PROCEEDINGS OF SCIENCE

Parameter Prediction in Fault Management Framework

Thanyalak Chalermarrewong[1]
Dept. of Computer Engineering,
King Mongkut's University of Technology Thonburi
Thailand
E-mail: thanyalak.cha@gmail.com

Tiranee Achalakul
Dept. of Computer Engineering,
King Mongkut's University of Technology Thonburi
Thailand
E-mail: tiranee@cpe.kmutt.ac.th

Simon Chong-Wee See
Dept. of Mechanical & Aerospace Engineering,
Nanyang Technological University, Singapore
Nvidia Corporation, Singapore
E-mail: mcwsee@ntu.edu.sg

Abstract— High performance computing systems can have high failure rates as they feature a large number of servers and components with intensive workload. The availability of the system can be easily compromised if the failure of these subsystems is not handled correctly. To ensure an availability of the computing resources, there is a need for an effective fault management framework. This research proposes a strategy to preserve system's availability focusing on a prediction model. An ARMA model is used to be a parameter prediction method of the framework. The main idea is to create an effective prediction model focusing on hardware failure. System parameters associated to hardware fault are input of our prediction model. This model uses prior data to predict future data. Each predicted parameter then will be used to predict availability of the system. Experiments show the effectiveness of this model and how to find appropriate interval of periodically gather data.

[1] Speaker

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

PROCEEDINGS OF SCIENCE

The International Symposium on Grids and Clouds (ISGC) 2012
Academia Sinica, Taipei, Taiwan
February 26 – March 2, 2012

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

http://pos.sissa.it

1. Introduction

Nowadays cloud computing has become popular because it has given many advantages such as high availability, energy saving, and on-demand services. The main idea of a cloud computing system is consolidating a large amount of resources and virtualizing them. Such facilities that house computer systems are often referred to as datacenters. Cloud-based datacenters, consisting of processors, memory units, disk drives, networking devices, and various types of sensors can support many applications and users. Any system running applications with such heterogeneous and intensive workload may sometimes be vulnerable to different types of failures. Among the failures, there are many sources of failure such as software, hardware, network, or environment. Hardware is the single largest cause of breakdowns [1]. Therefore, this research is designed to handle system failures which occur due to hardware faults.

Through reviewing the literature, there are many strategies currently employed to handle failures and maximize availability including replication of components and/or tasks, checkpointing and restarting, prediction, and migration techniques. Each strategy is suitable for some types of applications and systems. Some of these strategies handle failures by recovering them after the failures occur while others strategies prevent machine from failure. Checkpointing/restarting and replication may increase workload, capacity usage, power usage, and computational time. Checkpointing/restarting requires a large amount of extra disk spaces to store system states. The replication of tasks increases system workload and storage. Moreover, the system is rendered unavailable during recovery state and any change made after the restored point is lost. Therefore, we are interested in proactive strategies such as prediction and migration. Combining prediction and migration concepts can be an efficient way to emphasize system availability since the job running on the node predicted to fail can be migrated beforehand. The fault management framework is designed to migrate a job/process running on incoming failure machine to another healthy machine. Thus, the accuracy of the prediction model plays an important role on this strategy. We have to choose a proper prediction model which is appropriate for data characteristic.

In this paper, we propose a strategy for preserving system availability focusing on a prediction model. Our prediction model uses a failure indicator parameter to be the input. Other types of failure including software fault, environmental problem, and forced error will not be considered. Since each prediction model is suitable to each data characteristic and there are many models available for prediction such as classification and regression, we hypothesize that a regression model or time series model will suit the characteristic of parameters collected from such a system. The main goal is to develop a prediction model using time series model to predict future of system parameters with high accuracy for a fault management framework. The framework can migrate job on imminent failure node to others. In this paper we will show that our prediction model can accurately predict, so the framework can suggest the solution to handle failure on time. We assume that there is neither a second fault on the same node when being migrated nor that a second fault is dependent to the first one.

