A Survey on the Application of Machine Learning Algorithms to Predict Software Aging

Monisha M, Samyukta Sherugar, Shobhit Bansal, Rajat Mann & Biju R Mohan

Department of Information Technology, National Institute of Technology Karnataka, Surathkal, India
E-mail : monisha3107@gmail.com, samyuktasherugar@gmail.com, shobhiteng.955@gmail.com, rajatmann100@gmail.com, bijurmohan@gmail.com

***Abstract* – Software aging is considered one of the major causes of software outages. Thus, being able to predict their occurrence is of vital importance. In this paper we have presented the results of surveying 10 papers which relate to the field of applying machine learning techniques for prediction of software aging. These papers date between 2010 and 2012. Our survey compares the papers based on the machine learning algorithms studied, metrics monitored, tools used, platform used for implementation, results obtained and the result validation done. This survey will provide researchers with the latest trends and findings of this field.**

Keywords – Classifiers, Machine learning, Prediction, Software aging

I. INTRODUCTION

With the increasing complexity of software systems there has been an increase in the difficulty of managing these systems; there been a rising number of failures resulting in undesired behaviours. Due to our high dependence on software systems it is important to reduce these failures. An important area that has been the topic of much research in this context is the prediction of software aging.

A. Software aging and rejuvenation

Studies [1, 2] have reported that the software aging phenomenon is one of the causes of software failures. Aging is the accumulation of errors occurring in long-running operational software systems that leads to progressive resource depletion, performance degradation, and eventually to the hang or crash of the software system [3]. In order to mitigate the effects of software aging, the concept of software rejuvenation was proposed in [4] to delay or prevent the occurrence of failures. The software rejuvenation approaches can be categorized as: Time-based and inspection-based strategies. In time-based strategies, rejuvenation is applied regularly and at predetermined time intervals. Inspection-based rejuvenation measures the progress of the aging effects and, when it crosses a certain pre-specified limit, triggers the chosen rejuvenation mechanism. One of the inspection-based methods is prediction-based approaches. Some are applied to predict the time to the exhaustion of resources while others predict the time to failure caused by the software aging. Then, the rejuvenation algorithm is triggered based on the predicted time to exhaustion [5]. The Different prediction methods in use: machine learning, statistical approaches, or structural models [6 - 9].

B. Machine learning

Machine Learning methods have found numerous applications in performance modeling and evaluation. One of the applications is defect diagnosis which is based on predicting the state (normal or fault state) of a particular system by monitoring various observations of a system. Machine learning techniques are preferred in situations where manually created models cannot effectively handle the complexity of the problem. One of the major problems for applying ML algorithms in fault or aging prediction is the unavailability and scarcity of data for training the model. Most of the companies that own (or control) projects do not share their failure due to propriety interests and national security concerns [14].

Since 1995, year of publication of the seminal work of Huang et al. [4], several efforts have been directed towards characterizing and mitigating the Software Aging phenomenon. The developments in the initial years have been the subject of much research and survey. In our study we focus on the most recent developments. We have investigated 10 software aging

papers published between year 2010 and 2012. These papers were the relevant matches filtered from search results that were generated for the key words "software aging", "performance degradation", "machine learning" entered in plausible combinations. We have compared them based on a set of parameters namely machine learning algorithms studied, metrics monitored, tools used, platform used for implementation, results obtained and the result validation done. Papers chosen for this survey reflect the ideas and methods that have emerged in recent times, but however we cannot call this an exhaustive review of all the papers. The inclusion of papers to this study was based on the degree of similarity to each other and their relevance to the area of prediction of software aging.

C. Significant researchers

It is important to know of the people who have contributed to this field and so we have presented the active researchers; Javier Alonso has done significant work in the field of reliability, availability and performance modeling of computer and communication systems alongside several other prominent researchers. Kishor S. Trivedi, Grottke and V. Castelli are prominent researchers in the field of software aging and rejuvenation. The seminal work by Huang et al. [4], in the field of software aging and rejuvenation, is heavily referenced. The works of several of the above mentioned authors have been included in our study. The others are major contributors to this field and have been referenced in a number of papers that were part of our survey.

This paper is organized as follows: section II discusses our observations, section III concludes our paper and section IV presents a proposal for future work.

II. SURVEY OBSERVATIONS

We surveyed 10 papers in our study and the following section summarizes the observations we have made.

