Accurate Fault Prediction of BlueGene/P RAS Logs Via Geometric Reduction

Joshua Thompson[1], David W. Dreisigmeyer[2], Terry Jones[3], Michael Kirby[1], and Joshua Ladd[3]

[1]101 Weber, Department of Mathematics, Colorado State University , Fort Collins CO 80523, email: {thompson,kirby}@math.colostate.edu, Contact author: thompson
[2]601 Thackery Hall, Department of Mathematics, University of Pittsburgh , Pittsburg PA, 15260, email: david.dreisigmeyer@gmail.com
[3]Oak Ridge National Laboratory, Oak Ridge TN 37831, {laddjs,trjones}@ornl.gov

Abstract

This investigation presents two distinct and novel approaches for the prediction of system failures occurring in Oak Ridge National Laboratory's Blue Gene/P supercomputer. Each technique uses raw numeric and textual subsets of large data logs of physical system information such as fan speeds and CPU temperatures. This data is used to develop models of the system capable of sensing anomalies, or deviations from nominal behavior. Each algorithm predicted event log reported anomalies in advance of their occurrence and one algorithm did so without false positives. Both algorithms predicted an anomaly that did not appear in the event log. It was later learned that the fault missing from the log but predicted by both algorithms was confirmed to have occurred by the system administrator.

Keywords: *high performance computing,fault prediction,resiliency,MSET,NMF*

1. Introduction

This report concerns the detection and prediction of anomalous behavior in large data sets of supercomputer system logs. The goal is to identify a system failure in advance and to maximize the *leadtime* of an alarm, i.e., the difference between the timestamp of an alarm and the timestamp of the nearest fault. The problem is challenging given the large amount of information available at any instant in time, that must be analyzed to determine if a failure is imminent.

We propose two distinct approaches for solving this prediction problem. First, non-negative matrix factorization is used to build a model based on observed failures. Then, new testing data is presented to the system and the model bases its prediction on the similarity of the new data to the failure data. As such it may be viewed as a library based algorithm that exploits exemplars of behavior that is to be detected. However, it is based on a function evaluation of the new exemplar rather than a pattern matching approach. Secondly, we apply the MSET algorithm to the same data set. This involves a smooth auto associative map which approximates the identity on non-fault data and perturbs fault data. The residual difference between a data point and its image is a meausre of novelty. When new data contains novelty the mapping produces a larger than expected residual indicating a future failure.

In particular we examine data logs generated by Oak Ridge National Laboratory's Blue Gene/P supercomputer. Each technique uses carefully culled subsets of large data logs of physical system information such as fan speeds and CPU temperatures. We restrict our attention to the most severe errors that were identified in the event log as fatal.

2. Related Work

There has been substantial work in the general area of data analysis for computer fault prediction. As such, the field has proven fertile for a wide-range of techniques and has sparked diverse efforts yielding varying degrees of success. Our work joins the collection with several notable novel aspects. To our knowledge, we are the first to utilize these mathematical approaches and relatively few studies have dealt with large machines. Moreover, we utilize the raw, unreduced numeric values available from hardware resources in concert with text-based information available from system event logs; most studies have typically focused

on one or the other. Finally, our accuracy in terms of both percentage of events forecasted, and the suppression of false positives, is notable.

Our work is most closely related to that of Leangsuksun et al., which concentrates on the same domain (high performance computing) and incorporates some of the same motivations (improved resiliency as seen by applications), although we employ different techniques. Leangsuksun used polling and logs-mining to acquire status and events including hardware resource information. Our work differs primarily in the mathematical analysis employed and in the use of raw data; Leangsukuns team gathered event information and classified it into categories based on sensor type, threshold level, and severity (critical, ok, etc.,), then subjected the classified data to filtering and subsequent analysis [8].

