A Failure Detection and Prediction Mechanism for Enhancing Dependability of Data Centers

Qiang Guan, Ziming Zhang, and Song Fu

***Abstract*—Modern data centers continue to grow in their scale and complexity. They are changing dynamically as well due to the addition and removal of system components, changing execution environments, frequent updates and upgrades, online repairs and more. Classical reliability theory and conventional methods do rarely consider the actual state of a system and are therefore not capable to reflect the dynamics of runtime systems and failure processes. In this paper, we present an unsupervised failure detection and prediction method using an ensemble of Bayesian models. It characterizes normal execution states of the system and detects anomalous behaviors. We implement a prototype of our failure detection and prediction mechanism and evaluate its performance on a data center test platform. Experimental results show that our proposed method can forecast failure dynamics with high accuracy.**

***Index Terms*—Data centers, failure detection, failure management, dependable computing.**

I. Introduction

With ever-growing complexity and dynamicity of modern data centers, proactive failure management is an effective approach to enhance system dependability [1]. Failure prediction is the key to such techniques. It forecasts future failure occurrences in data centers using runtime execution states of the system and the history information of observed failures. It provides valuable information for resource allocation, computation reconfiguration and system maintenance [2]. In contrast to classical reliability methods, failure prediction is based on runtime monitoring and a variety of models and methods that use the current state of a system and the past experience as well.

Most of the existing failure prediction methods are based on statistical learning techniques [3]. They use supervised learning models to approximate the dependency of failure occurrences on various performance features [4], [1]. The underlying assumption of those methods is that the training dataset is labeled, i.e. for each data point used to train a predictor, the designer knows if it is corresponded to a normal execution state or a failure. However, the labeled data are not always available in real-world data center systems, especially for newly deployed or managed systems. How to accurately forecast failure occurrences in such systems is challenging.

In this paper, we propose a failure detection and prediction mechanism that uses Bayesian models to forecast failure dynamics in data centers. We tackle the problem from an anomaly detection viewpoint, for which we introduce an ensemble of Bayesian models. It works in an unsupervised learning manner and deals with unlabeled datasets. This model estimates the probability distribution of runtime performance data collected by health monitoring tools when servers perform normally.

The rest of this paper is organized as follows. Section 2 discusses the related works. Section 3 describes our failure detection and prediction mechanism. Conclusion is presented in Section 4.

Manuscript received June 19, 2012; revised August 5, 2012. This research was supported in part by U.S. NSF grant CNS-0915396 and LANL grant IAS-1103.

Q. Guan, Z. Zhang, and S. Fu are with the Department of Computer Science and Engineering, University of North Texas, Denton, Texas 76203 USA (e-mail: QiangGuan@my.unt.edu; ZimingZhang@my.unt.edu; Song.Fu@unt.edu, Tel.: +1-940-565-2341; fax: +1-940-565-2799).

II. Related Work

Failure and anomaly detection based on analysis of system logs has been the topic of a number of research articles. Hodge and Austin [5] provide an extensive survey of anomaly detection techniques developed in machine learning and statistical domains. A structured and broad overview of extensive research on anomaly detection techniques has been presented in [6]. Hellerstein et al. [7] developed a method to discover patterns such as message burst, periodicity and dependencies from SNMP data in an enterprise network. Yamanishi et al. [8] modeled syslog sequences as a mixture of Hidden Markov Models to find messages that are likely to be related to critical failures. Lim et al. [9] analyzed a large-scale enterprise telephony system log with multiple heuristic filters to search for messages related to failures. However, treating a log as a single time series does not perform well in large-scale computer systems with multiple independent processes that generate interleaved logs. The model becomes overly complex and parameters are hard to tune with interleaved logs [8]. Qiang et al. [10], [11] explored health data groups rather than a time series of individual data in anomaly detection.

Failure management is a crucial technique for understanding emergent, system-wide phenomena and self-managing resource burdens for system-level dependability and productivity assurance. The conventional method for failure management and fault tolerance relies on checkpointing/restart mechanisms, which periodically save a snapshot of a system to a stable storage and use it to recover the system from failures reactively; see [12] for a comprehensive review. However, checkpointing a job in a large-scale system could incur significant overhead. The LANL study [13] estimates the checkpointing overhead based on the current techniques to run a 100 hour job (without failure) by an additional 151 hours in petaflop systems. As a result, frequent periodic checkpointing often proves counter-effective.