Aging related studies stress the system with workloads and record system data. This data is fed to the machine learning algorithms to predict the performance degradation of the system. Most of these studies run web-server-performance benchmarks (TPC-W, Httperf) on an application server (Apache Tomcat) for data collection. The metrics monitored are usually system metrics like traffic metrics, load metrics and application and server metrics.

We observed that most studies select several machine learning algorithms and compare the error rates associated with the results provided by each. They then select the one that gives the best results. Some of the prominently used algorithms are decision trees, neural networks, SVM and Naïve Bayes. Some studies use readily available implementations of the algorithms in tools like WEKA, BNJ, etc. A few others have implemented their own version of the algorithms. The results were better wherever preprocessing strategies (like feature selection, filtering, bagging, clustering and bootstrapping) were used. In several papers, the training and testing was done using an n-fold cross validation technique to ensure reduced error in the results. Performance evaluators were used to gauge the accuracy of the model. These included precision, recall, mean absolute error and ROC curve.

Alonso, Torres, Berral and Gavalda [9] evaluated a machine learning prediction algorithm to predict time until failure due to both dynamic and non-deterministic software aging. The model was tested using TPC-W, a benchmark application which simulates the activities of a bookstore web application on a three-tier web J2EE application server (Apache Tomcat). The algorithm used was M5P (based on binary decision tree) classifier implemented in the tool WEKA. In order to achieve a more accurate prediction a set of derived metrics like consumption speed from every resource under monitoring was added to the dataset. A sliding window average over the last X speed observations from the resource was used to help smooth out the noise and fluctuation. There were 29 attributes in each record of the dataset corresponding to the various system metrics recorded. The total number of training and testing data instances used for the experiments are as follows: Deterministic Software aging: 2776; Dynamic and Variable; Software Aging: 1710; Software Aging Hidden within Periodic Pattern resource Behaviour: 1710; Dynamic Software Aging due to two resources: 2752. The prediction accuracy was measured using the Mean Absolute Error (MAE) and Soft Mean Absolute Error (S-MAE). The model was made more accurate during a crash by calculating the MAE for the last 10 minutes of every experiment (POST-MAE) and for the rest of experiment (PRE-MAE). If the approach had lower MAE in the last 10 minutes than the rest of experiment it showed that prediction became more accurate when it was more needed. It was observed that the model was achieving acceptable prediction accuracy against complex scenarios with small training data sets.

Zhao, Trivedi, Wang and Chen [15] proposed a BP neural network model (SPE) for the evaluation of software performance affected by aging using httperf on Apache 1.3 web server and a client connected via an Ethernet local area network. The model was built with 3-13 nodes allotted to the hidden layer and the sigmoid function adopted as the transfer function. SPE involves

the collection of experimental data on line, followed by feature selection. Then, each collected feature data set is normalized, after which the combined normalized data sets are processed by the SPE model. The data set consisted of 200 instances with 8 attributes (each) corresponding to the following system metrics: Reply rate, response time, no. of timeout errors, time out error rate, CPU usage, free swap space, free physical memory and average load. The SPE model was simulated in Matlab. MSE (Mean Square Error) was calculated to find the optimum number of nodes in the hidden layer of the network. The concept of the inflexion point was presented. Based on evaluation results of the SPE model, a robust locally weighted regression algorithm was presented to determine the inflexion points of the data set.

Alonso, Belanche and Avresky [16] analyzed a set of ML algorithms for predicting system crashes due to the resource exhaustion caused by software anomalies. They evaluated the following classifiers: Rpart (Decision trees), Naive Bayes, Support Vector Machines Classifiers (SVM-C), K-nearest neighbors, Random Forest and LDA/QDA implemented in the R Statistical Language tool. Three different execution scenarios were created, each having a different software anomaly that would cause a system crash by running TPC-W benchmark on an application server (Apache Tomcat). The training data sets were based on 2815 instances, 1688 instances and 3819 instances in Scenario 1, 2 and 3 respectively with 29 attributes in each. When the ML algorithm had parameters, different sets of values were used to create different configurations of the same algorithm (leading to a comparison of over 35 models). The error to compare the models was calculated using 5-fold cross-validation (CV) approach. Random Forest obtained a much better error (less than 1%) than the rest of the models in all three scenarios and Naive Bayes gave the highest errors. Lasso regularization was used to reduce (up to 60%) the number of parameters needed to build the Random Forest Model which in turn reduced the error in several cases. This was followed by analyzing the trade-off between the number of monitored parameters and accuracy.

