April 9, 2007

RT0723

Computer Science 12 pages

Research Report

Failure Prediction of Cluster Computer Systems Based on Point Process

Toshiyuki Yamane, Hideki Tai and Takayuki Kushida

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Failure Prediction of Computer Systems Based on Point Process

Toshiyuki Yamane*, Hideki Tai and Takayuki Kushida

Abstract

A method is presented for predicting the critical events which can lead to system down. This method is based on the point processes and online estimation of intensity function. By fitting a model of intensity function to the latest part of the point process data, we can predict the expected number of occurence of events after time T. We can also estimate how likely the system will go down after time T. Simulation results are also presented for artificially generated point process data.

1 Introduction

In recent years, a huge number of computational resources available for high-performance computing. In such computational environment, these systems inevitably suffer from relatively frequent system failures, such as disk failures, processor malfunctioning and memory errors. Therefore, proactive actions in system management are becoming more and more critical to prevent jobs from interruption or restart caused by failures. In particular, failure prediction technique is quite useful to enable us to take actions such as detatchment or replacement of suspicious component prior to critical failures. There exist some techniques to predict the future behavior of computer systems such as time series prediction. However, all methods based on stationary models cannot work for failure prediction because failures are sporadic events and are accompanied by usually non-stationary signs. In [7], some predictive algorithms for computer system activity are discussed including threshold violation. However, their method uses stationary models and may fail to catch the non-stationary sign of failures.

In this paper, we mainly pay our attention to memory failures. It is reported that memory failures account for 12% of root cause of system failures and considered to be one of the major cause of failures [5]. The degradation of memory modules generally follows gradual and phased process. In normal state, the errors rarely happens and the memory operates normally. However, some accidental exogenous factor such as cosmic rays can ocassionaly affect the memory

*{tyamane, hidekit, kushida}@jp.ibm.com

cells and convert the content of the momory cells. This is called soft errors, and can be detected and corrected as long as the number of errors stays within ECC capability. In this case, errors are called to be latent, that is, they are invisibe to system. On the other hand, the physical memory suffers from some faults due to aged deterioration, more and more momory cells become unreadable or unwritable in an accelarated pace. This is called hard errors and involves more frequent ECC events. When the memory deterioration finally transceneds the ECC capability, the memory cells become totally unavailable. So we can assume that the sign of system failure can be captured by increase of ECC frequency. In addition, we assume systems with more errors than ECC capability is unreliable and may go down. This picture of life of memory modules have the same view of bathtub curve used in reliability engineering.

The first step is to model the error events as a point process. The point process is an record of time stamps when events of interest happened. In our case, the time stamp is an occurence of ECC. To catch this signs of memory failures beyond ECC capability, we use a non-stationary Poisson process. If we can predict the frequency of the ECC in a online manner by non-stationry Poisson process, we can take some proactive actions while errors are latent.

2 Existing Failure Management Techniques

In this section, we summarize the existing failure management techniques which are embedded in commercially available systems.

2.1 First Failure Data Capture

First Failure Data Capture(FFDC) is a built-in reliability feature for self-healing and self-diagnosing, and employed by IBM eServer pSeries[2]. FFDC is composed of multiple error checking devices embedded in the system and used for recording persistent records of failures and significant software incidents. On occurence of the first error, it captures various log files, trace files, dumps and snapshots that describe where and when problems are detected. It also provides a means of associating failures to one another and thereby enable discovery of the root cause of a failure. Therefore, FFDC can eliminate time-consuming task of reproducing errors at the problem determination by identifying the parts to be replaced. FFDC can prevents critical failures from leading to a system failure and thus reducing repair time.

2.2 Predictive Failure Analysis

IBM eServer xSeries and Netfinity employ a dedicated service processor for reliability prediction called Predictive Failure Analysis (PFA), which generate early warnings when fault tolerant function catches recoverable error events. PFA-enabled hardwares include hard disk drives, cooling fans, processors, physical memories, and power supplies. Using advanced heuristic techniques and

periodic self-diagnostics, Predictive Failure Analysis (PFA) can detect when components operate abnormally out of specifications or approaching historical failure thresholds. PFA can predict the failure of supported components, often 48 hours before it occurs. Therefore, administrators are allowed to take immediate actions such as replacement of parts, or backups of valuable data, prior to actual failures. For example, as for hard disk drives, PFA measures several attributes, such as head flying height, to predict failures. When PFA finds aberration of flying height, PFA generates a warning to the host that a failure may occur in the near future. Since PFA basically relies on threshold logic, PFA can be useful for short term prediction. However, since it cannot predict how latent errorneous states evolve in the future, it is not helpful for scheduled action. If we can predict the future state while errors are latent, it can lead to planned system management action.

