** CITATIONS
SEE PROFILE

Adam J. Oliner
University of California, Berkeley
23 PUBLICATIONS **1,525** CITATIONS
SEE PROFILE

Irina Rish
Université de Montréal
189 PUBLICATIONS **4,892** CITATIONS
SEE PROFILE

Manish Gupta
VideoKen Inc.
132 PUBLICATIONS **4,805** CITATIONS
SEE PROFILE

Critical Event Prediction for Proactive Management in Large-scale Computer Clusters

R. K. Sahoo, A. J. Oliner*, I. Rish, M. Gupta, J.E. Moreira, S. Ma
IBM T.J. Watson Research Center
1101 Kitchawan Road, Yorktown Heights, NY 10598-0218
{rsahoo,ajoliner,rish,mgupta,jmoreira,sma}@us.ibm.com

R. Vilalta
Department of Computer Science
University of Houston, 4800 Calhoun Rd, Houston, TX 77204
vilalta@cs.uh.edu

A. Sivasubramaniam
Department of Computer Science and Engineering
Penn. State University, College Park, PA
anand@cse.psu.edu

ABSTRACT

As the complexity of distributed computing systems increases dramatically, systems management tasks require significantly higher levels of automation; examples include diagnosis and prediction based on real-time streams of computer events, setting alarms and performing continuous monitoring. The core of *autonomic computing*, a recently proposed initiative towards next-generation IT-systems capable of 'self-healing', is the ability to analyze data in real-time and predict potential problems; the goal is to avoid disastrous scenarios through prompt execution of remedial actions.

This paper describes an attempt to build a proactive prediction and control system for large clusters. We collected event logs containing various system reliability, availability and serviceability (RAS) events, and system activity reports (SARs) from a 350-node cluster system for a period of one year. The 'raw' system health measurements contain a great deal of redundant event data, which is either repetitive in nature or misaligned with respect to time. We applied a filtering technique and modeled the data into a set of primary and derived variables. These variables are used for establishing event correlations either through prediction algorithms or root cause solutions using probabilistic networks. We also evaluated the role of time-series algorithms, rule-based classification rules, and Bayesian network models.

Based on historical data, our results suggest that it is feasible to predict system performance parameters (SARs) with a high degree of accuracy using time-series models. Rule-based classification techniques can be used to extract machine-event signatures to predict critical events with up to 70% accuracy.

*Affiliated to MIT, Cambridge, MA. Work was carried out at IBM TJWRC. during Summer 2002.

1. INTRODUCTION

Event logs have been used on many computer systems for recording errors occurring in hardware and software components of the system. These logs are typically used by system administrators to monitor the health of the machine, respond to system outages, and plan activities like scheduled maintenance. These system management activities tend to be *reactive* in nature.

In accordance with Moore's Law, the computational power available for a fixed hardware budget has grown at an exponential pace. Clusters of workstations or personal computers are gaining popularity as a cost-effective computing platform [1]. However, clusters remain difficult to manage, and human system administration accounts for a large fraction of the total cost of ownership of these systems. With the constant need for greater reliability, availability, and serviceability (RAS) of systems, there are clear benefits to more automated and *proactive* system management [13].

Several techniques have been proposed in the literature for proactive system management, including prediction of failures, and software rejuvenation. However, it is widely accepted that further research is needed to enhance the effectiveness of these techniques before they are widely deployed. Successful prediction of errors in a computer system offers the promise of enabling significantly improved system management. For example, in a cluster system, prediction of a specific node failure can be used to steer jobs away from the failing node. This could be done prior to job launch (by avoiding such a node for scheduling jobs) or by migrating a running job to healthier nodes before failure strikes. Such predictions can reduce schedule system maintenance at appropriate times to avoid unplanned outages. Even if the prediction of an error happens too late to allow proactive action, the same analysis can serve as a foundation for more effective error isolation (e.g., identifying the node that led to an error, given an avalanche of errors subsequently observed in a cluster).

In this paper, we describe our efforts at building a proactive prediction and control system for large scale clusters. We collected event logs containing information about RAS events and system activity reports (SARs) from a 350 node cluster for a period of one year. We found that filtering of logs to eliminate redundant information was an important first step to allow meaningful analysis of log data, which also helps reduce the space requirements. We prepared a well defined set of primary and derived variables to be used for prediction algorithms. We applied a number of prediction algorithms that have been proposed in the literature, such as time-series algorithms, rule-based classification techniques and Baysian network models, to assess the effectiveness of these techniques for predicting failures in a cluster. Based on the analysis, it was established that different classes of algorithms are effective at predicting different kinds of system events. Our experimental results show that it is indeed feasible to predict the system performance related parameters with a high degree of accuracy using time-series models. Similarly, rule-based classification algorithms can predict the critical events with 70 % accuracy. The prediction accuracy improves further by considering the "warning-windows" within the calculation domain. Whereas, Bayesian network based algorithms can be successfully used to build the dependency graphs to isolate the root cause of problems.

The rest of the paper is organized as follows. Section 2 describes related work on event log analysis for both single node and cluster based systems features. Section 3 presents a brief description of the characteristics of the event logs, including the preprocessing and modeling of the data. Section 4 describes various time-series, rule-based classification and Bayesian network based prediction analysis and results. Finally, we conclude the paper with a summary of the results and our future work plans in Section 5.

2. RELATED WORK

There have been many research efforts on analyzing event logs and other system related health signals [5, 6, 15, 16, 24]. However, relatively few of these efforts have dealt with large clusters. Tsao [23] demonstrated the feasibility of tuple-based classification to reduce the data observed on a DEC system. Lee et al. [15] and Lin et al. [16] worked independently on analyzing the error trends for Tandem system and DCE environments, establishing Weibull, lognormal [19] and other specific distributions through observed data fitting and functions. Recently, Buckley [5, 6] carried out a study on a fairly large VAX/VMS cluster with a total of 193 systems, collecting 2.35 million events covering about 335-machine years of time. His work coalesced related events into a set of critical event logs, extending Tsao's tuple based studies. His scheme covered extensive analyses of the event logs with a vast amount of data, for which fault diagnosis and recovery can be carried out.