As the scale and complexity of production systems continue to grow, research on system dependability has recently shifted onto failure prediction and proactive management technologies [14], [15], [16], [17], [18], [19], [20]. Recent studies [21], [22], [23], [24], [25] apply execution migration techniques to enhance resource management by avoiding possible failures. They demonstrate the feasibility of exploiting proactive management methods for dependability assurance in networked computer systems. In this work, we exploit learning technologies to characterize system behaviors and propose an adaptive approach to forecasting failure occurrences in data center environments.

To realize proactive failure management, it is imperative to understand the characteristics of failure behaviors. Research in [26], [27], [28] studied event traces collected from clusters and supercomputers. They found that failures are common in large-scale systems and their occurrences are quite dynamic, displaying uneven distributions in both time and space domains. There exist the time-of-day and day-of-week patterns in long time spans [26], [28]. In addition, there has been prior work on monitoring and predicting failures for specific components in computer systems. Storage is one such subsystem which has received considerable attention because of its higher failure rates. S.M.A.R.T. is a recent technology, that disk drive manufacturers now provide, to help predict failures of storage devices [29]. SIGuardian [30] and Data Lifeguard [31] are utilities to check and monitor the state of a hard drive, and predict the next failure, to take proactive remedies before the failure. More recently, a Reliability Odometer [32] has been proposed for processors to track their wear-and-tear and predict lifetimes.

III. Failure Detection and Prediction Mechanisms

A recent system reliability study on a 512-node LLNL ASC White machine showed that the mean time to failure of a node was about 160 days [33]. We may label the runtime health related data with one of two classes, Class 0 for normal behavior and Class 1 for situations with failures. Then, Class 1 is very rare compared with Class 0. For Class 1, there many not be enough data available to allow a supervised learning algorithm to estimate a good probability model for that class. In addition, data from the rare class may be incomplete because of some collection problems. This is especially true when a node suddenly crashes which leaves no time for the monitoring tools to retrieve and save its performance data.

An alternative to supervised learning that tackles the unbalanced dataset is to build a probabilistic model of the majority class and use failure detection methods to cluster and characterize health-related data. Failure detection algorithms classify data as normal or not based on a probability model of normal behavior. A failure is a data point to which the majority class model assigns a very low probability of occurring. A failure detection algorithm can build a probability model and learn its parameters in (1) an unsupervised manner in which both normal and failure data are learned (This approach assumes the failure data are too rare to affect the model parameters significantly.) or (2) a semi-supervised manner in which only data of normal behavior are learned to construct a model. Semi-supervised learning is in general preferable because it may generate a more accurate model for the normal class. However, in real-world cases most collected data are not labeled and the failure class is a rare one. As a result, the unsupervised learning approach is more practical and useful.

A. Ensemble of Bayesian Models for Failure Detection

In order to detect possible failures, we analyze the health-related data and construct statistical models. A probabilistic model f is chosen for the data with reduced dimensionality. It takes a data point d as input and outputs a probability for that data point. The parameters of f are learned from training data in an unsupervised manner. Then d is detected as an failure if and only if $f(d) < t$, where t is a threshold, whose value can be determined based on assumptions of the rarity of failure data or from learning experiments if failures are labeled in the collected health-related data. A data point is labeled as normal or failure based on its probability of appearance as a normal data point.

To construct the probabilistic model and assure high detection accuracy, we develop an ensemble of Bayesian submodels to represent a multimodal probability distribution. Each submodel is a nonparametric data model, in which no single simple parameterized probability density is assumed. Its probability distribution is determined from the frequency counts of the training data. Each submodel has an estimated prior probability *p(m)*, where m is the submodel index. The probability estimate assigned to a data point d is

$$p(d) = \sum_{m \in sub\,\mathrm{mod}els} p(d \mid m) p(m), \tag{1}$$

where *p(d|m)* is the probability that submodel m generates data point *d*. Therefore, all submodels contribute to the probability of each data point. Also, a data point d is assigned to a submodel *m* with probability *p(m|d)*, whose value can be determined by using the Bayes' rule. This approach allows our model to fit the collected data better when it is unknown which submodel should be used to characterize the probability of a data point. After relevance reduction and redundancy reduction during feature selection, the features of a data point are conditionally independent in each submodel. Therefore, for a data point *d* with *k* features after dimensionality reduction, the conditional probability of the data point on a submodel *m* follows

$$p(d) = \prod_{i=1}^{k} p(d_i \mid m), \tag{2}$$

We use discrete Bayesian submodels, where the values of each feature are placed in a finite number of intervals. The discrete indexes of intervals replace the original value of a feature. We adopt the Bayesian expectation-maximization (EM) algorithm to estimate the probability that a feature d_i takes a given value *v* based on the counted frequency in the training dataset.

