Predicting Computer System Failures Using Support Vector Machines

Errin W. Fulp* Glenn A. Fink† Jereme N. Haack†

*Wake Forest University
Department of Computer Science
Winston-Salem, NC

†Pacific Northwest National Laboratory
Information and Infrastructure Integrity Initiative
Richland, WA

Abstract

Mitigating the impact of computer failure is possible if accurate failure predictions are provided. Resources, applications, and services can be scheduled around predicted failure and limit the impact. Such strategies are especially important for multi-computer systems, such as compute clusters, that experience a higher rate failure due to the large number of components. However providing accurate predictions with sufficient lead time remains a challenging problem.

This paper describes a new spectrum-kernel Support Vector Machine (SVM) approach to predict failure events based on system log files. These files contain messages that represent a change of system state. While a single message in the file may not be sufficient for predicting failure, a sequence or pattern of messages may be. The approach described in this paper will use a sliding window (sub-sequence) of messages to predict the likelihood of failure. The a frequency representation of the message sub-sequences observed are then used as input to the SVM. The SVM then associates the messages to a class of failed or non-failed system. Experimental results using actual system log files from a Linux-based compute cluster indicate the proposed spectrum-kernel SVM approach has promise and can predict hard disk failure with an accuracy of 73% two days in advance.

1 Introduction

Hardware failures, such as hard disk and processor failures, can impede the execution of applications on High-Performance Computing (HPC) systems since the recovery process can require unexpected amounts of time and resources. The impact of failure is more substantial for large-scale HPC clusters that consist of many computing elements. BlueGene demonstrated that the mean-time to failure of these systems is inversely proportional to the system size (number of computing elements) [1], which results in lower reliability.

We can improve reliability by predicting hardware failure and scheduling applications and services around it [10]. This strategy will enable us to fully harness the potential of the next generation HPC. For example Tantawi and Ruschitzka [12] added the concept of an *equicost* checkpointing strategy, which calculates the checkpoint interval by attempting to balance the checkpointing cost against the likelihood of failure. Oliner and Sahoo introduced a risk-based model that incorporates failure indicators from real job logs, communication topologies, scheduling policies, and real failures to predict processor failure. They found that risk-based cooperative checkpointing with prediction accuracies of only 10% can still produce performance improvements [8]. However, the success of this strategy will ultimately rely on accurate predictions of imminent failures with sufficient lead time to efficiently mitigate failure.

There have been several approaches for predicting system failure using system log files [4, 8, 11, 15, 14]. System log files consist of messages created by the different processes executing on the system. The information recorded varies from general messages concerning user logins to more critical warnings about program failures. Prediction methods include standard machine learning techniques such as Bayes networks, Hidden Markov Models (HMM), and Partially Observable Markov Decision Process (POMDP) [14].

The use of time-series analysis is common among these methods since a system message in isolation has been shown to be insufficient for predicting failure [9, 15]. We also believe that certain sequences of log messages may provide sufficient information to predict failure. But the large amount of information available in system log files makes finding the right pattern(s) difficult.

In this paper we introduce a new approach for predicting critical system events based on Support Vector Machines (SVMs). Given labeled training data, an SVM

can determine the maximum hyperplane that separates the two classes of data. The classifier that results from training is represented by a smaller portion of the training data, called the support vector. A classifier can be used to associate a system with a certain group, for example systems that fail, based on information that precedes the event thereby producing a prediction.

Aggregate features are often used for SVM-based classification, for example the the average number of messages during a period of time. However it is important to exploit the sequential nature of system messages. Unlike other applications of SVM classifiers [8, 15, 14], this paper will use a spectrum representation of message sequences, which represents a k-length sequence of messages as one feature. The SVM can then classify systems as either fail or non-fail using the number of occurrences of different message sequences. Experimental results using actual system log files from a Linux-based cluster show this approach can predict hard drive failure with 73% accuracy two days in advance (lead time). This three-orders-of-magnitude extension in the window length will be very useful for allocating processors to longer-running jobs. We believe our approach will greatly enhance the performance and reliability of HPC.

The remainder of this paper is structured as follows. Section 2 describes the information contained in system log files and how it may be used for predictions. Failure predictions using SVM and spectrum-representation is given in section 3. Experimental disk failure prediction results using actual Linux log files is given in section 4, while section 5 provides a summary and reviews some open questions about this promising prediction method.