For event prediction, a wide variety of algorithms including standard time-series, wavelet analysis and POMDP techniques are available in the literature [14, 18]. We define the event prediction problem for large cluster computer systems similar to the telecommunication problems reported in literature. Time-series based prediction tools have been mostly used to predict telecommunication related problems [7, 26, 27, 28, 29]. However, these prediction tools are not sufficient to address the requirements for computer systems event prediction, because of the mixed nature of the system health related data. Hence, either *dispersion frame* based techniques or heuristic based approaches have been used in the literature for prediction purposes [5]. Traditional classification methods like C4.5 [21] recursively split the instance space until each region is class uniform (i.e., they follow a discriminant-description strategy). Use of either time-series based techniques or heuristic approach for large-scale computer system would result in developing complex event based classification rules. Moreover the presence of uneven inter-arrival time for the events would require either variable inter-arrival time or "time-normalization" based techniques to predict the rare events through data mining [24]. Root cause isolation of problems through dynamic belief networks has been successfully used for Microsoft Windows operating systems [3, 12].

Table 1: Error Log Sample

Error Data:	`sp1n1en0.watson.ibm.com: 33164DD2`
	`0625102702 T H Worm Switch`
	`sender link sync error`
Usage Data:	`00:05:30 0 12 32 4 52`

The absence of any end-to-end system-health for large scale cluster systems and development of a proper proactive prediction and diagnosis motivated us to carry out our study reported in this paper.

3. EVENT LOGS AND OTHER DATA SETS

In order to establish a methodology to collect, filter and analyze the events recorded on clustered systems, all the RAS related events were collected from a 350 node cluster providing both scientific and commercial workloads. The collected data (both system activity reports and event logs) were retrieved through cron jobs.These information were filtered and processed to prepare a well defined set of primary and derived variables to be used for various time-series and other belief-network based algorithms including machine-learning based algorithms for proactive system management, prediction and probing.

3.1 Data Collection

The information about the system can be generalized into three basic categories: event logs, SAR (usage) data, and node topology. The node topology provides the static information about the system connectivity alongwith the respective application domains. The SAR and event log data provide the temporal status of system health and environment. Since the SAR data (collected data at regular intervals) and the event log data can be misaligned, the alignment needs to be carried out, together with filtering and preprocessing the data.

3.2 Event Log Characteristics

The event log from our system consists of a total of six fields of information (Table 1). In Table 1, the event log data represents the node number, event identifier, timestamp, event type, event class and short-descriptions for fields one to six respectively. The event log informations are recorded

Table 2: Event-type vs. Event-class

	Weightage	PERM	PERF	PEND	INFO	TEMP	UNKN
Weightage		5	2	4	0	1	3
Class H	4	155	5	6	9	181	34
Class O	1	11	0	1	35	14	0
Class S	2	213	2	13	28	120	47
Class U	3	9	1	2	0	6	8

through system calls and kernel interrupt mechanisms. Based on the data collected from our cluster, the following classification or characteristics of the events were recorded. Most of the errors can be categorized into either event types or event classes. A combination of event-class and event type can represent the way it is either affecting the system health or the overall system performance. The event types can be categorized as: (1) PEND: The loss of availability of a device or component is imminent. (2) PERF: The performance of the device/component has degraded to below an acceptable level. (3) PERM: Permanent Error (Unrecoverable/Most Severe Error). (4) TEMP: Condition recovered from after a number of unsuccessful attempts. (5) UNKN: Unknown error (Cannot determine the severity). (6) INFO: Entry is an informational/warning.

Similarly, events can also be categorized into the following classes: (1) Class H: Hardware related events, (2) Class S: Software related events, (3) Class O: Events for information only and (4) Class U: Undetermined events.

In other words the event-class and event-types are two dimensions of the space of events recorded through the RAS event log mechanism. The cluster event logs can be represented as an event-matrix (Table 2) with rows having the same event class and the columns representing the event types. More details about the event log characteristics are covered elsewhere [22].

3.3 System Performance Logs

Unlike the event log information, SAR information can be collected at regular intervals. We collected SAR informations for all nodes at a regular period of five minutes. Similar to event logs, this information has six fields, representing time, processor number, user time, idle time, CPU time and I/O time respectively (Table 1). The alignment of the event log data and SAR data was carried out by locating the nearest SAR data collected for that particular event, on that particular node.

3.4 Event Parsing and Filtering Mechanism

Error logs recorded and gathered by the system suffer from several problems:

- A single event may be repeated, due to the associated state being checked multiple times.
- An event may be suppressed because of the co-occurrence of another event. Due to poor resolution on the timestamps, which are accurate only to a minute, two unrelated problems could occur at nearly the same time, and appear as a single event, recorded twice.
- Sometimes the events get lost or deleted before they are logged into the log file because of problems in storing or communication.
- Unwanted information as a result of scheduled operations, time bound reporting status are also recorded in event logs.

In order to handle the problem of duplicate entries, we implemented a series of filtering algorithms. The initial elimination of duplicate adjacent lines from the error logs resulted in a nearly 90% reduction in the number of lines of data. A second filter was designed to address the problem of repeated attempts (e.g. system retry). This filter accepted a threshold time parameter. After an event of a given error identifier occurred at a particular node, the filter would ignore events of the same type occurring on the same node for the duration of the threshold. We selected the threshold parameter such that the duplicate events are less than 1% of the total filtered events.

3.4.1 *Parsing and Correlating System Logs*

Another challenge was how to handle missing event log or usage data, with SAR being collected at regular intervals, whereas the event logs arrivals were totally random. The missing data were mostly as a result of the way the data were collected. Finally, to address the problems of residual errors from a previous time and incomplete usage data, the errors for which we did not have corresponding usage data were removed.