$$p(d_i = v \mid m) = count(d_i = v \; and \; m) / count(m) , \quad (3)$$

$$p(m) = count(m) \; / \sum_{m \in sub\,\mathrm{mod}\,els} count(n) , \quad (4)$$

where *count(·)* is the number of data points in the training dataset that satisfy a specified condition. In Equations (3) and (4), *count()* is calculated as

$$count(d_i = v \; and \; m) = \sum_{d \in training\,set} p(m \mid d) \cdot I(d_i, v) , \quad (5)$$

$$count(m) = \sum_{d \in training\ set} p(m \mid d) , \quad (6)$$

$$I(d_i, v) = \begin{cases} 1, & if\ d_i = v \\ 0, & if\ d_i \neq v \end{cases} , \quad (7)$$

To train the ensemble model, we choose the number of submodels. We initialize the model by assigning data points randomly to submodels. The Expectation-Maximization algorithm is performed to determine the submodel and conditional data probabilities, *p(m)* and *p(d|m)* respectively. Then, Equation (1) is applied to calculate the data probability. The EM algorithm proceeds in rounds of an expectation step (E-step) followed by a maximization step (Mstep). For a data point *d*, the E-step calculates the probability of each submodel *m* generating *d*. In the calculation, we use the Bayes' rule, *p(m|d) = p(d|m)p(m)/p(d)*, where the right-hand side is computed by Equations (1) and (2). After an E-step completes, an M-step updates probabilities *p(d$_i$|m)* and *p(m)* by Equations (3) and (4). It maximizes the likelihood of the model given the expected probability. The E-step and M-step continue until the likelihood does not change.

B. Decision Tree for Failure Prediction

The failure detection method based on an ensemble of Bayesian models presented in the preceding section identifies anomalous behaviors in a data center. The anomalies are reported to the system administrations for verification. If they are confirmed as failures or as normal, the corresponding data points are labeled. As the data center continues operation, more data points will be labeled. These labeled data provide valuable information about the system states under failures. They should be exploited in failure prediction. In this section, we present a method using decision trees for failure prediction.

A decision tree is a hierarchical nonparametric model with local regions identified in a sequence of recursive splits [34]. A decision tree is composed of internal decision nodes and terminal leaves. Each decision node n implements a test function *f$_n$(d)* with discrete outcomes labeling the branches. When a test hits a leaf node, the classification labeled on the leaf is output. There are many possible machine learning approaches for failure prediction. While decision trees are not always the most competitive classifiers in terms of prediction, they enjoy the crucial advantages of a fast localization of the region covering an input and yielding human interpretable results, which is important if the method is to be adopted by real data center operators.

Learning a decision tree involves deciding which split to make at each node, and how deep the tree should be. Let *X* denote the feature set. For binary classification, the class label is in {0, 1}, where 1 denotes a failure and 0 normal. The root node of the decision tree contains all of the health-related performance data. At each node, the dataset is split according to the values of one particular feature. Splits are selected in order to maximize the gain in information. This process continues until no further split is possible or the node contains only one class. After the tree is built, sub-branches with low overall gain value are pruned to avoid over-fitting.

The goodness of a split is measured by *impurity*. A split is pure if after the split, for all branches, all the data taking a branch belong to the same class. We use entropy to quantify impurity. For a node n in the decision tree, the *entropy* is calculated by

$$H(n) = -\sum_{i=1}^{j} p_n^i \log_2 p_n^i , \quad (8)$$

where *j* is the number of classes and *j* = 2 representing the *normal* and *failure* classes, and p^i_n is the probability of class C_i (0 or 1), given that a data point reaches node *n*. Then the gain function *G* for feature x_i at node *n* is defined as

$$G(x_i, n) = H(n) - H(x_i, n) , \quad (9)$$

where *H(x$_i$, n)* denotes the sum of entropy of children nodes after making the split based on feature x_i.