The rest of this paper is organized as follows: related work on fault management, including concept and many prediction models, are presented in Section 2. The design overview of the fault management framework is described in Section 3. Details of autoregressive moving average and the initial experiment will be shown in section 4 and 5, respectively. And the last section, section 6 is a conclusion.

2. Related Works

Fault tolerant techniques (FT) used in traditional distributed systems can be categorized into fault prevention, fault detection, fault isolation, fault identification, and fault recovery [2]. Fault isolation can be done in two different ways: proactive fault tolerance and reactive fault tolerance [3, 4]. The concept of proactive fault tolerance is avoiding system failures by studying pre-fault indicators and taking preemptive actions before failures occur. Reactive fault tolerance focuses on recovery of a system from unpredictable failures. Both proactive and reactive fault tolerance have their own advantages and disadvantages. Despite the performance drawback, reactive techniques are still used more often because they are less affected by incorrect predictions and also relatively simple to implement. However, for high availability clusters, reactive fault tolerance may not be appropriate. Once a failure occurs, availability decreases. Our framework employs the proactive concept of fault tolerance since it is more effective than its reactive counterpart in the perspective of green computing. In proactive FT, there is no checkpoint step, therefore memory usage, energy usage, cost and overhead will be less than that of reactive FT. Faults may still occur but since the tasks can migrate beforehand, the impact on the overall system will be minimal. However, accurate fault prediction is needed for efficient migration selection. From the previous literature, prediction models can be classified into 3 groups: those based on statistical models, on artificial intelligence computing techniques, and on control theory. Each prediction model is suitable for some certain characteristics of data and system environments

The first group is the statistical methods called the time series model [6-9] which analyzes preceding data to determine what will occur in the future. Those time series model (mentioned here) are Autoregressive (AR), moving average (MA), autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), and linear regression (LR). The accuracy of the results is depended on the characteristic of data. Artificial intelligence computing techniques tend to be more suitable to non-linear data but the regression model is simpler.

Another prediction method is Artificial Intelligent Computing. This method performs computational analysis based on empirically observed data [10-11]. SVM (Support Vector Machine) and ANN (Artificial Neural Network) are the most popular as they can provide good solution for short term forecasting of non-linear data. However, these two models can be stuck in local minima as its initial state is random and the model can be slow to converge.

The method based on control theory used in fault prediction is the Multi-Input-Multi-Output system or MIMO. MIMO uses the control feedback and admission controls concept [12]. This method yields a prediction with high accuracy and speed, but requires more stable

design and higher overhead. Feedback loop control is employed under proactive fault tolerance in [3], where a failure on computational nodes in HPC can be prevented.

In this work, the health of each node is monitored. Since the application deployment environment is based on cloud computing Infrastructure as a Service (IaaS), all virtual machines running on nodes facing imminent failure are reallocated or migrated to another node. The database is also added to store history of failure prediction to improve upon the model accuracy and identify long-running failure patterns. In this research, we design a proactive fault management framework based on the ARMA model, which will give high accuracy if the data can fit to a linear model. Moreover, a fault tree is also introduced as a part of a prediction model since it is generally used in reliability analysis.

3. The Design Overview

The key idea of our fault management framework is to accurately predict the future state of a system. Figure 1 which consists of 5 components shows the overall framework. The five components in the framework are monitoring module, pre-processing module, availability analysis module, resource manager module, and migration module. Modules are linked to the database in order to store and retrieve historical and predicted data.

The monitoring module captures a set of hardware parameters from each physical machine on the cloud computing system. We select parameters that experience has shown are reliable predictors of hardware failures. The parameters are observed periodically based on a suitable predefined interval. We observe data periodically by interval time. Interval time corresponds to the unit of time to capture data. Interval time can be empirically got. Section 5 shows how to select an appropriate interval for monitoring data. Once the data are captured, the pre-processing module then aggregates data from all nodes and transforms them into a uniform format. The formatted data is then stored in a database. This formatted data is called “real time data” and then queried by the availability analysis module and used to analyze the chances of a system failure in the future. ARMA time series and fault tree analysis are employed to predict failures.