3.5 System Variables

Once the error logs and SAR logs have been filtered appropriately, it is possible to merge these logs in order to create a single, coherent summary of the system's behavior. The system usage data comes in the form of average loads over a period of time. These average loads are associated, using the timestamps, with the errors that occurred during that period. The node topology information indicates not only which nodes share a rack with each other, but also information about the nodes themselves, such as the size of the node. In this manner, it is possible to generate a log of events not only indicating the load on the system, but also covering the generated characteristics of the node at that instant of time. Based on the type of information collected directly or on judgement about hidden features to extract from the system, we defined primary and derived variables for analysis and establishing event correlations.

3.5.1 *Primary Variables*

The primary variables are basically the raw data collected from the system, either through event logs or through sys-

Table 3: Primary and Derived Variables

Variable	Type	Description
Timestamp (*time*)	Primary Variable	Time at which the event occurred
Severity (*Sev*)	Primary Variable	Event severity type
Node ID (*node*)	Primary Variable	Host node ID
Event ID (*eID*)	Primary Variable	Event Identification Number
Event Severity (*sev*)	Primary Variable	Classification based on
		type of problem
Event Class (*class*)	Primary Variable	Classification based on type
		of affected subsystem
CPU Utilization (*%sys*)	Primary Variable	Processor Utilization
User Utilization (*%user*)	Primary Variable	User Utilization
Idle Time (*%idle*)	Primary Variable	System idle time
Frame (*Frame*)	Primary Variable	System Frame
Unconditional Delay (*Delay*)	Derived Variable	Inter-arrival time,
		derived from base time
Node Conditional Delay (*nDelay*)	Derived Variable	Inter-arrival time
		within a particular node
Event Conditional Delay (*eDelay*)	Derived Variable	Inter-arrival time
		for a particular event
Event Node Conditional Delay (*bDelay*)	Derived Variable	Inter-arrival time
		for a particular event within a node

tem activity reporting. It is worth mentioning some of the important primary variables.

- **Severity**: From a unique error ID associated with a particular type of error, it is possible to assign a severity and to classify the error as occurring in a particular subsystem of the cluster. The severity and subsystem, or class, are numbered in such a way that the value of the number indicates the importance of the event. For example, an informational event is given a low weightage, whereas a catastrophic event is given a high weightage.

- **Base Time**: Error logs frequently contained residual errors from a previous time. In fact, some errors were found to be months old. It was necessary to filter out those errors that occurred prior to when we began collecting usage data. We refer this time as the '*base time*' of our data. The timestamps on our errors are made relative to the *base time*.

3.5.2 Derived Variables

In addition to the primary variables provided by the system in these log files, we generated several derived variables to assist in our analysis. An important class of derived variables is related to the inter-arrival time of two events based on certain criteria. The node delay (*nDelay*) indicates, for every error, the amount of time that has passed since an error last occurred at that node. Error delay (*eDelay*) indicates the time since an error with that ID has last occurred on any node. Further, *bDelay* measures the inter-arrival time between two events at the same node and of the same ID. Unconditional delay (*Delay*) simply indicates the time since any events were reported anywhere in the system.

3.5.3 Filtered Datasets

Table 4: Linear Time Series Models

Model	Description
MEAN	Average of Previous Values
LAST	Last Measured Value
BM(p)	Average of Previous N
	Values; N chosen to minimize
	minimize one-step ahead error
AR(p)	Purely Autoregressive;Uses
	Yule-Walker Technique
MA(q)	Moving Average; Uses
	Powell Minimization
ARMA(p,q)	Autoregressive Moving
	Average; Uses Powell

We believe that the combination of filtered error data and the SAR parameters alongwith the system topology information provide an overall system health data-set through proper alignment and preprocessing. There is a clear reduction in the quantity of data as a result of this processing. As an example: a collection of 11 days log (including error logs, usage logs, and node topology information together), there were 6,949,819 lines of data taking up more than 328 MB of space. After filtering and integrating this data, there were 2996 lines of data, totaling only 138 KB.

4. PREDICTION ALGORITHMS

A number of time-series and belief-network algorithms are available in the literature applicable to RAS event analysis. After studying various algorithms, we confined our analysis to three types of algorithms: (1) Time-series algorithms, (2) Rule-based classification algorithms and (3) Bayesian network algorithms. The algorithms are chosen based on the goal of the work, the type of data collected from the systems under analysis, and the applicability of each algorithm.

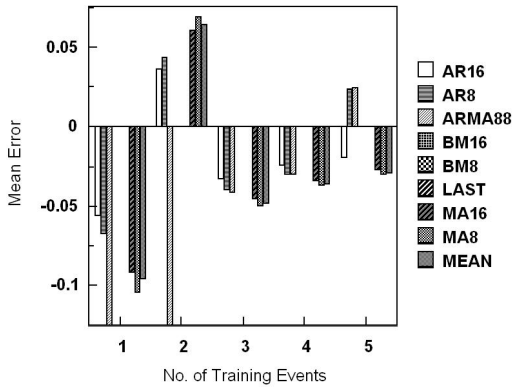

Figure 1: Mean error for event class prediction.

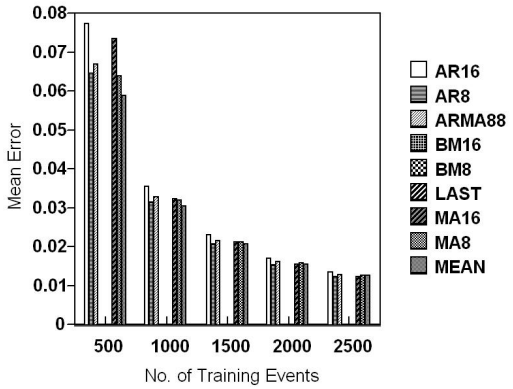

Figure 2: Mean error for system utilization prediction

4.1 Time-series Algorithms

Linear time-series models have been successfully used for forecasting and prediction in various fields. We use time-series models to predict system parameters like percentages of system utilization (*%sys*), idle time (*%idle*), and network I/O (*%IO*). For initial calculations, we assume the events to be distributed at equal time intervals, so that the corresponding system scalar functionalities can be easily used as input parameters for time-series models.