The use of classification as a primary step is not uncommon. Vilalta et al., used 26,000 system events, some of which were labeled critical by a person. Their analysis then concentrated on two types of target events labeled as critical [14, 13]. Lan et al., utilized a three phased approach in their work to analyze Blue Gene logs. In the first pass, data was scrubbed, the second pass consisted of multiple predictor schemes based on meta learning; and a third phase used meta learning and the combination of inputs from the second phase [8] Our use of unreduced data provided more accurate predictions albeit at the potential cost of scalability; the data size for 110 days of our Blue Gene/P machine proved feasible, but much longer periods or much larger machines may require modifications to our method.

Learning algorithms are a common theme in the literature. Sahoo et al., found rule-based classification algorithms can predict critical events with 70% accuracy. They used filtered system activity reports consisting of six fields (representing time, process number, user time, idle time, CPU time, I/O time) together with node topology and event logs [12]. Liang et al., used a customized nearest neighbor-based classifier strategy on system event logs. They found 529 failure events from 128K processors over 141 days (3.75 events per day) [10]. Here again, our geometric reduction based techniques stand apart in terms of accuracy and false-positive suppression.

The related field of failures among telephone switch equipment has also provided applicable contributions. Weiss et al., used a genetic algorithm to predict telecommunications equipment failures based on machine learning. Training based on 110,000 alarms reported from 55 4ESS switches, test set contained 40,000 alarms reported from 20 different 4ESS switches [16, 15].

Two more teams who utilized an ensemble of system information for fault prediction are located at Humboldt University and Sandia National Laboratory. Hoffman et al., used system data collected by SAR, a utility typically used for job accounting or system diagnostics, to predict events captured by system error logs (cputime, pageswaps, etc.,) [5]. The work of Brandt et al., is primarily intended for high performance clusters and cloud computing, but the same techniques are applicable to leadership class systems. They utilize statistical tools to determine anomalous behavior through outliers. Like our work, the Brandt work does include sensor data (e.g., cpu temp, core voltage, fan); they determine outliers by statistical means from raw data or by comparison to a given model [1, 7]. Our work differs in the mathematical analysis employed and the high sampling rates of our data.

3. Data Preprocesing

The log data consisted of many components and subcomponents each reporting asynchronously and sometimes erroneously. A good deal of the workload fell then to the pre-processing step, whose goal it was to produce usable, representative data. In this report we restrict our attention to one subset S of the system, namely data generated by *Rack 00 - Midplane 00*. Although the system behavior of adjacent midplanes was observed to affect each other in this preliminary report we simply use data generated by S to predict faults occurring in S.

Prior to any scaling or cleaning of the data, we performed a spline interpolation on all of the time series. This was done in order to have synchronous data. A scaling of the data was then performed so that large magnitude signals did not dominate potentially important low-magnitude signals. The scaling itself was a simple invertible, affine mapping of the raw data to the interval [0,1]. Portions of the raw data were corrupt, and contained sporadic extreme values. After the initial rescaling, we removed any obvious outliers and repeated the scaling. This was continued in an iterative fashion until the data was deemed 'clean'. While the spline interpolation was performed prior to any data cleaning, this did not affect any later analysis since the removed data was located far away from any faults.

A matrix D was created which contained all available numeric data. Groups of sensor data (fan speeds, voltages, etc.,) were represented in D as blocks of rows. That is a row of D was a time series of values for a particular sensor, eg. fan speed. A column of D was a snapshot of all sensors at a given time. In general a data point from here forth refers to a column of D.

The two prediction methods presented each chose various sub-rows of D for its analysis. This choice was guided by the geometry of each of the algorithms. In S there were six clusters of faults and the table below illustrates a portion of our results.