2 System Log Files

System log files are important for managing computer systems since they provide a history or audit trail of events [5]. In this context, an event is a change in system status, such as a user login or an application failure. Given the log file information it may be possible to determine causes of events such as system errors or security problems that have occurred. Although this type of forensic analysis is valuable, it is also possible to use the information contained in system log files for predicting events.

System log files typically are text files, that consist of messages sent to the logging service by applications. For example `syslog` is configurable general purpose logging application available for different Unix platforms [5]. Applications can send information to the syslog process, which stores the messages in a text file in the order that they arrive. In a typically cluster the syslog server process resides on a separate host and receives messages from each node over a network connection.

Syslog is primarily responsible for managing the log file while the message content is largely created by the application. As seen in figure 1, messages do have a specific format consisting of the six fields. In this example, the host field is the IP of the machine sending the message (since one syslog instance may serve multiple computers). The facility field is the source of the message, for example kernel or user space. The level indicates the severity of the message, which ranges from general to critical. The tag field is a positive integer that represents the facility and level fields where lower tag values represent more critical messages. For example in figure 1, a disk failure has a tag value of 1 while the execution of a command (first message) has a tag value of 189. Since this value is the combination of the facility and level, it does not reflect the actual message content. For example in figure 1, the tag value 38 occurs twice although the message content is different.

The time field is the time the message was recorded by the syslog facility. Finally, the message is the text portion of the entry that describes the event that has occurred. The message is created by the application and is a free-form field. This makes analysis more difficult requiring complex natural-language parsers. However, recent research has been successful in associating certain words that appear in the log file messages with critical future events [11].

2.1 SMART Messages

Although a wide variety of log file messages exist, Self-Monitoring Analysis and Reporting Technology (SMART) messages are of interest since they provide information specific to disk drive health. SMART boards are becoming a standard component of ATA and SCSI hard disks. SMART disk drives internally monitor their health and warn of impending hard disk drive problems. In many cases, the disk itself provides advance warning that something is wrong long before actual disk failure. Most implementations of SMART also allow administrators to perform self-tests and monitor performance and reliability attributes.

The `smartd` daemon regularly monitors the disk's SMART data for signs of problems. When `smartd` is started, it registers the system's disks then checks their status every 30 minutes for attributes indicating failure. If there is a failing health status or an increased number of errors or failed self-tests, the daemon sends this information to the system log file. Pinheiro *et al.* found that while some SMART parameters do correlate with drive failure, these messages alone are insufficient for prediction [9].

Host	Facility	Level	Tag	Time	Message
198.129.8.6	local7	notice	189	1171061732	sysstat
198.129.8.6	kern	info	6	1171061732	kernel md: using maximum available idle IO bandwidth
198.129.8.6	cron	info	78	1171061733	crond 2500 (root) CMD (/usr/lib/sa/sa1 1 1)
198.129.8.6	auth	info	38	1171062445	rsh(pam_unix) 2215 session opened for user by (uid=0)
198.129.8.6	auth	info	38	1171062445	in.rshd 2216 root@hpcs2.cs.edu as root: cmd=/root/temps
198.129.8.6	daemon	info	30	1171062590	smartd 88 Device: /dev/twe0 SMART Prefailure Attribute
198.129.8.6	auth	notice	37	1171062590	sshd(pam_unix) 12430 auth failure; logname=el-fork-o
198.129.8.6	kern	info	6	1171062590	kernel md: using 512k, over a total of 12287936 blocks.
198.129.8.6	cron	info	78	1171062601	crond 2500 (root) CMD (/usr/lib/sa/fork-it 1 1)
198.129.8.6	kern	alert	1	1171062692	kernel raid5: Disk failure on sde1, disabling device

Figure 1: Example `syslog` messages, each consisting of six fields (host, facility, level, tag, time, and message).

3 SVM Event Prediction

As described in the introduction, a Support Vector Machine (SVM) is a supervised machine-learning method that can be used for binary classification: associating a non-labeled sample to one class or another. Given a set of labeled training data (features) the SVM finds the best plane that separates the two classes. If the data is not linearly separable, then it is possible to translate the data into a higher-dimensional space to find a separator, referred to as the *kernel-trick*. Once the plane has been found, the SVM can associate new non-label data to a specific class. Furthermore SVMs allow the inspection of weight functions, making it possible to identify which features are most responsible for classification.