ARMA time series model is used to formulate a system model and predict a future value of each system parameters. Results from the model are used to analyze availability by Fault Tree. Fault Tree is top down analysis structure and is a popular method used in reliability analysis field. For Fault Tree, we define “Unavailability state” as a root. Afterwards, we list all hardware failure events that can lead to system unavailability as nodes of Fault Tree. Hardware failure events are then used to construct a tree by split each event into sub events (its cause).

We first build the prediction model from the data training set. When real time data is observed by monitoring module, it is fed to predict by ARMA prediction model which is already formulated in the Availability Analyzing module. The output from ARMA is then fed into the fault tree and also store back into a database. Basic events or leaf nodes of fault tree analyze threshold value of collected parameters to calculate their state. A fault tree will predict whether there is any likely failure on the future state. If so, the model will predict the time that failures will occurs and send triggering signals to alert the next module, the resource management module. This prediction module can also be self adjusted to control the accuracy.

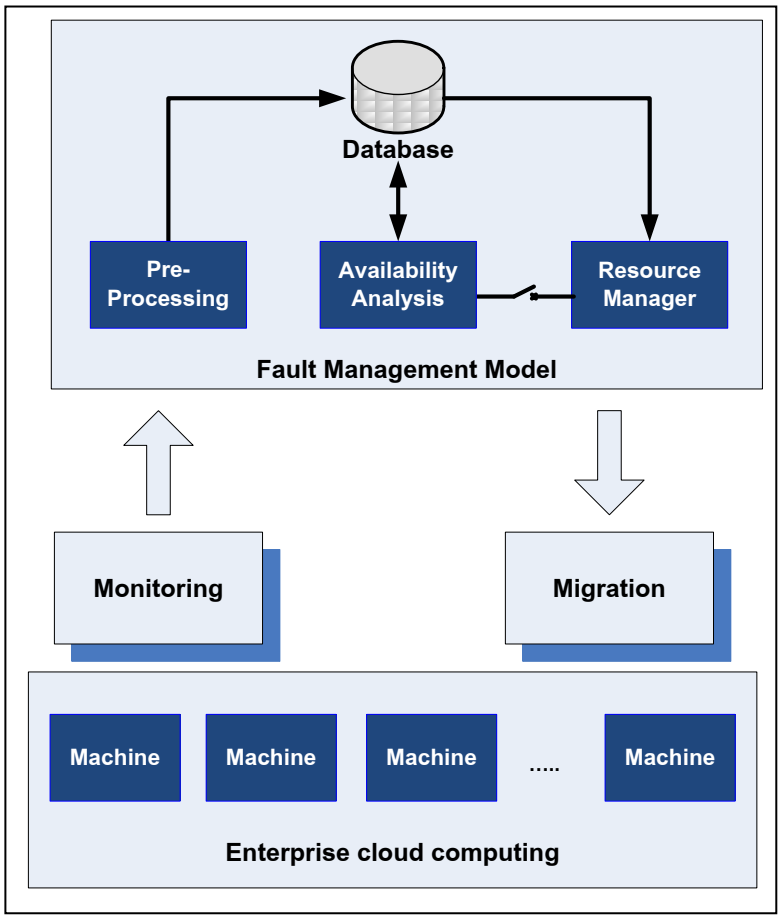

Figure 1. Fault Management Framework

We apply a fault tree to reduce the values of the predicted states for all system parameters into one state value for the root of the tree, representing system unavailability. We also use the fault tree to compute a probability of failure for each machine. The high probability of fault state identified risk state for machine(s), this value is used to selected destination node for migrated data. Since a signal is triggered to alert the resource management module, the resource manager can then perform a migration decision for the identified node(s). Once the triggering alert is received, an availability analysis module will halt a triggering signal from a same node until the migration process is finish. This can avoid repeat-migration on the same node. When several nodes are likely to fail at the same time, the migration sequence arranged based on the Time to Failure (TTF) values. The resource manager also selects a destination for each migration transaction. A compute node with low probability to failure is more likely to be target-node-chosen. Once a resource manager completes its decision, migration module can triggered the actual transfers.