FAULT CLUSTER	LEADTIME AFFINE MAP	LEADTIME IDENTITY MAP
1	2.5 min.	10.4 hours
2	5 min.	10.2 hours
3	80 min.	2.5 hours
4	32.5 min.	1 hour
5	10 min.	7 min.
6	-2.5 min.	-22 hours

Table 1. Fault Prediction Results

4. Affine Transformation and Threshold Detection

Since all of the data was positive, a Nonnegative Matrix Factorization (NMF) was performed on the data. This is a fairly standard technique [9] for data of the type considered in this paper. Here the initial nonnegative data matrix X is decomposed into two nonnegative matrices H and W:

$$X \approx H * W.$$

The columns of H give the important directions in feature space. NMF is a generalization of k-means clustering [17, 3]. So the columns of W can also be considered cluster centers. The columns of W classify the data into the various clusters, with the largest entry w_{ij}^* classifying the j^{th} point into the i^{th} cluster. In the present situation, we would expect the fault data to be separated from the normal operating data. What we would like to do is find a cluster for the fault data. NMF in conjunction with a thresholding for alarm signaling performed extremely well on this problem, consistently finding a vector that gave a rough classification for fault data.

For testing our prediction method, we either used the last three clusters to predict the preceding faults, or used the first three clusters to predict the latter faults. The affine transformation was done in a stepwise fashion. Given the fault times, we wished to find data points with an obvious fault signature. Given the rough classification above, the second step of our method was to use this to pick out a good direction for separating fault data from non-fault data.

The classification direction can be refined by performing a Generalized Linear Discriminant Analysis (GLDA). In the present case of only two classes (fault and non-fault), GLDA will result in a single direction that maximizes the between class separation and minimizes the within class scattering. The details of the method can be found in [6]. The numerical method used to solve the problem is the Generalized Singular Value Decomposition [4]. The result is that the fault data F and non-fault data N our decomposed as

$$F = M^{-1} * C * V^T$$
$$N = M^{-1} * S * W^T$$

where V and W are orthogonal and M is invertible. The matrices C and S are diagonal and

$$\begin{array}{ll} c_{ii} \geq c_{jj} \geq 0 & \text{when } i > j \\ 0 <= s_{ii} <= s_{jj} & \text{when } i > j \\ C^2 + S^2 = I. & \end{array}$$

Since c_{11} is the largest element of the above decomposition and s_{11} is the smallest, the first row of M gives the direction of optimal classification. In this direction the fault data will have the largest magnitude and the non-fault data will have the smallest magnitude.

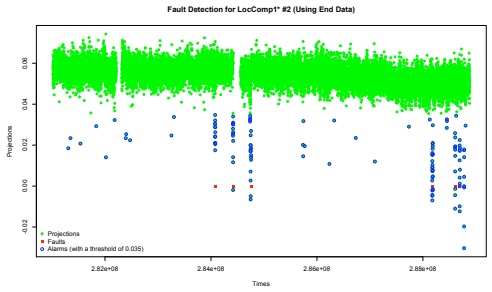
Note the logic here. The NMF is an initial clustering routine that performs its task semi-independently. The user must pick out the NMF direction that corresponds to the fault clustering. However, the data is not a priori labelled. The classification given by NMF is then used to separate the data into two classes. These are now fed into the GLDA algorithm which refines the direction used to classify the data. We found that the direction found by the GLDA algorithm was significantly better than that provided by the NMF algorithm for final classification of the algorithm. This method can be easily extended to multiple clusters.

The direction chosen by NMF should not be considered optimal for classification of the data, though it does seem to provide a good clustering algorithm. However, we can improve the classification direction by performing a Generalized Linear Discriminant Analysis (GLDA) [6] on the data. Here, every point that NMF classifies as fault is windowed, so that points within 50 time steps are not considered non-fault data. The data that remains after this is considered non-fault data, one of the classes for GLDA. The points that NMF classifies as faults are the second GLDA class.