The SVM can be used for predicting system failure by concluding that a series of messages are or are not associated with failure in the future. Let M represent a time-ordered set of log file messages for a single computer. Therefore, the sequence of messages that appear in M form a time-series representation of events that occurred. For this paper let each message in M be represented by its tag value, which provides an indication of message criticality. For the messages given in figure 1, the set M would be $\{189, 6, 78, 38, 38, 30, 37, 6, 78, 1\}$.

Using M it is possible to collect various aggregate features, such as the frequency of tag values. For the messages in figure 1: the tags 189, 30, 37, and 1 occur once; and the tags 78, 38, and 6 occur twice. This forms a vector describing M as (1:1, 6:2, 30:1, 37:1, 38:2, 78:2, 189:1), where each value is the *tag*:*count*. This vector can then be used to classify M as belonging to a fail or non-fail system.

3.1 Spectrum Representation

The frequency of tag values in M can help predict system events, however, additional valuable information is found in the sequence of messages. Identifying certain sequences of messages, represented by tag values, that are a prelude to a certain system events is one specific example. The spectrum kernel representation of the log messages will provide this ability for SVM classifiers.

The spectrum kernel representation of data was developed by Leslie *et al.* for determining protein sequence similarity [6]. This representation used a k-length sequence of amino acids as a feature for proteins. The proteins representation was then the number of times sequences occurred as a k-length window is passed over the sequence of amino acids. The spectrum-kernel representation has also been used for network connection classification [13] and intrusion detection systems [2]. In this paper we apply this approach to sequences of system log messages, specifically the tag values.

The spectrum representation of M will consist of the frequency of the different possible k-length sequences. The number of possible sequences is b^k, where b is the number of different tag values. Given the finite amount of memory available to store information, limiting either the window size k and/or the tag range b may be necessary. For example, assume there are 65536 possible tag values and the window length is 5, then there are 1.2×10^{24} possible tag sequences. Therefore, limiting the tag range allows the consideration of longer sequences and vice versa.

Assume M consists of the 10 tag values given in figure 2. Let the window size be 5, which is also the sequence length, k. Furthermore, associate each tag with one of three *levels*, high (tag $<$ 10), medium (10 $\leq$tag$\leq$ 140), low (tag$>$ 140). In this example b is three which results in 243 possible sequences, substantially lower than if no levels are used. Let the level representation of a tag, or encoding, be denoted as e. Assign the high level tag the encoding value 0, the medium level 1, and low level 2.

To find the sequences in M, pass a k-length window over M and record the sequence occurrences. For example consider again the messages in figure 2. The sequence $\{22212\}$ (which corresponds to three low level messages, one medium message, followed by one low level message) occurs twice. Similarly, the the sequence $\{21222\}$ twice, while the sequences $\{22122\}$, and $\{22122\}$ occur only once. All other possible se-

quences do not occur in this example.

Let each sequence be represented by a unique value, f. For example, the feature value 239 identifies the sequence $\{21222\}$. As described in [13], this can be done using a process that only requires storing the previous feature value f and the new tag number encoding. The $n+1$ feature value can be determined using the n^{th} feature and tag number encoding, the feature value for the $n+1$ feature is

$$f_{n+1} = \text{mod}(b \cdot f_n, b^k) + e_{n+1}$$

Determining the frequency of the different sequences observed in M creates the vector that represents the messages. The vector for the example given in figure 2 is (160:1, 215:2, 233:1, 239:2), where each value is the *sequence-value*:*count*. Again, the vector can be used to classify the system to be a member of the fail or non-fail group.

4 Experimental Results

The SVM approach to predicting system events was evaluated experimentally using actual system log files. The log files were collected over approximately 24 months from a 1024 node Linux-based compute cluster. Each computer was similarly configured and consisted of multiple processors and disks. The system event predicted was hard disk failure, since it is easy to identify in the log file, for example the last message in figure 1 is a disk failure.

The log files contained over 120 disk failure events. For a single computer it is possible to have multiple disk failures within a relatively short period of time, since each computer has multiple disks. In this case only disk failures that were separated by at least one day were considered. This was done to eliminate any messages that may have been due the previous failure. As a result, only 100 of the 120 failures were usable for the experiments.

The log files averaged 3.24 messages per hour per system. Therefore each machine averaged approximately 78 messages per day. The distribution of tag values are given in figure 4. There were 61 unique tag values, ranging from 0 (the most critical message) to 189 (least important message). As seen in the histogram certain tag values appear more frequently, a fact that we use to determine the appropriate tag ranges as described in the previous section.