The time-series models [4] we tried out can be obtained from the RPS toolkit implementation [8] . The analysis is performed on a single node basis rather than a for the whole cluster. The following assumptions are made for time-series based modeling and predictions.

- Each primary variable within time-series is assumed to be an independent system health component and uniformly distributed with respect to clock-time. For example, if we analyze the event types as a series of system health related events, we assume that the inter-arrival time does not affect the type of event and its occurrence.
- We assign different weights to both event classes and event types, based on the nature of the problem or how critical the event type or class is for the system. The details of the weight values are given in Table 2.

Figures 1-4 show comparisons of various time-series models for *class*,*%sys*, *sev* and *eID*. As a representative case, we present the comparison of the mean errors associated with different models evaluating the prediction of the 10th event, once the model has been trained through 500, 1000, 1500, 2000 and 2500 data points respectively. All the results are compared with the *LAST* model as a basis of comparison. Based on the results in Figures 1-4, the following observations can be made.

- The *LAST* models, along with *BM(16)* and *BM(8)*, overally do better than other models. This is mostly due to the small changes associated with the performance parameters, compared to the way event logs change with time.

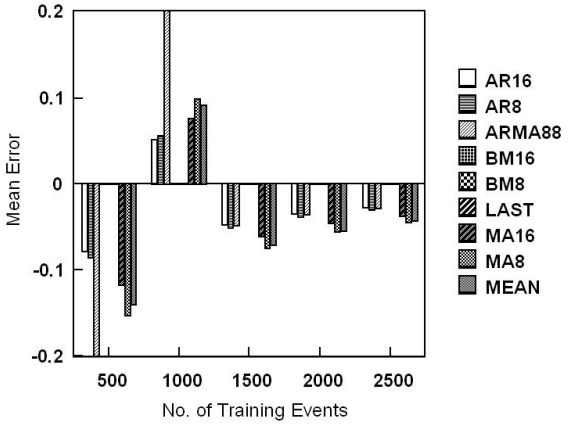

Figure 3: Mean Error for Severity Prediction

- The error associated with *ARMA* model with small amount of training data is quite high. However, for larger training data sets, its performance is better than that of other models.
- For continuous data like *%sys*, the mean error decreases monotonically with the size of the training set.

The mean error bars for some of the time-series models like *BM16*, *BM8* and *LAST* are often negligible compared to the mean errors associated with other models. Hence the error bars do not show up in the figures.

4.2 Rule-based Classification Algorithms

Learning to recognize rare events is a difficult task. The difficulty may stem from several sources: few examples support the target class; events are described by categorical features that display uneven inter-arrival times; and time recordings only approximate the true arrival times, such as occurs in computer-network logs, transaction logs, speech signals, etc.

Our prediction strategy for predicting rare or target events involves the following steps:

1. Finding all event types which frequently precede target events within a fixed time window. We refer to these

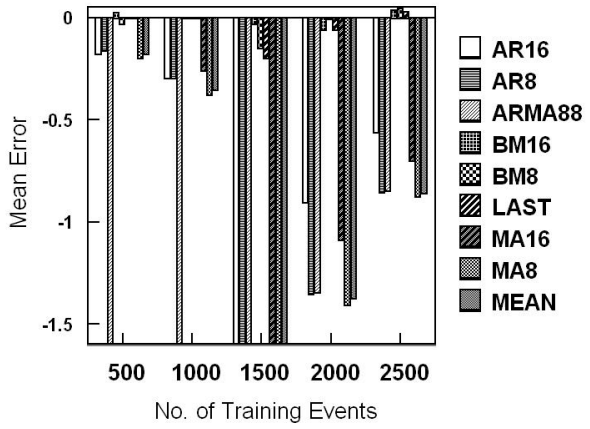

Figure 4: Mean error for event ID prediction

as *eventsets*.

2. Validating eventsets that uniquely characterize target events, and do not occur far from the time arrival of target events.

3. Combining validated eventsets to build a probabilistic rule-based system for prediction.

4.2.1 Searching for frequent eventsets

We are interested in finding sets of event types which occur frequently before a target event within a window of size W (Figure 5) [9, 25]. On every occurrence of a target event, all event types within the window are stored as a new transaction. Once all events have been analyzed, it is straightforward to apply an association-rule algorithm to find all eventsets above a minimum user-defined support.

In our approach, both the ordering of events and the inter-arrival times between events within each time window are not relevant. This is useful when an eventset occurs under different permutations, and when inter-arrival times exhibit high variation (i.e., signals are noisy). These characteristics are present in many domains, including the real production network used for our experiments. For example, we observed that a *DISK-ADAPTER* problem may be generated together with a variety of events under different permutations, and with inter-arrival-time variation in the order of seconds. Further, the filtering process described is Section 3 minimizes the noise and makes sure that the eventsets are free from redundant or false-alarm information.

4.2.2 Accurate eventsets

There is another data filtering process conducted within the algorithm based on the rules establishing the confidence levels. The general idea is to look at the number of times each of the frequent eventsets occurs outside the time windows preceding target events. Such information enables us to compute the confidence of each frequent eventset and to eliminate those below a minimum threshold.

In brief, our validation phase ensures that the probability of an eventset Z appearing before a target event is significantly larger than the probability of Z not appearing before

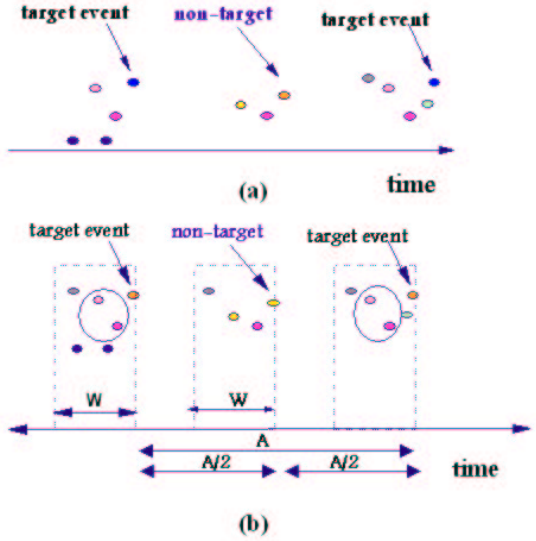

Figure 5: Rule-based classification algorithm

target events. The validation phase discards any negative correlation between Z and the occurrence of target events. In addition, this phase serves as a filtering step to reduce the number of candidate patterns used to build a rule-based model for prediction. We now turn to our goal of finding a model for prediction.