Poggi1, Carrera1, Gavald`a1 and Ayguad´e1 [17] introduced a novel technique using Machine Learning to predict the expected sales volume over time and look for deviations over the expected values during overload periods for an online retailer in order to reduce performance degradation. This approach was tested on logs from a top Online Travel Agency (OTA), using 3+ year long sales dataset, HTTP access log, and resource consumption logs for several weeks of 2010. The forecasting of sales was done by providing training (162,438 instances) and testing (8,151 instances) datasets with 7 attributes to the created prototype which is based on WEKA tool. Pre-processing steps included quantification of response time effects using performance logs which were produced using probes in PHP application, calculation of conversion rates as a function of response time, and validating workload results using Apache logs and monitoring system access. The relative absolute error for the numerical classifiers used is as follows: Linear Regression - 67.73%, M5P - 23.998%, REPTree - 21.64%, Bagging (M5P) - 23.2% and Bagging - (REPTree) 19.42%. The performance evaluators used were correlation coefficient, mean absolute error, root mean squared error, relative absolute error and root relative squared error. Results showed that the inflection points during which sales start to drop for different applications was when the response time is high. For the OTA, it was found that there is a tolerating response time threshold from 7 to 10 seconds, where some sales are lost, and a frustration threshold at 10 seconds, where each increase in 1 second interval increases total sale loss by 6%.

Magalhaes and Silva [18] developed a framework for detection of performance anomalies in web and component-based applications. The work was carried out in two steps. The first step involved collecting and conducting correlation analysis on historical data and then feeding it to machine learning classifiers in WEKA namely: naïve Bayes, two decision trees (J48 and LMT) and a neural network model (MLP). In the second step, synthetic aging scenarios such as memory leaks and CPU contention were induced in the application and the systems parameters were estimated using time series analysis (ARIMA and Holt-Winters). The experimental setup consisted of an application server (Glassfish) running TCP-W. Dataset A (Customer Registration) consisted of 5898 instances and the Dataset B (Home Transaction) had 6790. The parameters estimated from the datasets are submitted to the three classifiers that verify if such a combination may result in absence (”GREEN”), symptom (”YELLOW”) or strong indication (”RED”) of a performance anomaly. It was observed that both the ML algorithms have high precision and accuracy. The results obtained were preliminary and concluded that a single machine learning algorithm is not sufficient to predict the user-transactions response time and that a combination of time-series models should be considered for parameters estimation, to cope with the changes that may occur in both time and load.

Ohta and Hirota [19] proposed a scheme based on a machine learning technique to estimate the number of computers that should be running in a server cluster in order for the cluster to offer sufficiently good performance against changes in load. Httperf was run on

a power management system for a small cluster was implemented on a PC with a Linux OS to verify the feasibility and effectiveness of the proposed approach through experiments executed for the static and the dynamic load case. The functions of the system were measurement of the load metrics (CPU utilizations, disk and network interfaces, byte rate, packet rate, TCP connection establishment rate, TCP SYN loss rate) using captured packets during request generation period, feeding the constructed data set (241 data instances with 4 attributes) from the metrics values to the c4.5 decision tree classifier [20], counting the number of computers to be turned on or off and accordingly modify the on/off status. Two performance parameters: the average TCP connection establishment time and the average bit rate from the server to the client are estimated. In case of the static load the reduction was more noticeable for the light load, where only a small number of computers were needed. It was observed that the power consumed with the use of the proposed method decreased to 54% of that consumed without the method for light loads. Where as, the effectiveness of the proposed method was not so evident for the heavy load. For the dynamic load it was noted that the bit rate was greater than 10 Mb/s in most time periods which indicated sufficiently good performance. However, the bit rate was slightly lesser than 10 Mb/s in the period where the load reached the maximum. It was understood that the reason for the criterion violation was not the incorrectness of the classifier but because the number of server computers was set at its maximum during this period. The other criterion for the performance was the connection establishment time which was always less than 0.1 s. From the above result, it was concluded that the proposed method accurately estimates the number of computers required for a sufficiently good performance and successfully reduces the power consumption of server computers.