A set of 1200 messages (approximately 15 days of data) was collected from systems that did and did not experience a disk drive failure. For computers that experienced a failure, the last message of the 1200 is a disk failure message. Starting at the first message in the set, a sub-set of messages, M, was used to make a prediction. The length of M equaled 400, 600, 800, 1000 or

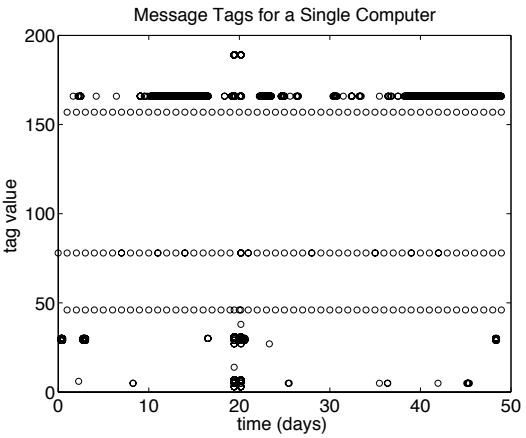


Figure 3: An example of the logged messages for a single computer over a 50 day period. Each circle is a message and the tag value for each message is plotted on the y-axis.

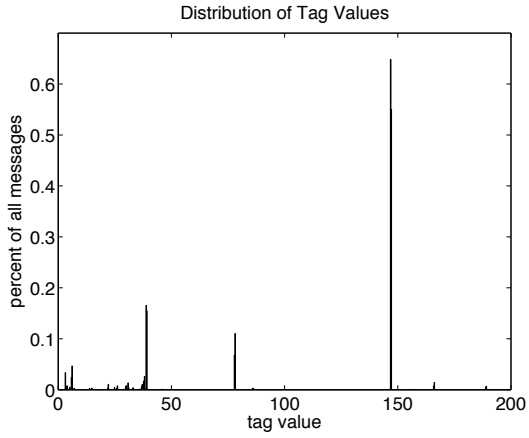


Figure 4: Distribution of tag values for the system log files used in the experiments.

tag	**Encoding** (e)	**Sequence**	f (base 10)
148	2	2	
148	2	22	
158	2	222	
40	1	2221	
158	2	22212	239
188	2	22122	233
188	2	21222	215
88	1	12221	160
158	2	22212	239
188	2	22122	215

Figure 2: Example spectrum representation of 10 tag values.

1100 messages. Therefore when M equals 1000 messages, only 200 messages remain before the failure event. Larger values of M should result in better predictions since more data is available.

Hold-out was used to partition the samples, where half of the disk failure sets are randomly selected for training and the other half is used for testing. An equal number of fail and non-fail sets were used per experiment. Hold-out was repeated 100 times per experiment and the average performance was recorded.

4.1 Performance Measurements

As described in the introduction, the objective of event prediction is to accurately predict disk failure as far as possible in the future. Given a binary classifier and an input, four outcomes are possible [3]. Assume the input is in the positive class. If the classifier predicts positive then the outcome is a true positive. If the classifier predicts negative then the outcome is a false negative. If the input is negative and the outcome is negative, then it is a true negative. If the input is negative and the outcome is positive, then it is a false positive. For this paper let the positive class represent drives that do not fail. Therefore a false positive is a drive predicted not to fail but does, in contrast a false negative is a drive predicted to fail but does not.

The percent accuracy, precision, and recall was recorded for each experiment. Accuracy is the total number of correct predictions (fail and non-fail) divided by the total number of inputs. Precision is the number of true positive divided by the number of true positives and false positives. Recall is the number of true positives divided by the number of positive inputs. A predictor with a high precision has fewer false positive errors (predicted good but actually fails), while a predictor with a high recall has fewer false negative errors (predicted to fail but is good).