In this paper, we focus on the ARMA prediction model which is in an availability analysis module. After building the ARMA model, real time data is used to predict the future value of each parameter that is used to analyze the chances of a system failure by next step that is fault tree. The details are described below.

4. Concept of Prediction with Autoregressive-Moving-Average-Model

The time series model, ARMA, is introduced to model each system parameter. We use the ARMA model to model the system because system parameters input have linear characteristic which is suited to model by time series. First of all, the ARMA model is built from training data collecting from target system. After building the model, it is then used to predict future values of each system parameter. As shown in Figure 2, real time data of each parameter is monitored and fed to ARMA model. Once ARMA model finishes predicting, the result is then fed to analyze availability of overall system.

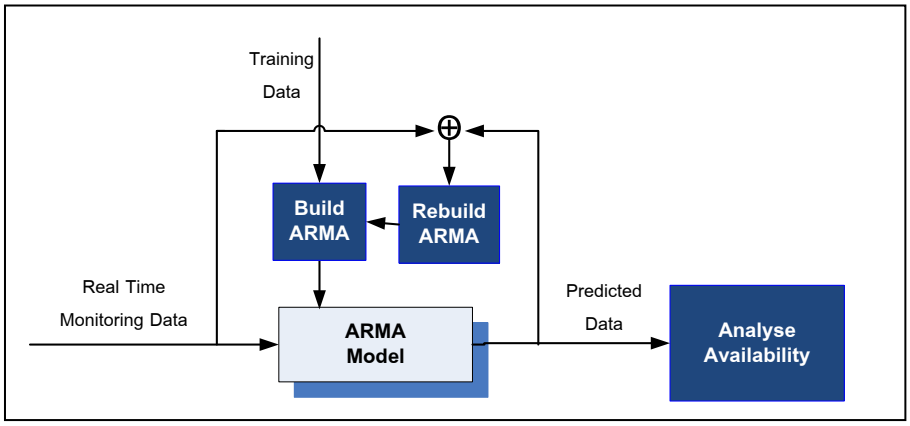

Figure 2. ARMA Prediction Model

To build ARMA model, we use the former states of data ass input. There are former values of parameter and former error. Number of former value of parameter used in model is called *p* and number of former error used in model is called *q*. Both *p* and *q* are obtained empirically. Details of each parameter of model is given in (1).

$$\bar{x}_{t+1} = \gamma + \sum_{i=0}^{p-1} \alpha_i x_{t-i} - \sum_{j=0}^{q-1} \beta_j e_{t-j} + \varepsilon \qquad (1)$$

Where x_{t-i} is an observed system parameter value at time, *t-i*, and e_{t-j} presents a series of error previously predicted, observed at time *t-j*. We use *p* former values of *x* and *q* former values of *error* to formulate the model. '*p*' represents a window size or order of autoregressive terms, while '*q'* represents the order of lagged forecast errors. The values of γ, α, β are coefficients obtained from regression using a set of historical data. Finally, $\bar{x}_{t+1}$ is a predicted system parameter value at future time. Once a future state of each parameter is predicted, it is stored in a database.

Because the ARMA model is built by a training data set, the model's accuracy depends on resemblance of predicting data and real time data. The predicted values are then fed into the Rebuild ARMA model to evaluate model's accuracy. Prediction accuracy of the ARMA model is evaluated periodically in order for the model to self-adjust. Equation (2) shows how to evaluate accuracy of model by comparing real time data and predicted data.

$$RMSE_n = \sqrt{\frac{\sum_{t=1}^{n}(X_{T-t} - \hat{X}_{T-t})^2}{n}} \qquad (2)$$

Where X_{T-t} shows real time system parameter on previous t interval, $\hat{X}_{T-t}$ indicates predicted of system parameter on previous t interval, and n is a number of period to evaluate model. When $RMSE_n$ value exceeds threshold in some period, ARMA model will be rebuilt using current observing data. So an accurate fault prediction can preserve availability of the observed system.