Hayashi and Ohta [21] proposed a method to detect performance degradation by passively measuring the traffic exchanged by virtual machines. Multiple traffic metrics including bit-rate, packet rate, connection rate, TCP SYN loss rate and number of flows were monitored and estimated every 60 seconds by a program that was built using the C language and the pcap [22] library which used the captured packets as the input. The mapping from these metrics to performance states was understood to be a complex function that had to be determined from the measured sample data. Apache Web server was run on the virtual machines built on the Xen hypervisor. Httperf, a web benchmark was run on the client machines used to measure the request-response rate of files being transferred between the virtual machines and clients. This data along with the recorded traffic metrics formed the training (600 instances) and test data (1000 instances) having 5 attributes for the c4.5 [20] machine learning classifier. The algorithm constructed a decision tree that detected the performance from traffic metrics. The error ratio was found to be as small as 2.2% for 1000 test data instances. Thus, degradation detection by the proposed method was considered reliable in most cases.

Rughetti, Di Sanzo, Ciciani and Quaglia [23] utilized a machine learning approach to predict the performance of a Software-Transactional-Memory (STM) system as a function of the number of concurrent threads in order to dynamically select the optimal concurrency level during the whole lifetime of the application. In this method, the STM is coupled with a neural network and an on-line control algorithm that activates or deactivates application threads in order to maximize performance via the selection of the most adequate concurrency level, as a function of the current data access profile. System metrics like concurrent threads, average wasted execution time and workload profile values were used in this study. Filtering out samples stemming from sampling intervals in which more than 99% of the transactions have been aborted was done in order to improve the efficiency. The final data set contained 800 instances each having 8 attributes. An experimental study using TinySTM, an open-source package and the STAMP benchmark suite was conducted. The experimental data confirmed the proposed scheme constantly provided optimal performance, thus avoiding performance loss phases caused by unsuited selection of the amount of concurrent threads.

Dwyer, Fedorova, Blagodurov, Roth, Gaud and Pei [24] proposed a practical method for estimating performance degradation on multicore processors and its application to HPC workloads by building a model for modeling performance degradation on multicore systems using machine learning. The process of building the model consisted of three steps: (1) collection of the training data, (2) attribute selection, (3) model training. To confirm the portability of the model, it was built and tested on two systems, Intel and AMD using exactly the same procedure. Before building the model the number of attributes in the dataset (500 co-schedules) was reduced from 340 to 19 to eliminate the attributes that were redundant or unrelated to degradation, to reduce the training time and to allow a new dataset to be recorded with fewer events sets, leading to more accurate recording. The machine learning classifiers (in WEKA) used in this study include: REPTree , Linear Regression, Gaussian Process, PLS Classifier, Decision Table , Isotonic Regression, Simple Linear Regression, SVM Reg, Neural Network, SMO Reg, Conjunctive Rule, M5P, Decision Stump, Pace Regression and

M5Rules. The accuracy of the model was evaluated using cross validation and calculating the error rate. It was observed that the accuracy increased when bagging technique was used. In addition, a confidence predictor was proposed that successfully anticipates when the model is likely to produce an inaccurate estimate and reduces the maximum error by a factor of three.

Galar, Kumar and Fuqing [25] proposed a novel RUL (Remaining Useful Life) prediction method that addresses multiple challenges in complex system prognostics, where many parameters are unknown and is inspired by feature maps and SVM classifiers. Conditional Maintenance Data for bearings in a complex mechanical system were used to create a classification by SVM hyper planes. System metrics corresponding to time domain features (9 of them) were studied. A Multi-class SVM by combining different SVM binaries was implemented and used for the prediction Time to Failure (TTF) or Remaining Useful Life (RUL).

A small set of features were selected and fused in a self-organizing feature map (SOFM) where the current status of the analyzed element will be represented according to the selected parameters. For a test instance of the same system, whose RUL was estimated, degradation speed was evaluated by computing the minimal distance defined based on the degradation trajectories, i.e. the approach of the system to the hyper plane that segregates good and bad condition data at a different time horizon. Therefore, the final RUL of a specific component was estimated, and global RUL information was obtained by aggregating the multiple RUL estimations using a density estimation method.

In Table I we have presented the highlights of the papers included in this study. Column I contains the data preprocessing strategies followed as well any techniques that are applied to improve accuracy. Column II contains the results and conclusions of each paper.