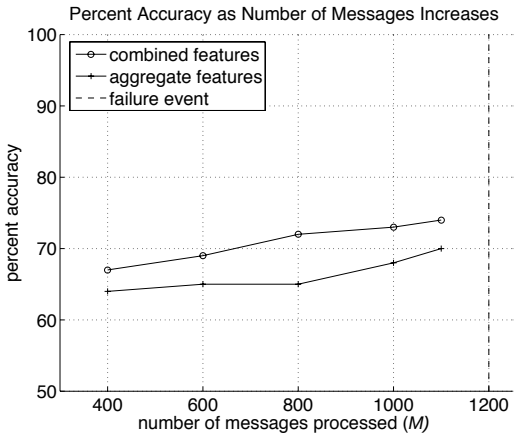
Another measure of classifier performance is the Receiver Operating Characteristics (ROC) curve [3]. As seen in figure 5(b), false positive rate is plotted against the true positive rate. The graph shows the trade-off benefits (true positives) and costs (false positives). For example, the point $(0, 0)$ represents a classifier that never generates a positive classification. This classifier would not have any false positives but would not have any true positives either. The opposite is true for the point $(1, 1)$, which is a classifier that never generate a negative classification. The point $(1, 0)$ represents a perfect classifier since the true positive rate is maximized and the false positive rate is minimized. Therefore, an ROC curve that is close to the top left-hand corner of the graph is considered a good classifier. In contrast, the line between points $(0, 0)$ and $(1, 0)$ represents a random classification outcome for any given input.

Finally, the Area Under the Curve (AUC) for an ROC is often used to measure classifier performance. The area for the perfect classifier is 1 while a random classifier is 0.5 (area below the diagonal), therefore a classifier should have an AUC between 1 and 0.5.

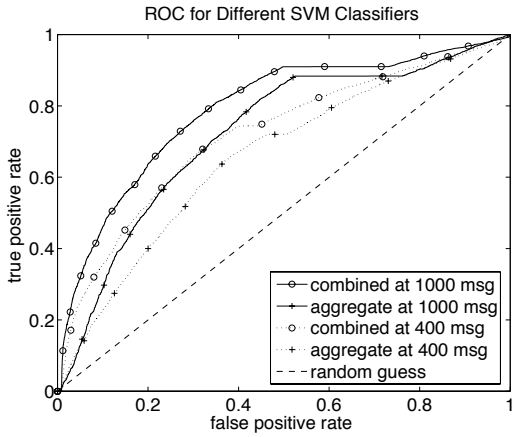
4.2 SVM Features and Results

Two different SVM classifiers were used for each experiment, differing only in the features used. One SVM used only those aggregate features which occured in the logs, which is similar to the approach taken by [8]. Since only 190 tag values occurred, only 190 features are possible. The second classifier used aggregate and a spectrum representation of the messages, will be referred to a combined features. The spectrum representation considered sequences of 5 messages. Tag values were associated with one of 19 ranges. The range values were determined from the tag histogram and group the most popular tag values together. As a result, 2,476,289 features are possible.

As previously described, given 1200 messages, the



(a) Percent accuracy as the number of messages processed increases, for two different SVM feature sets.



(b) ROC curves for different SVM feature sets and number of messages (400 or 1000).

Figure 5: Percent accuracy and ROC curves for predicting hard disk failure. SVM features consisted of either aggregate message information only, or a combination of aggregate and sequence information. SVM with combined features consistently performed better.

first 400, 600, 1000, and 1100 messages were used to make a prediction. Figures 5 and 6 show the performance as the messages are processed. As seen in figure 5(a), the accuracy of the aggregate features ranged from 64% to 70% as the number of messages increased from 400 to 1100. These values are commensurate with other SVM based approaches. The increase in accuracy is expected since more messages provide a more information to make a prediction. The precision and recall percentages were similar to the accuracy.

The performance of the SVM using the combined features (aggregate and spectrum representations) was higher than the SVM using only aggregate features. The accuracy ranged from 67% using only 400 messages to 73% using 1000 messages and 74% using 1100 messages. Therefore at 200 messages before the failure event (approximately 60 hours before failure), the SVM can predict failure with 73%. The precision and recall had similar values. This increase in performance indicated the addition of sequence information can improve performance.

The ROC curves, given in figure 5(b), show the same performance benefits. At 400 messages the combined features has an advantage over aggregate features only. The AUC was 0.72 for the combined features and 0.65 for the aggregate features. Aggregate features are only slightly better than a random classification. However when the number of messages processed is 1100, there is an increase in performance. The AUC is 0.79 for the combined features and only 0.72 for the aggregate. The small linear portion of the combined feature ROC curves indicate classifier generated false positives for certain sets of messages. Regardless, the combined features consistently performed better.