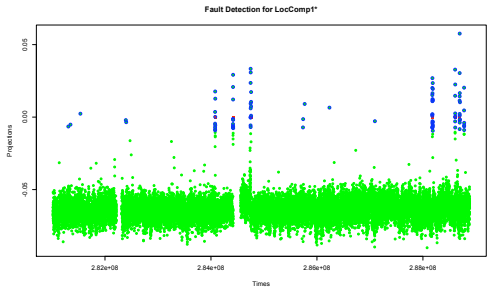
4.1. Results

After finding the optimal (GLDA) direction, incoming data is simply scaled and projected onto the one-dimensional subspace. This magnitude was then thresholded to classify a new time point as sounding an alarm or not. A sliding window of length three sounded a fault detection if there were two alarms within the window. This lead to prediction of five out of six faults with leadtimes of approximately 2.5 mins, 5 mins, 80 mins, 32.5 mins, 10 mins, and -2.5 mins.

The last time is the fault we missed, though from the system administrator we know there was a fault three minutes after this timestamp. This discrepancy may be attributed to the coarsness of the data sampling. We discuss this more in the next section. These results are using the first three clusters to predict the last three. Similar results were found



(a) This uses the last three fault clusters to predict the earlier ones.



(b) This uses the first three fault clusters to predict the latter ones.

Figure 1. Fault detection using an affine transformation with a windowed thresholding.

when we used the last three clusters to predict the first three. See Figure 1.

5. Detection Using a Mapping of the Identity on Healthy Data

The Multivariate State Estimation Technique (MSET) is a well-known non-parametric technique that has been used to predict anomalous system behavior in large systems [18]. Since coordinates of data points correspond to physical properties like fan speeds and temperatures, it is assumed that data points near faults will have different geometry than data points arising from healthy periods of the system. These differences may be subtle, however, and an MSET algorithm can be tuned to reveal and exploit these differences.

Data that is considered to be healthy, X is used to define a non-linear dimension preserving transformation Φ of the data, which factors through a map φ of the data into a so-called similarity space. New data points that share similarities with X are left relatively unchanged by Φ, while points that are unfamiliar to X are changed significantly by Φ. Thus the geometrical differences between a data point and its image under Φ are used to define a *Residual* which is used as a threshold to predict faults. We achieved the best results when we compared both the distance and the angle between a data point and its image under Φ. In the results below, we predict a fault at time t if at time t the MSET residual is above the given threshold.

The similiarity map φ is computed using an inverse matrix M^{-1}, which may be calculated or approximated. This calculation is done only as often as one needs to create a new model for the system and may be done off-line. To create a mapping representative of a large range of data (weeks,months) efficiently, we cluster the data into 1500 model points using the well-known LBG clustering algorithm [2]. We applied this algorithm to various subsets of X, but saw the best results when applied to smaller, large-valued portions of X. Nominal states of the system often exhibited near constant behavior which caused the matrix M to be non-invertible. Therefore a regularization technique was applied which slightly *spheres* the data into an invertible matrix that is a close approximation to the original data set. This has virtually no effect on the prediction ability.

5.1. MSET Details

A memory matrix X of size (m, n) is obtained from a collection of sensors over time. In the case here, X was created by clustering roughly 10K points of D into 1200. The m^{th} row of X contains n samples from the m^{th} sensor. The i^{th} column is an observation vector $X^{(i)}$ of the system at time i

$$X^{(i)} = [x_1(i), x_2(i), \ldots, x_n(i)]^T$$

where $x_j(i)$ is the measurement from sensor j at time i.

Given a new pattern P we construct a feature vector W which exposes hidden similarity between each $X^{(i)}$ and P. This feature vector is found by applying a chosen non-linear kernel operator $\otimes$ to the memory matrix X and the pattern P as shown

$$W \equiv W(P) = (X^T \otimes X)^{-1}(X^T \otimes P).$$

The matrix $X^T \otimes X$ is called a *similarity matrix* and expresses the self-correlations of the samples $X^{(i)}$ in the memory matrix. The matrix $X^T \otimes P$ expresses the similarity of the given pattern P with each sample in the memory.