Table I : Highlights of the surveyed papers
which relate to the field of applying machine learning

Paper No.	Preprocessing strategies and Additional techniques used to improve accuracy	Results and Conclusion
1	Adding a set of derived metrics calculated using the sliding window average of the resources	The model achieved acceptable prediction accuracy against complex scenarios with small training data sets.
2	Calculated MSE to find the optimum number of nodes in the hidden layer of the network	A robust locally weighted regression algorithm is employed to identify the inflection point of some threshold range
3	Lasso regularization was used in order to reduce the number of parameters needed to build the Random Forest Model	Random Forest obtained a much better error(less than 1%) than the rest of the models and Naïve Bayes gave the highest errors.
4	Quantification of response time effects using performance logs; produced using probes in PHP application, Calculation of conversion rates as a function of response time, Validating workload results using Apache logs and monitoring system access	Relative absolute error: Linear Regression – 67.73%, M5P – 23.998%, REPTree – 21.64%, Bagging (M5P) 23.2% and Bagging (REPTree) 19.42%
5	Monitoring module (AOP – AspectJ) aggregates collected data into time intervals, Performance Analyser measures the correlation between the response time and no. of transactions processed using Pearson's correlation.	Response time is finally classified into: (”GREEN”), symptom (”YELLOW”) or strong indication (”RED”) of a performance anomaly. It was concluded that a single machine learning algorithm is not sufficient to predict the user-transactions response time.
6	Load metrics computed using captured packets during request generation period	Static Load (Light) : 54% reduction of power consumed, Dynamic Load: Bit rate was 10Mb/s in most time periods – good performance, Connection establishment time was always less than 0.1 s
7	Metrics estimated from captured packets using C and the pcap library	Error ratio: 2.2% for 1000 test data instances, Proposed method was considered reliable in most cases

Paper No.	Preprocessing strategies and Additional techniques used to improve accuracy	Results and Conclusion
8	Filtered out samples stemming from sampling intervals in which more than 99% of the transactions have been aborted	Proposed Self-adjusting STM system which constantly avoided performance loss phases caused by unsuited selection of the no. of concurrent threads
9	Attribute selection, Bagging	Best performance – Bagged REPTree. Proposed confidence predictor that successfully anticipates when the model is likely to produce an inaccurate estimate and reduces the maximum error by a factor of three
10	Selected features fused in Self-Organized Feature Maps (SOFM)	Effective RUL prediction method that addresses multiple challenges in complex system prognostics

III. CONCLUSION

This study surveyed papers belonging to the discipline techniques for prediction of software aging. We observed that machine learning techniques proved to be effective and efficient. However, the general sentiment was that it is not possible to choose any one classifier as the best performer across all datasets; the results are expected to vary with the nature of the dataset. Further, results can be improved by using data preprocessing techniques.

Our interests lie in the area of predicting performance degradation in a virtual machine monitor (VMware). As a result of this survey we have a better insight into using machine learning techniques for this purpose.

IV. REFERENCES

[1] K. S. Trivedi, K. Vaidyanathan and K. Goseva-Popstojanova. "Modeling and Analysis of Software aging and Rejuvenation". IEEE Annual Simulation Symposium, April 2000

[2] T. Dohi, K. Goseva-Popstojanova and K. S. Trivedi. "Analysis of Software Cost Models with Rejuvenation." IEEE IntI. Symposium on High Assurance Systems Engineering, 2000

[3] Cotroneo, D.; Natella, R.; Pietrantuono, R.; Russo, S.; , "Software Aging and Rejuvenation: Where We Are and Where We Are Going," Software Aging and Rejuvenation (WoSAR), 2011 IEEE Third International Workshop on , vol., no., pp.1-6, Nov. 29 2011-Dec. 2 2011

[4] Y. Huang, C. Kintala, N. Kolettis, N. Fulton, "Software rejuvenation: analysis, module and applications", in: The 15th International Symposium on Fault- Tolerant Computing, FTCS '95, 1995, pp. 381–390.

[5] J. Alonso, R. Matias, E. Vicente, A. Maria, K.S. Trivedi, "A comparative experimental study of software rejuvenation overhead", Performance Evaluation, 20 September 2012

[6] R. Matias, P.J.F. Filho, "An experimental study on software aging and rejuvenation in web servers", in: Proceedings of the 30th Annual International Computer software and Applications Conference - Vol. 01, IEEE Computer Society, Washington, DC, USA, 2006, pp. 189–196.