The algorithm for building a decision tree works as follows. It begins with the root node which includes all the features. For each feature x_i, the gain value from splitting on x_i is calculated using Equation (9). Then, the feature x_{best} with the highest gain value is selected. A decision node that splits on x_{best} is created. Repeat the preceding process on the sublists obtained by splitting on x_{best} and add those nodes as children nodes.

We grow the decision tree full until all leaf nodes are pure and we have zero training error. The tree might be too deep and complex. We then find subtrees that cause over-fitting and we prune them. From the initial labeled health-related data, we set aside a *pruning dataset*, unused in training. For each subtree, we replace it by a leaf node labeled with the training data points covered by the subtree. If the leaf node does not perform worse than the subtree on the pruning set, we prune the subtree and keep the leaf node because the additional complexity of the subtrees is not necessary; otherwise, we keep the original subtree.

After the decision tree is built and pruned, we exploit it for failure prediction. For a new and unlabeled data point, the failure predictor traverses the decision tree from the root. At each internal decision node, the predictor reads the value of the feature associated with the node from the input data point and selects a path to a child node accordingly. This process is repeated until a leaf node is hit. The predictor outputs the label (normal or failure) of the leaf. To achieve high prediction accuracy, we apply boosting to the decision tree classifiers and obtain an ensemble of trees. A voting is performed and the majority is selected as a failure prediction decision.

IV. Conclusion

Large-scale and complex data centers are susceptible to software and hardware failures and administrators' mistakes, which significantly affect the system performance and management. In this paper, we present a failure detection and prediction mechanism. At the initial stage of health monitoring and control, no labeled data are available. We propose to use an ensemble of Bayesian models to characterize normal states of the system and to detect anomalous behaviors in an unsupervised learning manner, and use decision trees to predict failures by utilizing newly labeled verification data. We implement a prototype of our failure detection mechanism and test its performance in a data center test platform. Experimental results show that our proposed method can forecast failure dynamics with high accuracy.

Acknowledgment

We would like to thank the anonymous reviewers for their constructive comments and suggestions. This research was supported in part by U.S. NSF grant CNS-0915396 and LANL grant IAS-1103.

References

[1] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam, "Critical event prediction for proactive management in large-scale computer clusters," In *Proceedings of ACM International Conference on Knowledge Discovery and Data Dining (KDD),* 2003.

[2] A. J. Oliner, R. K. Sahoo, J. E. Moreira, M. Gupta, and A. Sivasubramaniam, "Fault-aware job scheduling for BlueGene/L systems," In *Proceedings of IEEE/ACM International Parallel and Distributed Processing Symposium (IPDPS),* 2004.

[3] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Computing Surveys*, vol. 42, no. 10, pp. 1–42, 2010.

[4] J. W. Mickens and B. D. Noble, "Exploiting availability prediction in distributed systems," In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI),* 2006.

[5] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review,* vol. 22, pp. 85–126, 2004.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 15, pp. 1–58, 2009.

[7] S. Ma and J. L. Hellerstein, "Mining partially periodic event patterns with unknown periods," In *Proceedings of IEEE Conference on Data Engineering (ICDE),* 2001.

[8] K. Yamanishi and Y. Maruyama, "Dynamic syslog mining for network failure monitoring," In *Proceedings of ACM Conference on Knowledge Discovery in Data Mining (KDD),* 2005.

[9] C. Lim, N. Singh, and S. Yajnik, "A log mining approach to failure analysis of enterprise telephony systems," In Proceedings of *IEEE Conference on Dependable Systems and Networks (DSN)*, 2008.

[10] Q. Guan and S. Fu, "auto-AID: A data mining framework for autonomic anomaly identification in networked computer systems," In *Proceedings of IEEE International Performance Computing and Communications Conference (IPCCC),* 2010.

[11] Q. Guan, D. Smith, and S. Fu, "Anomaly detection in large-scale coalition clusters for dependability assurance," In *Proceedings of IEEE International Conference on High Performance Computing (HiPC),* 2010.

[12] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Computing Surveys*, vol. 34, no. 3, pp. 375–408, 2002.