Definition 1 *Given $X \in R^m \times R^n$ and $Y \in R^m \times R^k$, we define $X \otimes Y \in R^n \times R^k$ as the matrix whose (i, j) coordintate is given by*

$$X \otimes Y_{(i,j)} = 1 - \frac{\| X^{(i)} \|^2 - \| Y^{(j)} \|^2}{\| X^{(i)} - Y^{(j)} \|^2}.$$

The MSET mapping φ defined as

$$\varphi(P) \equiv X * W(P)$$

where $*$ indicates the matrix product, maps the feature vector $W(P)$ onto a particular linear combination of the observed patterns.

The i^{th} coordinate of the feature vector corresponds to the i^{th} pattern vector in the memory matrix X. So $\varphi(P)$ is an approximation of P in terms of the observed patterns.

Assume the pattern P is contained in X, say $P = X^{(j)}$. The feature vector $W(P)$ then has one non-zero coordinate in the j^{th} position

$$P = X^{(j)} \in X \Longrightarrow W(P) = [0, \ldots, 0, 1, 0, \ldots, 0]^T.$$

Therefore $W(P)$ is transformed by X directly into P. Otherwise P is novel and $W(P)$ will have many non-zero coordinates and $\varphi(P)$ will differ from P.

5.2. Results

The data sets contained six clusters of faults, and we let **N1** denote the non-fault data whose timestamps ranged from $2.81e8$ to $2.83e8$. This is the first large chunk of non-fault data in Figure 5.2. We let **N2** denote the non-fault data whose timestamps fell in the range $2.855e8$ to $2.875e8$. This is the large chunk of non-fault data in Figure 5.2.

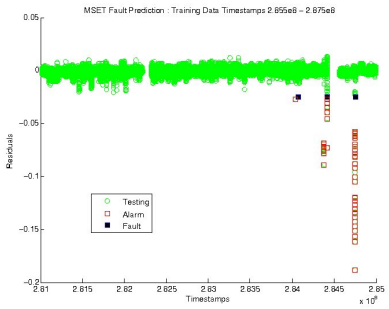
Figures 5.2 and 5.2 below indicate results from our prediction scheme. Each training set was used to predict both the first 3 and the last 3 fault clusters. A *Residual* in the figures below is a scalar multiple of the norm of the difference between data point x_t and its image under Φ

$$\mathcal{R} = \| \Phi(x_t) - x_t \| .$$

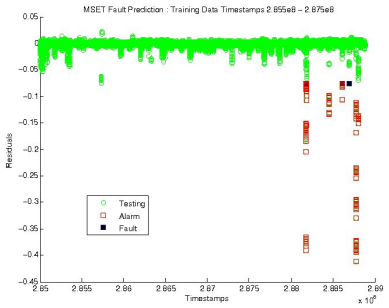
The scalar multiple contains information about the angle at which the two given vectors intersect.

The leadtimes generated from using **N1** to predict the last faults, as well as the leadtimes from using **N2** to predict the first faults are presented in Table 1. The leadtimes generated from using **N1** and **N2** to predict the first and last faults, respectively are: -15 mins, 10.2 hours, 2.5 hours, 55 mins, 13 mins, -22 mins. This iteration of the algorithm missed the first and the last fault. We observed that the first fault is of a different geometric character than the others, and our MSET mapping needs to be further tuned to reflect this.

The last fault was missed however we know from the system administrator that there were other problems with the machine at this time. Also there was another fault that present, that both algorithms caught although it did not appear in the event logs. Indeed, we asked the BlueGene/P System Administrator to check his notes for this time, since



(a) MSET Using **N2** to Predict First Three Faults



(b) MSET Using **N2** to Predict Last Three Faults

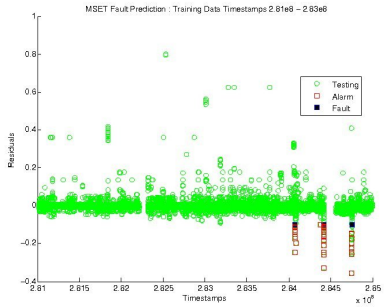
Figure 2. Fault detection using a mapping of the identity on non-fault data

we thought the strong pulse in our Residual was too peculiur to overlook. He reported back that, as part of his own routine checks he observed anamolous behavior near this timestamp. Interestingly, his routine check happend 15 minutes AFTER our predicted alarm. So we not only caught a fault that was missed by the event log, we outperformed a human who happened to be running a routine check during this time.