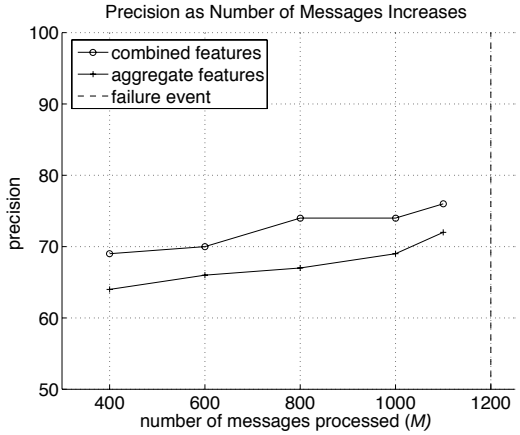
The importance of certain features can be considered by analyzing the feature weights and is possible since the linear kernel was used. Figure 7 shows the weights of the aggregate and sequence features. Of the 2,476,289 possible combined features, only 2,251 had a non-zero weight when used for failure classification. A positive weight indicates the feature is useful for classifying a disk failure, while a negative weight is useful for classifying a non-failure. Larger values (positive or negative) indicate a better feature.

As seen in figure 7(a), only 22 aggregate features were used for classification. Of these features 9 were for non-failure and 13 were for disk failure. The remaining 2,232 features, seen in figure 7(b) represented the occurrence of different message sequences. As noted from the experimental results, the message sequences play an important role in failure prediction.

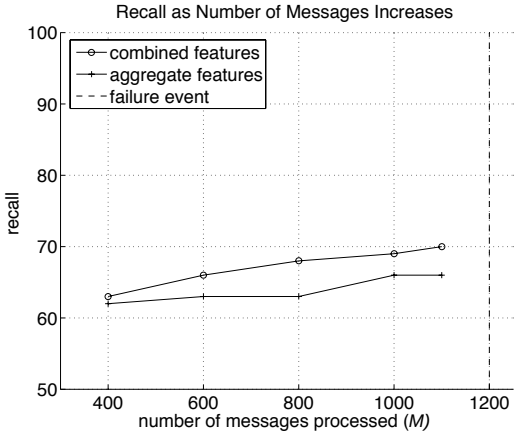
5 Summary and Conclusions

Computer system failures result in the loss of productivity and increased operation cost [7]. Better management of computer resources, applications, and services is possible given accurate predictions of system failures.

Log files typically contain useful information about system failures. These files record the history of the system's state which provides administrators information to determine the causes of critical events. Although log file analysis has been primarily performed after an event has occurred, increasingly this information is being used to

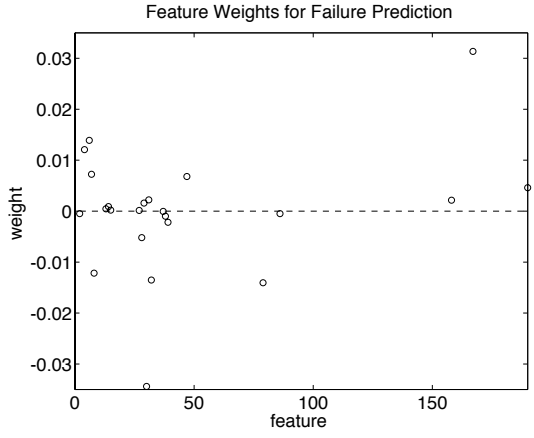


(a) Precision for two different SVM feature sets.

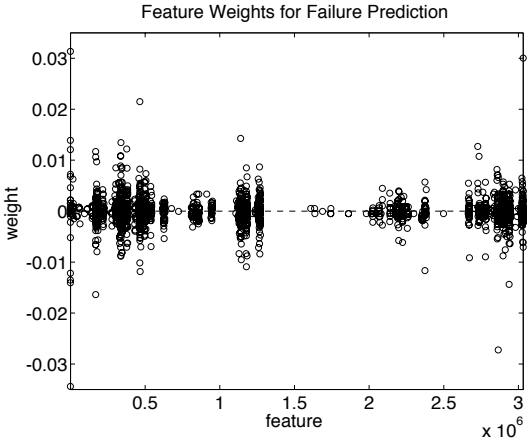


(b) Precision for two different SVM feature sets.

Figure 6: Precision and recall for predicting hard disk failure. SVM features consisted of either aggregate message information only, or a combination of aggregate and sequence information. SVM with combined features consistently performed better.