2.3 Error Checking and Correction

In general, fault-tolerant system refers to a system design which enables the system to keep running, possibly at a reduced performance, rather than failing completely, even when some part of the system fails. That is, the system as a whole is not stopped even in the presence of hardware or the software failures as long as they stay within the fault tolerance of the system. The heart of fault torelant system lies in the redundancy of the system which prevents the errors from manifesting themselves and leading the system down. For example, HDD RAID system, memory chipkill^{TM}. In this case, the errors are called to be latent and are not visible from the oustside of the system. In our context, the fault tolerant feature gives us signs of crucial failure events in the future because it enables the system to continue to work under not so severe erroneous state.

Error checking and correction (ECC) has been widely and successfully used in a number of practical applications such as storage and communication systems. ECC circuitry for testing the accuracy of data as it passes in and out of memory. ECC appends extra parity bits to the original data bits to correct single error bit and detect double-bit errors and some triple-bit errors. In recent years, error detection and correction (EDAC) are supported at the Linux kernel module level [1]. EDAC can capture the events of error detection and correction and report them to syslog of Linux. Figure 1 shows the example of syslog record of Linux EDAC messages.

Figure 2 shows the timestamp of events of error correction or detection.

3 Fault prediction technique based on point process

3.1 A short review of point process

A point process is a type of stochastic process and represents a sequence of time stamps when events of our interest happened. The realization of a point process

```
1/7 7:38:49 percs-kb kernel: EDAC MC0: UE page 0x2c, offset 0x0, grain 4096, row 0...
1/7 11:52:31 percs-kb kernel: EDAC MC0: UE page 0x2c, offset 0x0, grain 4096, row 0...
1/7 18:37:28 percs-kb kernel: EDAC MC0: UE page 0x2c, offset 0x0, grain 4096, row 0...
1/9 7:09:01 percs-kb kernel: EDAC MC0: UE page 0x2c, offset 0x0, grain 4096, row 0...
1/9 10:35:52 percs-kb kernel: EDAC MC0: UE page 0x2c, offset 0x0, grain 4096, row 0...
1/10 2:16:20 percs-kb kernel: EDAC MC0: UE page 0x2c, offset 0x0, grain 4096, row 0...
1/10 10:31:16 percs-kb kernel: EDAC MC0: UE page 0x2c, offset 0x0, grain 4096, row 0...
1/10 23:26:09 percs-kb kernel: EDAC MC0: UE page 0x2c, offset 0x0, grain 4096, row 0...
1/11 14:31:39 percs-kb kernel: EDAC MC0: UE page 0x2c, offset 0x0, grain 4096, row 0...
```

Figure 1: Example of syslog output of EDAC

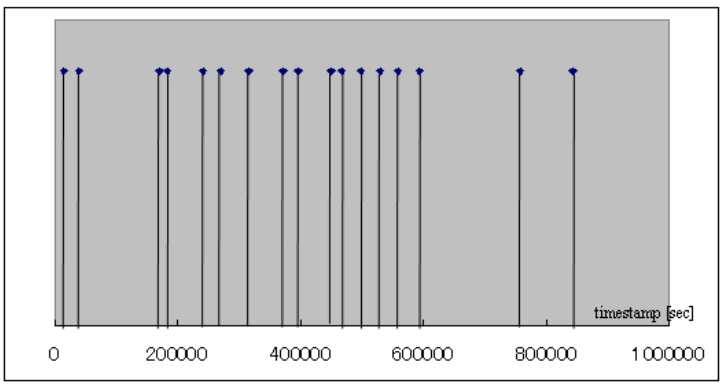

Figure 2: Point process data of EDAC. Each spike shows the timestamp of EDAC event.

is denoted as $(t_1, t_2, \cdots, t_N)$, where the number of events N can be different for different observations. A point process is characterized by its intensity function (also called hazard function) $\lambda(t)$ defined by