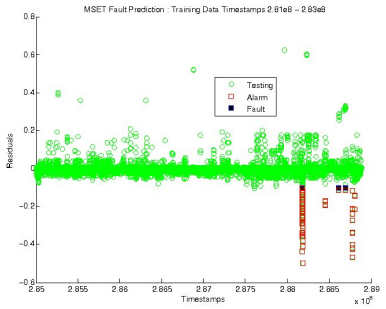
6. Discussions and Proposed Algorithm Enhancement

We have proposed two algorithms for predicting anomalies in supercomputer log data. Each method presented was successful in predicting 5 of 6 event-log recorded anomalies and both algorithms caught an anomaly not registered in the log data. This study was preliminary and the methods were not highly tuned. We now propose several enhancements to the model that should improve their predictive qualities.

The non-negative matrix factorization method is an affine transformation onto a 1-D subspace followed by a classification using the magnitude of this projection. One could generalize our method as follows:



(a) MSET Using **N1** to Predict First Three Faults



(b) MSET Using **N1** to Predict Last Three Faults

Figure 3. Fault detection using a mapping of the identity on non-fault data

- Continue to use the NMF to classify the affinely transformed data into fault and normal. A window will still be put around any NMF predicted fault cluster.

- With this classified data, we will find a separating hyper-plane of the fault from non-fault data. The normal of this hyper-plane will be the projection direction to classify the data by a simple threshold (perhaps using the windowing method above).

- We should be willing to accept misclassifying normal data as faults since these misclassifications tend to be separated in time. If an alarm is sounded when two out of three points are classified as faults, false alarms should be avoided (as above). This needs to be incorporated into our optimization routine.

Notice what we end up with is a SVM classifier with an affine (versus nonlinear) transformation of the original data. A reasonable approach is to use positive slack variables only for the non-fault data. Let N be the set of normal/non-fault data points. Modifying the standard SVM optimization problem results in the primal problem

$$\min : \quad \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i\in N}\lambda_i \tag{1a}$$

subject to the constraints

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq \quad 1-\lambda_i \quad , \text{for } i \in N \tag{1b}$$
$$y_j(\mathbf{w}\cdot\mathbf{x}_j + b) \geq \quad 1 \quad , \text{for } j \notin N \tag{1c}$$
$$\lambda_i \geq \quad 0. \tag{1d}$$

Here the $\mathbf{x}_k$ are the coordinates of the data points, and $y_k = \pm 1$ denote the classification of the data points from the NMF procedure. C is chosen a priori, though an ROC analysis with training and test data could be used to refine the choice. Final classification is given by the function $f(\mathbf{x}) = sign(\mathbf{w}\cdot\mathbf{x} + b)$

Improvements to the MSET algorithm begin with the similarity mapping φ. This map depends on a choice of a *metric* i.e., a way to compare points in the data set. We witnessed classification rates rise when φ was tuned to encode both the distance and the angle between two data points. We propose to extend φ further to encode the subspaces spanned by two points. These subspaces correspond to patterns of physical system behavior such as a combination of fan speed, temperature and voltage. Optimizing the resulting MSET routine over all possible subspaces would not only produce the best fault detection but would simultaneously find the subspaces which carry the most predictive power.

Secondly, when the MSET algorithm is iterated certain points quickly converge to the training data while others

converge to the training data much more slowly. This iteration can be done on-line and in parallel since the main inverse matrix computation need not be repeated and the iteration of any point p_1 is independent of that of p_2. The varying convergence rates provide a measure of an anomaly, i.e., an extremely strong fault will have a slow convergence rate. Such a measure could be used to improve classification and reduce false positives.