4.2.3 A rule-based model

As the last step, we combine frequent and accurate set of eventsets into a rule-based model. The rationale behind our rule-based system is to find the most accurate and specific rules first [17].

Specifically, let $\mathcal{F}'$ be the set of large and validated eventsets. The first step sorts all eventsets according to their confidence. In the next step, our algorithm selects the next best eventset Z_i and removes all other eventsets Z_j in $\mathcal{F}'$ more general than Z_i. This step eliminates eventsets that refer to the same pattern as Z_i but are overly general. The resulting rule is of the form $Z_i \rightarrow$ targetevent. The search then continues for all eventsets capturing different patterns preceding the occurrence of target events. The final rule-based system $\mathcal{R}$ can be used for prediction by checking for the occurrence of any of the eventsets in $\mathcal{R}$ along the event sequence used for testing. The model predicts finding a target event within a time window of size W after any such eventset is detected.

Before describing our experimental results, we explain additional algorithm parameters. First, we limit the maximum number of negative windows (i.e., windows not preceding target events) as a percentage of the number of target events. One would normally like to consider all negative windows, but this is computationally very expensive. Setting this limit helps reduce the amount of memory and computation. Our default value is 10 times the number of target events. We set to 10% the minimum support threshold (i.e., minimum no. of occurrences for a pattern to be considered frequent). Finally our quality metric for rule ranking is information gain [21], but other metrics could potentially be used (e.g.,

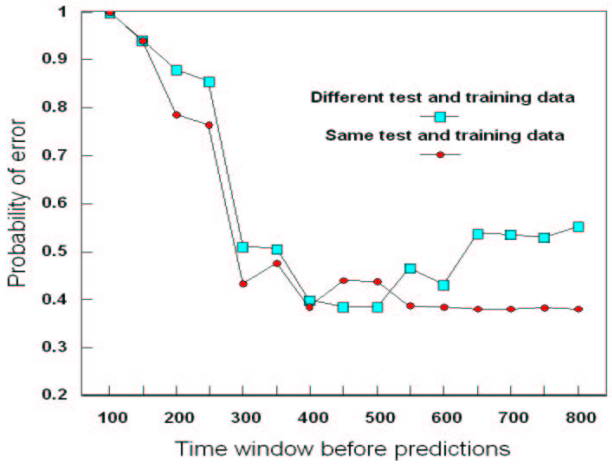

Figure 6: False-negatives through rule-based classification

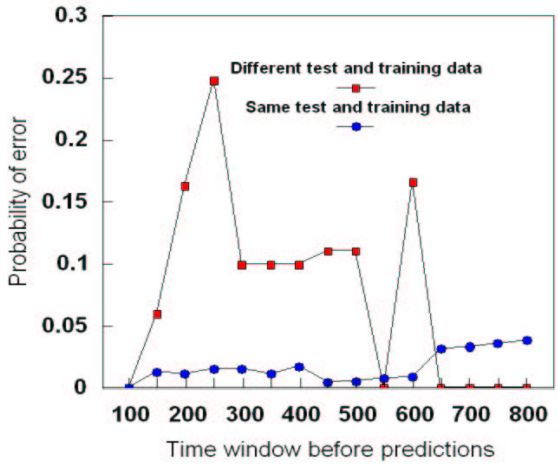

Figure 7: False-positives through rule-based classification

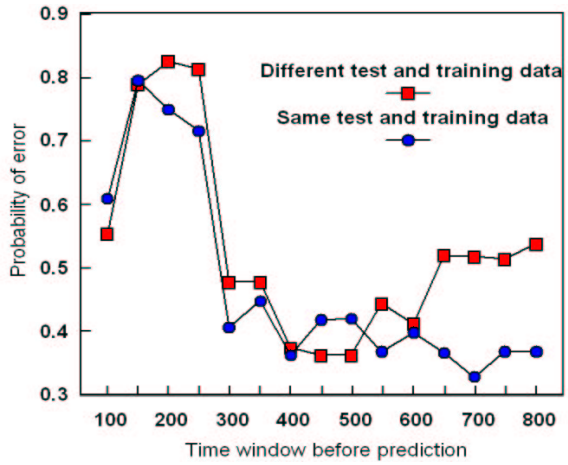

Figure 8: Total errors through rule-based classification

gini, χ^2, Laplace).

We have evaluated the algorithm to compute the accuracy of the prediction of several critical errors with varying time windows, before the occurrence of actual events. As a representative case study, we cover one of the critical hardware error events in detail. Based on the filtered data collected from a 350 node cluster, we chose EventID 193 happening on node 354 as our target event within the cluster. The *eID* 193 represents *Adapter Error* happening on Node 354. We varied the time window size to build the model by training the data. Figures 6, 7 and 8 represent the validation of the model, with either same or different test and training data sets. Figure 6 represents the false-negatives; i.e. the model fails to predict when actual-events might occurr. Here we predict up to 70% accurate the occurrence of the event *eID 193* with an optimum window size of 400 seconds. Figure 8 represents the probability of false-positives; i.e., the model misprediction results for the same set of data as in Figure 6. At a window size of about 400 seconds, the probability of false positives (with different training and test data) is about 0.1, which seems acceptable for actions like avoiding a node, which is likely to fail, for long running jobs. Again from Figure 8 (total error), the optimum window size for the target event is around 400 seconds. The calculations in Figures 6, 7 and 8 do not include the "*warning window*" or time associated with an event for a preventive action to take place. However, when we used the rules extraction process with "*warning window*", the probabilistic analysis resulted in improving the accuracy for various "*warning window*" sizes (Figure 9). We are further investigating the strong periodicity observed for the "*warning window*" in Figure 9.