(a) Aggregate feature weights.



(b) Message sequence feature weights.

Figure 7: Feature weights for the SVM classifiers. Each feature has a unique number, where 0 through 160 are aggregate features (frequency of a certain tag number) and the remaining numbers represent certain message sequences. Positive weights indicate the feature classifies a failure, while negative weights classify a non-failure.

predict events.

This paper introduced a new system failure prediction method using Support Vector Machines (SVM) based on the information contained in log files. The proposed approach takes advantage of the sequential nature of log messages and determines which sequence of messages are precursors to failure. Messages were represented using the tag value, which offers an indication of message criticality. A spectrum kernel representation of the tag values was then used to describe a set of messages, which measures the frequency of k-length message sequences. Experimental results using log files from a large 1024 node Linux-based compute cluster indicate the the spectrum-representation of messages combined with a SVM classifier can achieve an accuracy of 73% and a two day lead (amount of time before the event). Results also show that more log messages consistently provide better predictions.

Although the results indicate the spectrum-representation has promise, there are several open questions. Additional features, such as message interarrival times [8], should be considered. The performance of the system could also be improved by leveraging the message content, instead of solely relying on the tag value. For example, the information provided by SMART messages could be used as a feature. However adding more message information must be balanced with the sequence length, since the number of features grows exponentially. Finally, more research is needed to study the impact of message diversity, which can be the result of machine purpose and log generation rates.

References

[1] ADIGA, N., AND ET AL. An overview of the bluegene/l supercomputer. *Supercomputing, ACM/IEEE 2002 Conference* (Nov. 2002), 60–60.

[2] CHUANHUAN YIN, S. T., AND MU, S. Using gap-insensitive string kernel to detect masquerading. In *Proceedings of the First International Conference on Advanced Data Mining and Applications* (2005).

[3] FAWCETT, T. An introduction to roc analysis. *Pattern Recognition Letters 7* (2006).

[4] FU, S., AND XU, C.-Z. Exploring event correlation for failure prediction in coalitions of clusters. In *Proceedings of the IEEE/ACM International Conference on High Performance Computing, Networking, Storage and Analysis (SC) 2007* (Reno, NV, USA, Nov. 15-21, 2007), pp. 1–12.

[5] GARFINKEL, S. *Practical UNIX and Internet Security*. O'Reilly, 2003.

[6] LESLIE, C., ESKIN, E., AND NOBLE, W. S. The spectrum kernel: A string kernel for svm protein classification. In *Proceedings of the Pacific Symposium on Biocomputing* (2002), pp. 566–576.

[7] LI, Y., GUJRATI, P., LAN, Z., AND HE SUN, X. Fault-driven re-scheduling for improving system-level fault resilience. In *Proceedings of the IEEE International Conference on Parallel Processing* (2007).

[8] LIANG, Y., ZHANG, Y., XIONG, H., AND SAHOO, R. Failure prediction in ibm bluegene/l event logs. In *Proceedings of the IEEE International Conference on Data Mining* (2007).

[9] PINHEIRO, E., WEBER, W.-D., AND BARROSO, L. A. Failure trends in a large disk drive population. In *Proceedings of the USENIX Conference on File and Storage Technologies* (2007), pp. 17–29.

[10] SCHROEDER, B., AND GIBSON, G. A. Understanding failures in petascale computers. *Journal of Physics 78* (2007).

[11] STEARLEY, J., AND OLINER, A. J. Bad words: Finding faults in Spirit's syslogs. In *Proceedings of the 8^{th} IEEE International Symposium on Cluster Computing and the Grid (CCGrid) 2008: Workshop on Resiliency in High Performance Computing (Resilience) 2008* (Lyon, France, May 19-22, 2008).

[12] TANTAWI, A. N., AND RUSCHITZKA, M. Performance analysis of checkpointing strategies. *ACM Trans. Comput. Syst. 2*, 2 (1984), 123–144.

[13] WILLIAM H. TURKETT, J., KARODE, A. V., AND FULP, E. W. In-the-dark network traffic classification using support vector machines. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2008).

[14] XUE, Z., DONG, X., MA, S., AND DONG, W. A survey on failure prediction of large-scale server clusters. In *Proceedings of the International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing* (2007), pp. 733–738.

[15] YAMANISHI, K., AND MARUYAMA, Y. Dynamic syslog mining for network failure monitoring. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (2005), pp. 499–508.