Supercomputer Hardware Fault Detection

Josh Thompson[1], David W. Dreisigmeyer*[,2], and Michael Kirby[1]

[1]Colorado State University, Department of Mathematics
[2]University of Pittsburgh, Department of Mathematics

February 3, 2010

Abstract

This investigation presents two distinct approaches for the prediction of system failures occurring in the Oak Ridge National Laboratory's Blue Gene/P supercomputer. Each technique uses 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 log reported anomalies in advance of their occurrence without false positives. Both algorithms failed to detect the same anomaly.

1 Introduction

This investigation concerns the detection of anomalous behavior in large data sets of supercomputer systems 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 immiment.

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. MSET is based on building a mapping of the identity based on non-fault data. When new data contains novelty the mapping produces a larger than expected residual indicating a future failure.

*Email: david.dreisigmeyer@gmail.com

In this studey we examine data logs generated by the 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. In this study we restrict our attention to the most severe errors that were identified in the system log as fatal.

2 Data Preproccesing

Obtaining usable data from these system logs proved to be a respectable task unto itself. The data consisted of many components and sub-components each reporting asynchronously and sometimes erroneously. A good deal of the workload falls then to the pre-processing step, whose goal 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 system behavior from one rack / midplane pair was observed to affect system behavior of an adjacent rack / midplane pair, we found enough interesting questions to explore by simply using data generated by S to predict faults occuring in S.

Prior to any data transformations the data was cleaned and interpolated with splines. The interpolation was done to create a synchronous time-series of each sub-component. The two prediction methods presented each chose various subsets of S 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	ALARM LEADTIME AFFINE MAPPING	ALARM LEADTIME IDENTITY MAPPING
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

3 Affine Transformation and Threshold Detection

A simple (non-invertible) affine transformation of the data in conjunction with a thresholding for alarm signaling performs very well on the 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. First, the data was

normalized to remove scale effects. Given the fault times we wished to find data points with an obvious fault signature. We found that the Non-negative Matrix Factorization (NMF) [**?**] method performed extremely well in classifying data. So the second step of our method was to use NMF to pick out a good direction for separating fault data from non-fault data.

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) [**?**] on the data. Here, every point that NMF classifies as fault is windowed, so that points with 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 is the second GLDA class.

3.1 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, which may be mis-stamped. These are using the first three clusters to predict the last three. Similar results were found when we used the last three clusters to predict the first three. An additional (very strong) fault signal was found near the end of the time series (approximately 11 hours after the last marked fault). See Figure **??**.

4 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 [**?**]. 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, H is used to define a non-linear mapping Φ of the entire data set. New data points that share similarities with H are left relatively unchanged by Φ, while points that are unfamiliar to H 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. In the results below, we predict a fault soon after time t if at time t the MSET residual is above the given threshold.

The MSET mapping is calculated using an inverse matrix, 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 [**?**]. We applied this algorithm to various subsets of S but saw the best results when applied to smaller more correlated portions of S.

4.1 Results

Figures **??** through **??** below indicate results from our prediction scheme using either of the following two groups of training data, A whose timestamps ranged from $2.81e8$ to $2.83e8$ and B whose timestamps fell in the range $2.855e8$ to $2.875e8$. Each training set was used to predict both the first 3 and the last 3 fault clusters. The map Φ is constructed to encode the angle and magnitude of the the training data. A *Residual* in the figures below is the difference between the norm of a data point x_t and the norm of its image under Φ.

The leadtimes generated from using A to predict the last faults, as well as the leadtimes from using B to predict the first faults are presented in Table 1. The leadtimes generated from using A and B 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 observe 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 we think may be mis-labeled, since the strong signal that occurs at the end of our data set is not marked as a fault.

5 Dissusions 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 anomalies and both algorithms suggested the presence of an anomaly not registered in the log data. This study is 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:

- 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.

6 Conclusions