4.3 Bayesian Network Model

In this section, we describe our work in progress related to learning probabilistic dependency models, such as Bayesian networks, from event data. A Bayesian network model [20] describes domain variables (such as event occurrence, event severity level, etc.) and probabilistic dependencies among specified by conditional probability distributions. Bayesian networks provide a compact representation of multi-variate joint distributions and support efficient algorithms for inference tasks such as prediction and diagnosis.

Formally, a Bayesian network is a directed acyclic graph (DAG) where the nodes correspond to random variables $\mathbf{X} = \{X_1, ..., X_n\}$, and each node X_i is associated with its *conditional probability distribution* $P(X_i|\mathbf{pa_i})$ where pa_i are the *parents* of the node, i.e. nodes directly pointing to X_i. Bayesian network represents a joint probability distribution over variables in $\mathbf{X}$ in a product form:

$$P(x_1, ..., x_n) = \Pi_{i=1}^{n} P(x_i|\mathbf{pa_i}), \quad (1)$$

As a part of our initial analysis we attempted to reconstruct the dependencies between the primary and derived variables for the cluster event data. First, we focused only on variables describing the events coming from a single node (called herein the 'single-node' analysis). Then we considered all variables describing the whole cluster ('cluster analysis'). The results were obtained using B-Course[2], an interactive web-based tool that allows to learn Bayesian network models from data and to perform inferences based on observations.

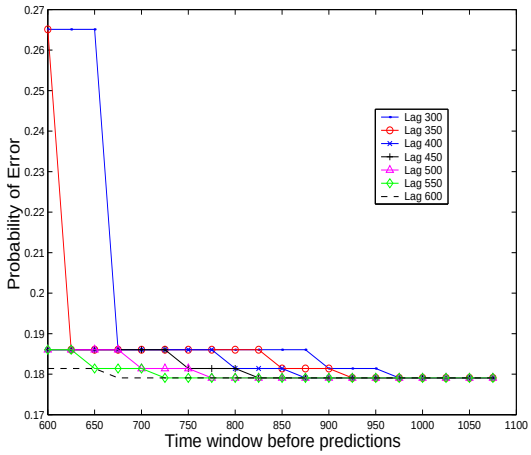

Figure 9: False-positive errors for different "*warning window*" size.

4.3.1 *Single Node Analysis*

From a total of fifteen variables (both primary and derived variables), only eight were included into single node analysis. Some of the variables were excluded because, for single-node analysis, they were constants. These excluded variables were node, size, and frame. Also excluded were *wio* (which was always 0), *%sys* (which was nearly constant), *eDelay*, and *bDelay*. Figure 10 and 11 represent the dependency graphs resulted from evaluating 1907381 candidate models. Actually, the last 1887361 evaluations did not result in finding better models than the current model. The black lines indicate that the model would be less than a billionth as probable if that dependency was removed, i.e., it indicates a strong dependency under the given model. Blue line indicates one thousandth and light blue indicates a weak dependency.

Figures 10 and 11 show snapshots from B-Course's [2] interactive JAVA playground. They depict a Bayesian network model constructed by the tool, and the probabilities associated with each node. B-course attempts to find a model that best fits the data [11, 12], however, it may find several models that fit the data similarly well but differ in the directionality of the arcs, so that we should not necessarily view the directed arcs as causal dependencies. Indeed, in our case different runs of B-course resulted in slightly different DAG structures. However, the 'families' of nodes (node and its parents), and the encoded joint probability distribution, were preserved among different DAGs learned from same set of data.

In Figure 10, all nodes were 'opened' in order to show their prior marginal probability distributions in the absence of observations (no node is assumed to have a particular value). Since most of the variables have too many values (hundreds or thousands), the values were split in groups (shown as intervals in Figures 10 and 11). We can see that most of the events (98%) had high severity, that 99% of all events affected the subsystem of class 3, and that in 96% of cases the activity of user processes was high.

However, more interesting conclusions can be made by *probabilistic inference* that finds *conditional* probability of variables of interest given the *observations*. For example, in Figure 11, the node *Class* was *assigned* value 2 (the node assigned value is highlighted). Given such observation, we can now find conditional probabilities of other variables which makes the dependencies more clear. For example, we can conclude that:

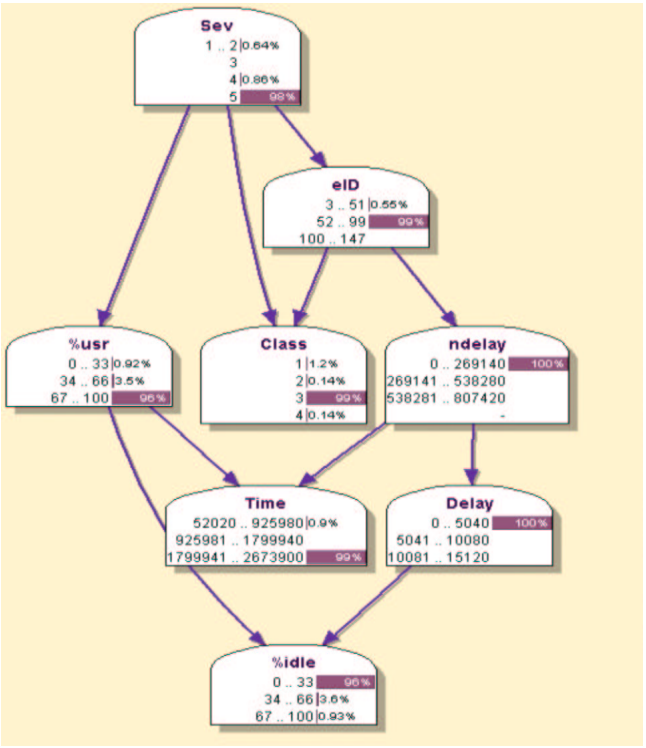

Figure 10: Single-node analysis: a Bayesian network learned by B-Course. Scenario 1: there are no observed nodes, and every node shows its marginal distribution according to the model, prior to any observations.