[7] A. Andrzejak, L. Silva, "Using machine learning for non-intrusive modeling and prediction of software aging", in: IEEE/IFIP Network Operations & Management Symposium, NOMS 2008, 2008, pp. 7–11.

[8] K. Vaidyanathan, R.E. Harper, S.W. Hunter, K.S. Trivedi, "Analysis and implementation of software rejuvenation in cluster systems", in: Proceedings of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '01, ACM, New York, NY, USA, 2001, pp. 62–71.

[9] Alonso, J.; Torres, J.; Berral, J.L.; Gavalda, R.; , "Adaptive on-line software aging prediction based on machine learning," Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on , vol., no., pp.507-516, June 28 2010-July 1 2010

[10] B. Boehm and P. Papaccio, "Understanding and controlling software costs," IEEE Transactions on Software Engineering, vol. 14, no. 10, pp. 1462–1477, 1988.

[11] M. Harrold, “Testing: a roadmap,” in Proceedings of the conference on the future of software engineering, 2000, pp. 61–72.

[12] S. Sherer, “Software fault prediction,” Journal of Systems and Software, vol. 29, no. 2, pp. 97–105, 1995.

[13] Cagatay Catal, Ugur Sevim, Banu Diri, “Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm", Expert Systems with Applications, Volume 38, Issue 3, Pages 2347-2353, March 2011

[14] Twala, B.; , "Software faults prediction using multiple classifiers," Computer Research and Development (ICCRD), 2011 3rd International Conference on , vol.4, no., pp.504-510, 11-13 March 2011

[15] Jing Zhao; Trivedi, K.S.; Yanbin Wang; XiaoYong Chen; , "Evaluation of software performance affected by aging," Software Aging and Rejuvenation (WoSAR), 2010 IEEE Second International Workshop on , vol., no., pp.1-6, 2-2 Nov. 2010

[16] Alonso, J.; Belanche, L.; Avresky, D.R.;, "Predicting Software Anomalies Using Machine Learning Techniques," Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on , vol., no., pp.163-170, 25-27 Aug. 2011

[17] Poggi, N.; Carrera, D.; Gavalda, R.; Ayguade, E.; , "Non-intrusive Estimation of QoS Degradation Impact on E-Commerce User Satisfaction," Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on , vol., no., pp.179-186, 25-27 Aug. 2011

[18] Magalha~es, J.P.; Silva, L.M.; , "Prediction of performance anomalies in web-applications based-on software aging scenarios," Software Aging and Rejuvenation (WoSAR), 2010 IEEE Second International Workshop on , vol., no., pp.1-7, 2-2 Nov. 2010

[19] Ohta, S.; Hirota, T.; , "Machine Learning Approach to the Power Management of Server Clusters," Computer and Information Technology (CIT), 2011 IEEE 11th International Conference on , vol., no., pp.571-578, Aug. 31 2011-Sept. 2 2011

[20] J. R. Quinlan, C4.5: Programs for machine learning, Morgan Kaufmann, 1993

[21] Hayashi, T.; Ohta, S.; , "Performance Management of Virtual Machines via Passive Measurement and Machine Learning," Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2012 9th International Conference on , vol., no., pp.533-538, 4-7 Sept. 2012

[22] TCPDUMP/LIBPCAP Repository, Available: http://www.tcpdump.org/

[23] Rughetti, D.; Di Sanzo, P.; Ciciani, B.; Quaglia, F.; , "Machine Learning-Based Self-Adjusting Concurrency in Software Transactional Memory Systems," Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on , vol., no., pp.278-285, 7-9 Aug. 2012

[24] Tyler Dwyer, Alexandra Fedorova, Sergey Blagodurov, Mark Roth, Fabien Gaud, Jian Pei, “A Practical Method for Estimating Performance Degradation on Multicore Processors and its Application to HPC Workloads”, SC '12 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Article No. 83, Nov. 2012

[25] Galar, D.; Kumar, U.; Yuan Fuqing; , "RUL prediction using moving trajectories between SVM hyper planes," Reliability and Maintainability Symposium (RAMS), 2012 Proceedings - Annual , vol., no., pp.1-6, 23-26 Jan. 2012

◈◈◈