ARMA will be run every, $\triangle t$, time period. In other words, ARMA predicts whether a fault may occur in "$\triangle t$" unit time in the future. The value of "$\triangle t$" can be obtained empirically. If the value is too small, once a fault is predicted there might not be enough time to migrate VMs. If the value is too large, faults might occur undetected. Empirical results in the next section present experimentation on ARMA model and evaluation of accuracy of the model.

5. Initial Experiments

In this study, we monitored the performance parameters and recorded the entire anomaly that occurred on each machine in a simulated environment. These parameter values are then used to predict the future performance of the system. The details of the experiment are described below.

5.1 Interval of monitoring parameters

System parameters values are periodically captured on some pre-defined interval. The interval period may affect the accuracy of the prediction model. In our experiment, we empirically select the interval periods. Using a UNIX shell script, the values of system parameters are recorded at every one second, five seconds, fifteen seconds, twenty seconds, and thirty seconds.

Data from all interval periods are used as inputs to the prediction model. Mean absolute error (MAE) and Root Mean Square Error (RMSE) values from the prediction of '% CPU utilization" are then collected. MAE in equation (3) is used to measure the average magnitude of the errors without considering their direction. RMSE in equation (4) is used to measure the prediction error. $\hat{X}_t$ represents '% CPU utilization" predicted value and X_t is the actual measurement.

$$MAE = \frac{\sum_{t=1}^{N} \left| X_t - \hat{X}_t \right|}{N} \qquad (3)$$

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N} (X_t - \hat{X}_t)^2}{N}} \qquad (4)$$

Table 1 summarizes the MAE and RMSE of '%CPU load prediction' on each cluster. We collect data based on ARMA(1,1) model.

Table 1. MAE and RMSE on sets of interval observation

Interval(seconds)	MAE	RMSE
1	0.019	3.74E-04
5	3.50E-04	1.22E-07
15	7.45E-05	5.54E-09
20	8.66E-04	7.51E-07
30	8.25E-04	6.80E-07

Using the interval period of 15 seconds gives the best (lowest value) MAE and RMSE. In our experiment the monitored period is thus15 seconds.

5.2 Prediction with Autoregressive-Moving-Average-Model

In this initial experiment, „*% CPU utilization*" and „*CPU fan speed*" are used to evaluate the effectiveness of adopting the ARMA model (shown in equation 4). From equation (4), x_t is an observed system parameter value at current time *t*, and e_t presents an error previously observed at time *t*. While γ, α_1, β_1 are coefficients. We use one order of autoregressive terms of both observation and error term.

$$\bar{x}_{t+1} = \gamma + \alpha_1 x_t - \beta_1 e_t + \varepsilon \qquad (4)$$

Predictive results of '*% CPU utilization*' and '*CPU fan speed*' using ARMA(1,1) are shown on Figure 3 and Figure 4, respectively. Data of '*CPU fan speed*' tends to be larger and ARMA model can rightly capture this increasingly trend. *'% CPU utilization*', on the other hand, does not have a trend. Each data record is roughly the same. ARMA(1,1) also correctly predict the values. RMSE of '*% CPU utilization*' and '*CPU fan speed*' are 1.22 and 26.93 respectively.

Since RMSE shows acceptable accuracy for both parameters, ARMA(1,1) will be used in our framework to predict the parameter values. These values will then be used as input data of the Fault Tree. The fault Tree will calculate future unavailability state of the system and probability of unavailability of the system. The state predicted from Fault Tree model will indicate solution for availability.