- most of the events (93%) that had affected subsystem in class 2 were of low severity, and 71% of the such events had *eID* in the range 3 to 51 (note that overall the event IDs were mostly coming from different range, 52 to 99 as shown in Figure 10);
- while the overall user process activity was mostly high (Figure 10), the events affecting the subsystem of class 2 often happened during periods of low activity (e.g. in 44% of cases the activity was quite low).

This is just a simple example of useful inferences that can be drawn from a Bayesian network model. Generally, a diagnosis or prediction task can be expressed as finding $P(X|Y)$ where X is the variable we are trying to predict or diagnose and Y is the set of observations (e.g., features in usual classification framework).

4.3.2 *Cluster Analysis*

In case of the 350 node based cluster, all the fifteen variables were considered for Bayesian network analysis through

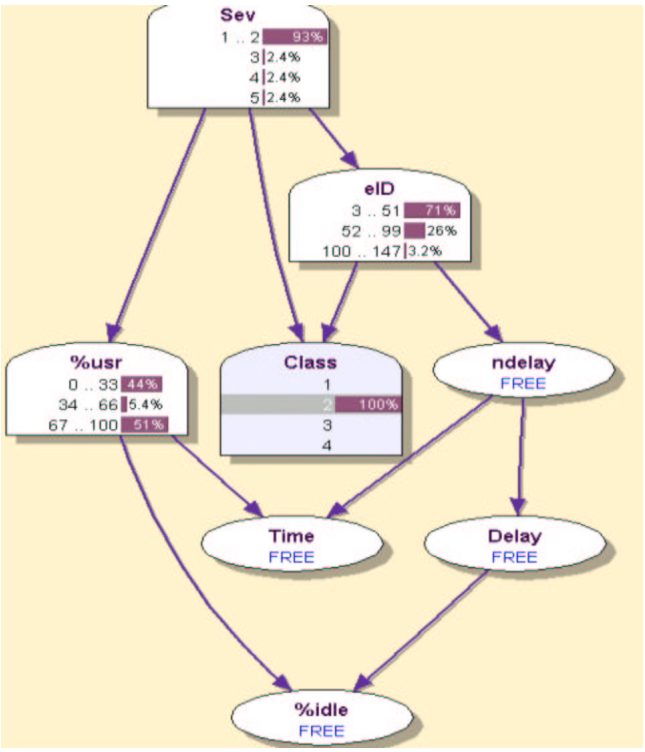

Figure 11: Single-node analysis: a Bayesian network learned by B-Course. Scenario 2: node *Class* takes value 2, and the posterior distributions of nodes *Sev*, *%usr* and *eID* change accordingly.

B-course. Figure 12 represents the network model associating the primary and secondary variables derived from the cluster wide event log and SAR data. Note that the boldface arcs indicate that the model would be less than a billionth as likely if that arc were removed, i.e., it indicates a strong dependency under the given model. Remaining arcs, shown in lighter color, indicate weaker dependencies (removing those reduces the likelihood of the model by smaller factors. A similar 'what-if' analysis based on probabilistic inference can also be carried out for the whole cluster using the Bayesian network extracted for the variables. Based on the explanations above, it is clear that, irrespective of the availability of the detailed knowledge about the cluster system, a dependency graph with probabilistic model can be established. In other words, using no assumptions about the system, these Bayesian tools would be able to determine statistical relationships among the variables of a node or a number of nodes within a cluster system.

We plan to continue the current work in order to come up with Bayesian network diagnosis and prediction algorithms based on event streams. Our future work includes learning *temporal* probabilistic dependencies between the events using Dynamic Bayesian networks [10], and using them together with time-series analysis in order to predict future events based on the observed ones.

5. SUMMARY

A total system health realted information processing, prediction and control analysis effort has been carried out based

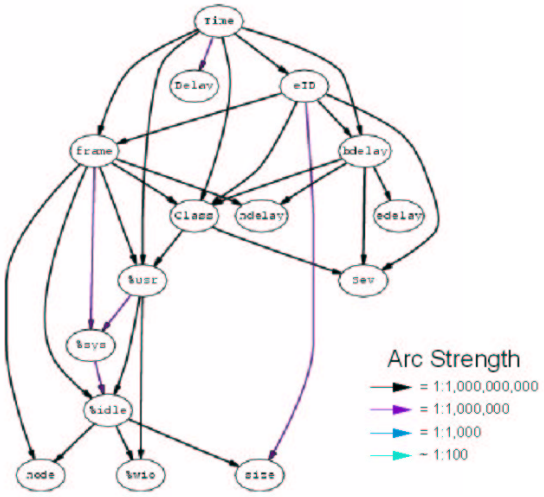

Figure 12: Bayesian network model for primary and secondary variables based on B-course for a 350 node cluster.

on event logs, system activity reporting, and other system related parameters. The data analyses are based on the collection of a variety of system health related information gathered from a 350 node cluster over a period of one year, logging a total of 350 machine year time.

An event parser or filtering mechanism has been developed to take into account a number of both discrete and continuous signals including system and event collection intermittency. The parsing and filtering of data were carried out to minimize false reporting, reporting of artificial events and other scheduled system maintenance operations to name a few; which can be categorized as non-system related information for system RAS. The filered data was found to be less than 1% of the total collected data. Based on the processed data, a set of primary and derived variables were established. These variables can be either discrete or continuous variables. The decision about the variables were based on the requirement from the prediction algorithms for proactive rare event prediction, diagnosis and control.

In terms of prediction algorithms for discrete and continuous variables; time-series analysis was found to be more useful for predicting continuous variables like *usage* and *idle time* than for predicting event characteristics based variables. A rule-based classification algorithm could predict the critical rare events with up to 70% accuracy based on the associative data mining rules within a specified time window. By including the *warning window* parameter into the analysis, rule-based classification results were further improved in terms of prediction accuracy.