The basic regularization approach used here should be optimized, and the windowing method for fault detection is simple yet effective. We expect that a method such as Sequential Probability Ratio Testing (SPRT) [11] will be more effective. Lastly, a similar analysis could also be done with regards to other types of anomaly than the fatal errors discussed here.

7. Conclusions

Raw numeric and textual log data from Oak Ridge National Laboratory's Blue Gene/P supercomputer was used to build two anomaly detection algorithms. The methods, based on a non-negative matrix factorization and MSET, were used to successfully predict extreme events (faults) experienced by the machine.

Non-negative matrix factorization used samples of failures to build a model capable of detecting similar failures. It worked well whether the first three fault clusters were used to predict the last three, or vice-versa. Lead times on the first five clusters were 15+ minutes.

The MSET algorithm was trained using only non-fault data. It was designed to fix any point similar to what it had seen, and perturb points that were unfamiliar. Therefore, in contrast to non-negative matrix factorization, MSET predicts anomalies that it has not seen before. As a result, it is particularly well suited to predict new types of failures in real time. Results were generally consistent regardless of which training set was used. Overall, MSET was able to predict the same instances of failure as non-negative matrix factorization but with lead times from seven minutes to ten hours.

Of the six faults in the event log, five were predicted by both algorithms with lead times ranging from seven minutes to ten hours. Both algorithms predicted a seventh fault, which was confirmed to exist by the system administrator, though it did not appear in the log. The last fault cluster was never predicted, however we have learned from the system administrator that the system did experience a fault near our predicted time.

8. Acknowledgments

This work was made possible by the Directorate of Central Intelligence Postdoctoral Program and Grant HM1582-08-1-0041.

References

[1] J. Brandt, B. Debusschere, A. Gentile, J. Mayo, P. Pebay, D. Thompson, and M. Wong.

[2] Y. L. A. Buzo and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–94, 1980.

[3] C. Ding, X. He, and H. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. *Proc. SIAM Int'l Conf. Data Mining*, pages 606–610, 2005.

[4] G. Golub and C. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[5] G. Hoffman, F. Salfner, and M. Malek. Advanced failure prediction in complex software systems. *SRDS*.

[6] P. Howland and H. Park. Generalizing discriminant analysis using the generalized singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):995–1006, 2004.

[7] O. P. http://ovis.ca.sandia.gov.

[8] C. Leangsuksun, T. Liu, S. Scott, T. Rao, and R. Libby. A failure predictive and policy-based high availability strategy for linux high performance computing cluster. *Proceedings of 5th LCI International Conference on Linux Clusters*, 2004.

[9] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[10] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo. Failure prediction in ibm bluegene/l event logs. *Seventh IEEE International Conference on Data Mining*, pages 583–588, 2007.

[11] J. L. Romeu. Understanding binomial sequential testing. Technical report, Reliability Analysis Center, 2005.

[12] R. Sahoo, A. Oliner, I. Rish, M. Gupta, J. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam.

[13] R. Vilalta and S. Ma. Predicting rare events in temporal domains. *ICDM*, IBM Research T. J. Watson Research Center, Yorktown Heights, NY:474–481, 2002.

[14] R. Vilalta and S. Ma. Predicting rare events in temporal domains using associative classification rules. *Technical Report*, IBM Research T. J. Watson Research Center, Yorktown Heights, NY:426–435, 2002.

[15] G. Weiss and H. Hirsh. Learning to predict rare events in categorical time-series data. *AAAI Workshop*, 1998.

[16] G. Weiss and H. Hirsh. Learning to predict rare events in event sequences. *KDD*, IBM Research T. J. Watson Research Center, Yorktown Heights, NY:359–363, 1998.

[17] R. Zass and A. Shashua. A unifying approach to hard and probabilistic clustering". *International Conference on Computer Vision (ICCV)*, 2005.

[18] N. Zavaljevski and K. Gross. Sensor fault detection in nuclear power plants using multivariate state estimation technique and support vector machines. *Third International Conference of the Yugoslav Nuclear Society YUNCSC 2000*, 3, 2000.