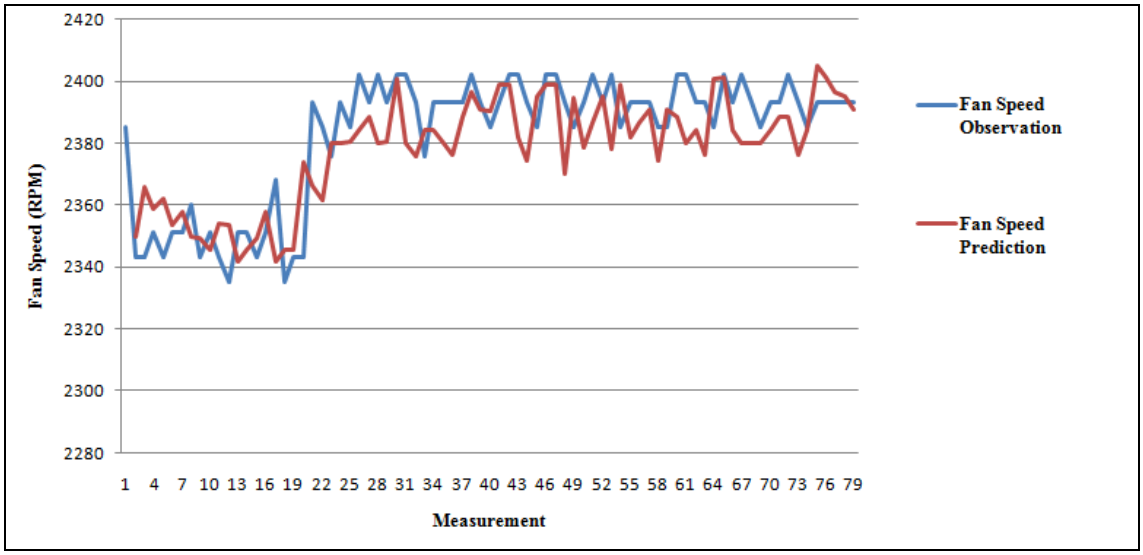

Figure 3. Comparison prediction and observation value of CPU fan speed

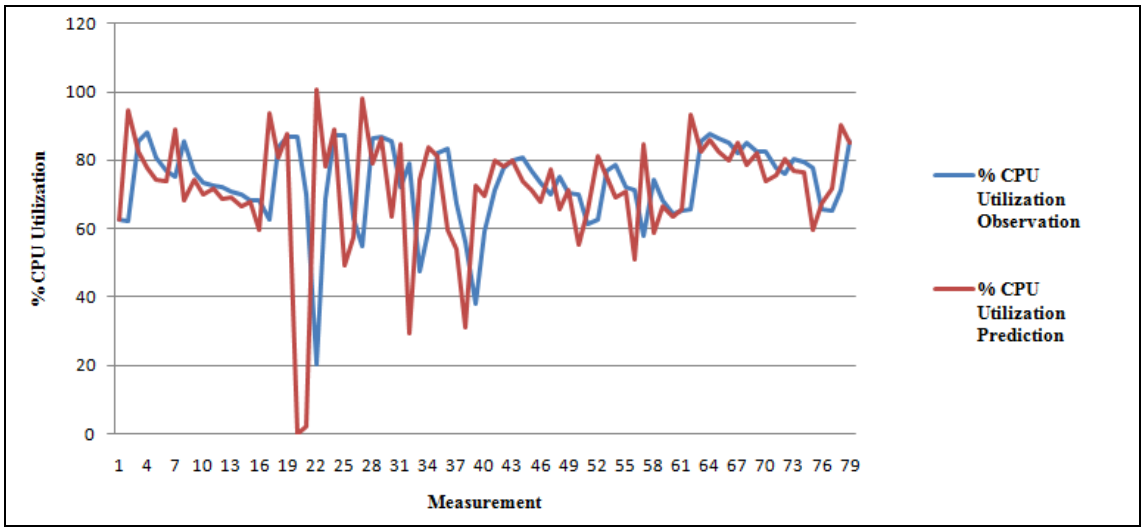

Figure 4. Comparison prediction and observation value of % CPU utilization

6.Conclusion

To obtain availability of the system, we employ ARMA as our prediction model on availability analysis of a fault management framework. This paper focuses on failure prediction on a datacenter with the emphasis on hardware faults handling. Different to other techniques, time series model such as ARMA is proper to normal characteristic of parameters gathered from datacenters. To monitor data, interval time to capture data is also important. We empiricism to select the interval that gives the best error. Once ARMA model is constructed, real time data that are captured on our selected interval can be continuously sent to be an input of model.