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} P(N(t, t + \Delta t] = 1). \tag{1}$$

or equivalently

$$P(N(t, t + \Delta t] = 1) = \lambda(t)\Delta t, \tag{2}$$

where $N(t, t+\Delta t]$ is the number of events that happens during the time interval $(t, t + \Delta t]$. Intuitively, the intensity function shows the degree of tendency for just one event to happen in a very short period of time. The relation between the interval distribution for successive events and the intensity fucntion is given as follows:

$$\begin{aligned}
\lambda(t) &= \lim_{\Delta t \to 0} \frac{1}{\Delta t} P(N(t, t + \Delta t] = 1) \\
&= \lim_{\Delta t \to 0} \frac{1}{\Delta t} P(t < L < t + \Delta | L > t) \quad (L \text{ is the timestamp of an event}) \\
&= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \frac{P((t < L < t + \Delta) \text{ and } (L > t)))}{P(L > t)} \\
&= \frac{1}{P(L > t)} \lim_{\Delta t \to 0} \frac{P(t < L < t + \Delta)}{\Delta t} \\
&= \frac{1}{P(L > t)} \lim_{\Delta t \to 0} \frac{F(t + \Delta t) - F(t)}{\Delta t} \quad (F(t) \text{ is the cumulative distribution function of } L.) \\
&= \frac{F'(t)}{1 - F(t)}.
\end{aligned}$$

The above differential equation can be solved analytically and the solution is given by

$$F(t) = 1 - \exp\left(- \int_0^t \lambda(s) ds \right). \tag{3}$$

3.2 Non-stationary Poisson process modelling and its estimation

Suppose we have point process data $t_1, \cdots, t_n$ during period of time of length T. The probability that no events occur during time interval $(t_{i-1} + \Delta t_{i-1}, t_i]$ and just one event occurs during time interval $(t_i, t_i + \Delta t_i]$ is given by

$$\exp\left(- \int_{t_{i-1} + \Delta t_{i-1}}^{t_i} \lambda(t) dt \right) \lambda(t_i) \Delta t_i \tag{4}$$

Since each occurence of events are independent, the joint probability is given by the product of all the terms shown above:

$$f(t_1, \cdots, t_n)\Delta t_1 \cdots \Delta t_n = \exp\left(- \int_0^{t_1} \lambda(t) dt \right) \lambda(t_1) \Delta t_1 \cdots \exp\left(- \int_{t_n}^{T} \lambda(t) dt \right) \lambda(t_n) \Delta t_n$$

$$= \prod_{i=1}^{n} \lambda(t_i) \exp\left(-\int_0^T \lambda(t)dt\right) \Delta t_1 \cdots \Delta t_n.$$

Dividing both sides by $\Delta t_1 \cdots \Delta t_n$ and letting $\Delta t_i \longrightarrow 0$ in (4), we can obtain the joint probability density function as

$$f(t_1, \cdots, t_n) = \lambda(t_1) \cdots \lambda(t_n) \exp\left(-\int_0^T \lambda(t)dt\right) \tag{5}$$

3.3 Models of intensity function and its estimation

We fit some models of $\lambda(t)$ to the latest part of point process data to see if the intensity is increasing or decreasing. In reliability engineering, the failure rate of system in its entire life follows a U-like profile, which is called bathtub curve. The Weibull distribution is often used to describe the bustub curve. The Weibull distribution is a interval distribution of point process with intensity function

$$\lambda(t) = \lambda p(\lambda t)^{p-1}. \tag{6}$$

The interval distribution for this intensity function is given by equation (3) as follows:

$$\begin{aligned} F(t) &= 1 - \exp\left(-\int_0^t \lambda(s)ds\right) \\ &= 1 - \exp(-(\lambda t)^p), \end{aligned}$$

which is known as Weibull distribution.

The parameter p controls how likely failures occur. That is,
- $p < 1$ gives decreasing failure rate (called early infant mortality failure),
- $p = 1$ gives constant failure rate,
- $p > 1$ gives increasing failure rate (called wearout failure).

Therefore, the choice of $\lambda(t) = \lambda p(\lambda t)^{p-1}$ as intensity fucntion is natural from the viewpoint of reliability engineering. We can estimate parameters using maximization of likelihood method. Taking logarithm of the likelihood function, we can obtain

$$L(\lambda, p) = np \log \lambda + n \log p + (p-1) \sum_{i=1}^{n} t_i - \lambda^p T^p. \tag{7}$$

The maximum likelihood estimator of λ_0 and λ_1 are given by solving the following likelihood equation

$$\begin{aligned} (T\lambda)^p &= n \\ np \log \lambda + n + p \sum_{i=1}^{n} \log t_i &= n \log n. \end{aligned}$$

The parameter p and λ are given by

$$\begin{aligned} p &= \frac{n}{n\log T - \sum_{i=1}^{n}\log t_i}, \\ \lambda &= \frac{n^{1/p}}{T}. \end{aligned}$$

Note that in the stationary case where $\lambda(t) = \lambda$, the above procedure reduces to the estimation $\hat{\lambda} = n/T$.