Bayesian network models can be effectively used for establishing causal relationship among the defined primary and derived variables, including the failure probability of the nodes with the type of errors.

For proactive management and control, the system performance parameter prediction results from time-series analysis, rare event occurrence from rule-based classification re-

sults and the root causal analysis through Bayesian network based analysis could be combined. Moreover, it would be possible to take actions in terms of job submission, process migration to avoid potential problem domains within large clusters.

Our future work plan includes, developing a hybrid model linking the three prediction and probing components together and verifying the integrated hybrid model for a large cluster RAS system. We are also, evaluating to establish an online system to carry out an automated system management and control including developing a cluster probe manager as our next target.

6. ACKNOWLEDGEMENTS

We acknowledge Peter D. Dinda for using RPS toolkit and B-Course, an online Bayesian network tool for some of the analyses carried out in this paper.

7. REFERENCES

[1] N. R. Adiga, et al. An Overview of the BlueGene/L Supercomputer. *Supercomputing (SC2002)*, 2002.

[2] A web-based Data-analysis tool for Bayesian Modeling. *http://b-course.cs.helsinki.fi.*

[3] J. Berger. Statistical Decision Theory and Bayesian Analysis. *Springer-Verlag, New York*,1985.

[4] P. J. Brockwell, R. Davis. Introduction to Time-Series and Forecasting. *Springer-Verlag*, 2002.

[5] M. F. Buckley and D. P. Siewiorek. VAX/VMS Event Monitoring and Analysis. *FTCS-25*, Computing Digest of Papers, Pasedena, CA, 414–423, June, 1995.

[6] M. F. Buckley and D. P. Siewiorek. A Comparative Analysis of Event Tupling Schemes. *FTCS-26*, Intl. symp. on Fault-Tol. Computing, 294–303, June, 1996.

[7] T. Dietterich, R. Michalski. Discovering Patterns in Sequences of Events. *Artificial Intelligence*, 187–232, 1985.

[8] P. Dinda, A Prediction-based Real-time Scheduling Advisor. *Proceedings of IPDPS*, 2002.

[9] C. Domeniconi, C. S. Perng, R. Vilalta, S. Ma A Classification Approach for prediction of Targetted Events in Temporal Sequences. *SIGMOD/PODS 2002*, ACN SIGMOD/PODS 2002 Conference.

[10] N. Friedman, K. Murphy and S. Russell. Learning the Structure of Dynamic Probabilistic Networks. *Proceedings 14th. Conf. on Uncertainty in Artificial Intelligence (UAI)*, 139–147,1998.

[11] N. Friedman. Learning Belief Networks in the Presence of Missing Values and Hidden Variables. *Proc. 14th Intl. Conf. on Machine Learning.* 125-133, 1997.

[12] D. Heckerman. A Tutorial on Learning with Bayesian Networks. *Tech. Rep. MSR-TR-95-06,*, Microsoft Research, 1996

[13] P. Horn. Autonomic Computing: IBM's Prospective on the State of Information Technology *http://www.research.ibm.com/autonomic/*, IBM Corporation, 2001.

[14] F. Jensen. An Introduction to Bayesian Networks. *UCL Press, London*,1996.

[15] I. Lee, R. K. Iyer, D. Tang. Error/Failure Analysis using Event Logs from Fault Tolerant Systems. *Proc. 21st Intl. Symposium on Fault-Tolerant Computing,*, 10-17, June 1991.

[16] T. Y. Lin, D. P. Siewiorek. Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis. *IEEE Trans. On Reliability*,39, 4, 419-432, Oct. 1990.

[17] R. S. Michalski A Theory and Methodology of Inductive Learning . *Machine Learning*, 1995.

[18] K. P. Murphy. Active Learning of Causal Bayes Net Structure. *http://citeseer.nj.nec.com/451402.html.*

[19] T. Nakagawa, S. Osaki. The Discrete Weibull Distribution. *IEEE Trans. Reliability*, Vol R-24, 300-301, Dec. 1975.

[20] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, San Mateo, California, 1988.

[21] J. R. Quinlan. C4.5: Programs for Machine Learning. *Morgan Kaufmann*, 1994.

[22] R. K. Sahoo, M. Bae, R. Vilalta, J. Moreira, S. Ma and M. Gupta. Providing Persistent and Consistent Resources through Event Log Analysis and Predictions for Large-scale Computing Systems. *SHAMAN, Workshop, ICS'2002*, New York, June, 2002.

[23] M. M. Tsao. Trend Analysis and Fault Prediction. *PhD Dissertation, Carnegie-Mellon University,* , May, 1983.

[24] R. Vilalta, C. Apte, J. Hellerstein, S. Ma, S. Weiss. Predictive Algorithms in the Management of Computer Systems. *IBM Systems Journal, Special Issue on Artificial Intelligence Vol. 41, 3, 2002*

[25] R. Vilalta, S. Ma. Predicting Rare Events in Temporal Domains. *Proceedings IEEE Conf. on Data Mining (ICDM.02) Maebashi, Japan*

[26] G. M. Weiss and H. Hirsh. Learning to Predict Rare Events in Event Sequences. *Proceedings 4th. Intl. Conf. on KDD (KDD-98)*, 4:359–363, 1998.

[27] G. M. Weiss and J. P. Ros. Implementing Design Patterns with Object-oriented Rules. *Journal of Object-Oriented Programming*, 11(7):25–35, 1998.

[28] G. M. Weiss, J. P. Ros, and A. Singhal. Answer: Network Monitoring using Object-oriented Rules. *Proceedings 10th. Conf. IAAI (IAAI-98)*, 10:1087–1093, 1998.

[29] G. M. Weiss. TIMEWEAVER: A Genetic Algorithm for Identifying Predictive Patterns in Sequence of Events. *Proceedings 10th. Conf. IAAI (IAAI-98)*, 10:1087–1093, 1998.