Predicted data from ARMA is sent to fault tree which is a strategy to analyze occurrence of system failures later. An accurate fault prediction can preserve system availability, a model can be dynamically adjusted by periodically evaluate accuracy. Empirical results show accuracy of model but ARMA can re-model if the model gives a bad result on an accuracy evaluation procedure in the future.

For future work, we plan to adapt ARMA model by introducing a Multi-Step-ARMA prediction model. Since ARMA can only predict for the next nearest step. If the chosen step interval is too small, then the projection to significantly larger step intervals is unreliable. Conversely, if too large a step interval is chosen, then there is the possibility that system failures may occur during the step interval, that will result in system unavailability. We will use many models in many data time steps to predict future parameter. Using adaptive prediction model, we can predict both nearby data and remote data.

References

[1] B. Schroeder and G. A. Gibson, *Understanding failures in petascale computers*, Journal of Physics: Conference Series, Volume 78, No. 1, 2007.

[2] A.S. Tanenbaum and M.V. Steen, *Distributed Systems: Principles and Paradigms*, Prentice Hall, pp. 393- 400, 2002.

[3] C. Engelmann, G.R. Vall´ee, T. Naughton, and S.L. Scott, *Proactive Fault Tolerance Using Preemptive Migration*, Proc. of the International Conference on Parallel, Distributed, and Network-based Processing, Weimar, Germany, 2009.

[4] A. Litvinova, C. Engelmann, and S.L. Scott, *A Proactive Fault Tolerance Framework for High-Performance Computing*, Proc. of the 9th IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN), Innsbruck, Austria, February 16-18, ACTA Press, Calgary, AB, Canada, 2010.

[5] V. Gianuzzi, *Data Replication Effectiveness in Mobile AdHoc Networks*, PE-WASUN'04, pp. 17-22, 2004.

[6] P.A. Dinda, D. O'Hallaron, *An Evaluation of Linear Models for Host Load Prediction*, Proc. the 8th IEEE International Symposium on High-Performance Distributed Computing (HPDC-8), pp.10, Redondo Beach, CA, 1999.

[7] S. Casolari, M. Colajanni, *Short-term Prediction Models for Server Management in Internet-based Contexts*, Journal ACM Transactions on the Decision Support Systems (DSS), Volume 48 Issue 3, No.1, Jul. 2008.

[8] P. Lall, P. Gupta, M. Kulkarni, and J. Hofmeister, *Time–Frequency and Autoregressive Techniques for Prognostication of Shock-Impact Reliability of Implantable Biological Electronic Systems*, IEEE Transactions on Electronics Packaging Manufacturing, Volume. 33, Issue 4, pp. 289 – 302, 2010.

[9] S. Fu and C.Z. Xu., 2007, *Exploring Event Correlation for Failure Prediction in Coalitions of Clusters*, Proc. of ACM/IEEE Supercomputing Conference (SC).

[10] Ying-feng Zhang Biao, Ma Jin-song, Zhao Hai-ling Zhang, 2010, *Fault prediction of Power-Shift Steering Transmission Based on Support Vector Regression*, IEEE International Conference on Information and Automation (ICIA), Pages 273 - 277.

[11] G. Hu, L. Hu, H. Li, K. Li, W. Liu, 2010, *Grid Resources Prediction with Support Vector Regression and Particle Swarm Optimization*, Third International Joint Conference on Computational Science and Optimization (CSO), Volume1, pp.417-422.

[12] S.M. Park, M.A. Humphrey, 2011, *Predictable High-Performance Computing Using Feedback Control and Admission Control*, IEEE Transactions on Parallel and Distributed Systems, Volume. 22, No. 3, pp. 396-411.