Remark We can use other functions for $\lambda(t)$. Here we present two other possible models for $\lambda(t)$ One is linear models of the form

$$\lambda(t) = \lambda_0 + \lambda_1 t. \tag{8}$$

The other is exponential model of the form

$$\lambda(t) = \theta_1 e^{\theta_2 t}. \tag{9}$$

As with Weibull type model, we can estimate parameters using maximization of likelihood method. Taking logarithm of the likelihood function of linear model, we can obtain

$$L(\lambda_0, \lambda_1) = \sum_{i=1}^{n}\log(\lambda_0 + \lambda_1 t_i) - \lambda_0 T - \frac{\lambda_1 T^2}{2}. \tag{10}$$

The maximum likelihood estimator of λ_0 and λ_1 are given by solving the following likelihood equation

$$\begin{aligned} \sum_{i=1}^{n}\frac{1}{\lambda_1 t_i + \lambda_0} &= T \\ \sum_{i=1}^{n}\frac{t_i}{\lambda_1 t_i + \lambda_0} &= \frac{T^2}{2}. \end{aligned}$$

As for exponential model, we can obtain

$$\begin{aligned} \sum_{i=1}^{n} t_i + n\left(\frac{1}{\theta_2} - \frac{Te^{\theta_2 T}}{e^{\theta_2 T} - 1}\right) &= 0 \\ \theta_2 \frac{n}{e^{\theta_2 T} - 1} &= \theta_1. \end{aligned}$$

The shortcoming of these models is that the above equations is inherently nonlinear and we must handle them with the help of numerical solvers such as Newton-Raphson method. This is not preferable from the viewpoint of computational load for data analysis platform. In contrast, the intensity function 6 can give analytical solution of maximum likelihood estimator and therefore more efficient. Therefore, we use Weibull type model in the subsequent arguments.

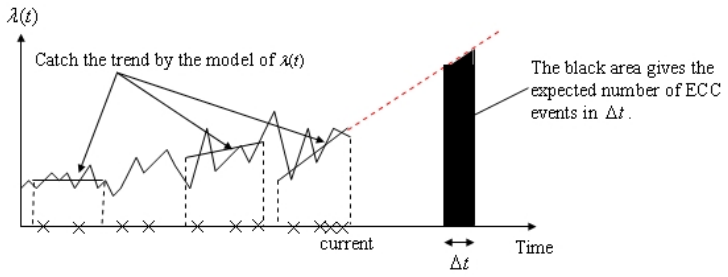

Figure 3: Prediction technique based on non-stationary Poisson process

The estimated $\lambda(t)$ can be used for failure prediction as shown in the graph. We extend the estimated lambda toward the future, and calculate the black area, which gives the expected number of events in Δt. Figure 3 shows our approach based on non-stationary Poisson process. The black area gives the expected number of ECC events which happen in time interval $(T, T + \Delta t)$. The black area grows as time T evolves into the future. When the black area exceeds ECC capability, we consider the system will be unreliable and generate alerts for potential failures.

The choice of parameter N relates to the precision of estimation. Since the maximum likelihood estimator $\hat{\theta}$ aymptotically approaches to normal distribution $N(\theta, N^{-1}I(\theta)^{-1})$, where $I(\theta)$ is a Fisher Information matrix, we can compute the necesarry number of data to achieve predefined accuracy. More important thing is that determination of parameters T. This parameter should be determined depending on the time scale of the target failure mode. For example, ECC events occur normally once in several hours. Therefore, T should be at least as long as 24 hours.

3.4 Utilization of side information

The intensity function $\lambda(t)$ can change due to fluctuation of memory access rate, which does not reflect real state of memory module. If memory access rate is available, we can adjust the $\lambda(t)$ to remove the factor of memory access rate.

4 Simulation results

To validate the effectiveness of our approach, we make a simulation using artificially generated point process data. The realization of non-stationary Poisson process with intensity function $\lambda(t)$ can be generated by time-scale change of stationary Poisson process as follows:

$$T = \Lambda^{-1}(U), \tag{11}$$

where U is a time-stamp of event occurence of stationary Poisson process with $E[U] = 1$ and Λ^{-1} is the inverse function of $\Lambda(t) = \int_0^t \lambda(s)ds$. This can be easily checked by the following argument:

$$\begin{aligned} P(N(u, u+\Delta u) = 1) &= P(N(t, t+\Delta t) = 1) \\ &= \int_t^{t+\Delta t} \lambda(s)ds \\ &= (u+\Delta u) - u \\ &= \Delta u. \end{aligned}$$

Using the above time-scale change, we can generate simulation data in the following steps.

1. Generate uniform random numbers on $[0, 1]$ as $\{u_1, \cdots, u_n\}$.
2. Transform $\{u_1, \cdots, u_n\}$ into the data $\{e_1, \cdots, e_n\}$ with exponential distribution Ex(1) by inverse function method.

$$\begin{aligned} F(x) &= \int_0^x e^{-x}dx \\ e_i &= F^{-1}(u_i) = -\log(1-u_i). \end{aligned}$$

3. Generate data for the stationary Poisson distribution $\{p_1, \cdots, p_n\}$ by

$$p_i = \sum_{n=1}^{i} e_i. \tag{12}$$

4. Transform $\{p_1, \cdots, p_n\}$ into non-stationary Poisson process data $\{t_1, \cdots, t_n\}$ by

$$\begin{aligned} t_i &= \Lambda^{-1}(p_i), \\ \Lambda(t) &= \int_0^t \lambda(t)dt. \end{aligned}$$

Figure 4 shows a typical transition of systems from normal state(flat part of $\lambda(t)$) and abnormal state (increasing part of $\lambda(t)$). The following data is a realization of timestamp data generated by Figure 4.
$\{1.31684, 3.19278, 5.18288, 6.01396, 6.1545, 8.04205, 8.29949, 8.81266,$
$9.83644, 12.1488, 12.1801, 12.573, 12.873, 13.1006, 13.6092, 15.7427,$
$17.1669, 18.2702, 19.3409, 19.583, 19.5871, 20.0274, 20.081, 20.2653, 20.6953\}$
The estimation result for the first 15 data is $\hat{p} = 1.40, \hat{\lambda} = 0.438$. On the other hand, the estimation for the data from 16th to 25th is $\hat{p} = 3.04, \hat{\lambda} = 0.425$. The value for *lambda* is almost the same for two estimations. However, the estimation for p is significantly different. This is due to the increase of intensity fucntion and the estimation can successfully cath the trend.

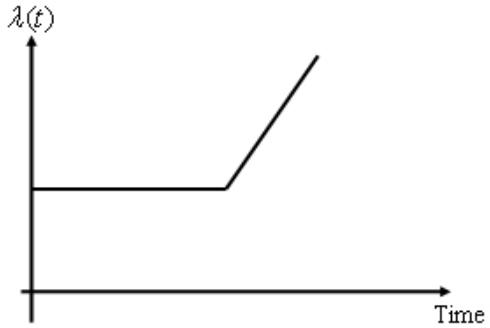

Figure 4: An artificial example

5 Overview of Failure Prediction System

Failure predcition system can be implemented in two different ways - distributed and centralized. In the first case, failure prediction system is implemented on each server. However, it is undesirable to put an additional software or overhead to managed node although it is expected to consume just a little computational resource like CPU cycles and memory. In such a case, the second implementation is useful, that is, the failure prediction system is implemented on a central management server and each managed node sends their events and status periodically. Figure 5 shows the overview of centralized failure prediction system. DAP (Data Analysis Platform) [4] is a runtime and framework for implementing such centralized online analysis. It is designed to achieve higher analysis throughput and scalability so that the central server can support large number of nodes with its limited hardware resource. Each node just send events and side information to the central management server, and the central management server performs the analysis for multiple managed nodes. Event analyzer performs online estimation of $\lambda(t)$ either on managed node or a central management server each time an event (execution of ECC) occurs to a managed node. If side information like CPU load and memory access rate can be considered to affect the occurrence rate of the events, event analyzer monitors the side information at regular interval. We set the sliding window of fixed length and we move it every time event analyzer receives a new event. The monitored side information is kept for a while (e.g. several minutes) for later use when an event occurs. The old side information can be discarded in sequence. If T can be fixed in advance, the failure probability calculator updates failure probability each time an event occurs to a node. If T cannot be fixed in advance e.g. job scheduler wants to query each time it starts a job whose expected execution time is T, the failure probability calculator calculates the probabiliby on receiving a request using the current estimation of $\lambda(t)$. Then, failure probability calculator

outputs probability of system failure in time t at node k based on the estimated $\lambda(t)$.

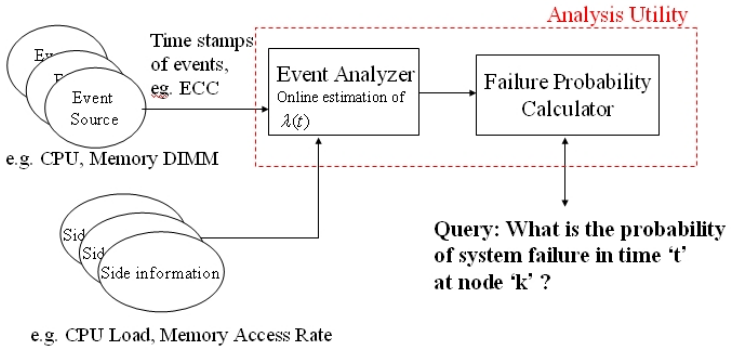

Figure 5: Overview of Failure Prediction System

6 Discussion

Generally speaking, computer systems is full of various kinds of log files in which the event descriptions with their timestamps are recorded. Our method, which is based on point process, is easily extended to handle these timestamped log file data. For the application of failure management, it is important to see the correlation among various events. Since the point process data is not collection of numerical data, which is treated in ordinary time series analysis, it is not proper to compute the correlation of two point process data in the same way as time series data. Instead, we can compute the event correlation as the correlation of their intensity functions, which are ordinary time-dependent data.

7 Conclusion and future works

We have presented a method of predicting failure events in computer systems before they become critical for system activity. Our method is based on model of error events as point process and hence has generality for other error events other than ECC. The application to real environment and data is left to future work.

References

[1] Linux EDAC Project, http://bluesmoke.sourceforge.net/

[2] First Failure Data Capture, United States Patent Application: 0040024726.

[3] IBM Netfinity Predictive Failure Analysis, http://john.ccac.rwth-aachen.de:8000/ftp/mirrors/ps2supersite.homedns.org/pccbbs/pc_servers/pfaf.pdf

[4] Hideki Tai and Takayuki Kushida, "Failure Management for Large Scaled Systems," IBM Research Report, 2007.

[5] A.J.Oliner, R.K.Sahoo, J.E.Moreira, M.Gupta, A. Sivasubramaniam, "Fault-aware Job Scheduling for BlueGene/L Systems", IPDPS2004.

[6] R.K.Sahoo, A.J.Oliner, I.Rish, M.Gupta, J.E.Moreira, S.Ma, "Critical Event Prediction for Proactive Management in Largescale Computer Clusters"

[7] R.Vilalta, C.V.Apte, J.L.Hellerstein, S.Ma, S.M.Weiss, "Predictive algorithms in the management of computer systems", IBM SYSTEMS JOURNAL, VOL 41, NO 3, pp.461-474, 2002.

[8] Sayantan Chakravorty Celso L. Mendes Laxmikant V. Kalé, "Proactive Fault Tolerance in Large Systems"

[9] J. M. Brandt, A. C. Gentile, D. J. Hale, and P. P. Pebay. "OVIS: A Tool for Intelligent, Real-time Monitoring of Computational Clusters," In Proceedings of 20th International Parallel and Distributed Processing Symposium, 2006.

[10] Ting-Ting Y. Lin, Daniel P. Siewiorek, "Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis, IEEE TRANSACTIONS ON RELIABILITY, VOL. 39, NO. 4, pp.419-432, 1990 OCTOBER.

[11] Yawei Li, Zhiling Lan, "Exploit Failure Prediction for Adaptive Fault-Tolerance in Cluster Computing," pp. 531-538, Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06), 2006.

[12] Greg Hamerly, Charles Elkan, "Bayesian approaches to failure prediction for disk drives"

[13] Takeuchi, K. Shimohigashi, K. Kozuka, H. Toyabe, T. Itoh, K. Kurosawa, H. "Origin and characteristics of alpha-particle-induced permanentjunction leakage" IEEE Transactions on Electron Devices, Volume: 37, Issue: 3, Part 1, pp. 730-736, Mar 1990.

[14] Y. Katayama, E. J. Stuckey, S. Morioka, and Z. Wu, "Fault-Tolerant Refresh Power Reduction of DRAMs for Quasi-Nonvolatile Data Retention," 1999 IEEE International Symposium on Design and Fault Tolerance in VLSI Systems.

[15] Bianca Schroeder Garth A. Gibson "Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?", FAST'07: 5th USENIX Conference on File and Storage Technologies.