[13] I. Philp, "Software failures and the road to a petaflop machine," In *Proceedings of Symposium on High Performance Computer Architecture Workshop*, 2005.

[14] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, and R. K. Sahoo, "BlueGene/L failure analysis and prediction models," In *Proceedings of IEEE Conference on Dependable Systems and Networks (DSN),* 2006.

[15] S. Fu and C. Xu, "Exploring event correlation for failure prediction in coalitions of clusters," In *Proceedings of ACM/IEEE Supercomputing Conference (SC)*, 2007.

[16] S. Fu and C. Xu, "Quantifying temporal and spatial correlation of failure events for proactive management," In *Proceedings of IEEE International Symposium on Reliable Distributed Systems (SRDS),* 2007.

[17] Z. Zhang and S. Fu, "Failure prediction for autonomic management of networked computer systems with availability assurance," In *Proceedings of IEEE Workshop on Dependable Parallel, Distributed and Network-Centric Systems in Conjunction with IEEE International Parallel and Distributed Processing Symposium (IPDPS),* 2010.

[18] Z. Zhang and S. Fu, "A hierarchical failure management framework for dependability assurance in compute clusters," *International Journal of Computational Science*, vol. 4, no. 4, pp. 313–326, 2010.

[19] S. S. Gokhale and K. S. Trivedi, "Analytical models for architecture-based software reliability prediction: A unification framework," *IEEE Transactions on Reliability*, vol. 55, no. 4, pp. 578–590, 2006.

[20] S. Fu and C. Xu, "Quantifying event correlations for proactive failure management in networked computing systems," *Journal of Parallel and Distributed Computing*, vol. 70, no. 11, pp.1100–1109, 2010.

[21] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott, "Proactive fault tolerance for HPC with Xen virtualization," In *Proceedings of ACM International Conference on Supercomputing (ICS),* 2007.

[22] S. Fu, "Failure-aware resource management for high-availability computing clusters with distributed virtual machines," *Journal of Parallel and Distributed Computing*, vol. 70, no. 4, pp. 384–393, 2010.

[23] S. Fu and C. Xu, "Proactive resource management for failure resilient high performance computing clusters," In *Proceedings of IEEE International Conference on Availability, Reliability and Security (ARES)*, 2009.

[24] C. Wang, F. Mueller, C. Engelmann, and S. L. Scott, "Proactive process-level live migration in HPC environments," In *Proceedings of ACM/IEEE Conference on Supercomputing (SC),* 2008.

[25] S. Fu, "Failure-aware construction and reconfiguration of distributed virtual machines for high availability computing," In *Proceedings of IEEE/ACM Symposium on Cluster Computing and the Grid (CCGrid)*, 2009.

[26] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance-computing systems," In *Proceedings of International Conference on Dependable Systems and Networks (DSN)*, 2006.

[27] Y. Liang, Y. Zhang, A. Sivasubramaniam, R. Sahoo, J. Moreira, and M. Gupta, "Filtering failure logs for a BlueGene/L prototype," In *Proceedings of Conference on Dependable Systems and Networks (DSN),* 2005.

[28] R. K. Sahoo, A. Sivasubramaniam, M. S. Squillante, and Y. Zhang, "Failure data analysis of a large-scale heterogeneous server environment," In *Proceedings of IEEE International Conference on Dependable Systems and Networks (DSN)*, 2004.

[29] G. Hughes, J. Murray, K. Kreutz-Delgado, and C. Elkan, "Improved disk-drive failure warnings," *IEEE Transactions on Reliability,* vol. 51, no. 3, pp. 350–357, 2002.

[30] Siguardian. [Online]. Available: http://www.siguardian.com/.

[31] Data lifeguard. [Online]. Available: http://www.wdc.com/wdproducts/library/other/2779-001005.pdf.

[32] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "A reliability odometer - lemon check your processor," In *Proceedings of ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS),* 2004.

[33] H. Song, C. Leangsuksun, and R. Nassar, "Availability modelling and analysis on high performance cluster computing systems," In Proceedings of *IEEE International Conference on Availability, Reliability and Security (ARES),* 2006.

[34] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth and Brooks, 1984.

Qiang Guan is currently a Ph.D. candidate in Computer Science and Engineering at the University of North Texas. He received the BS degree in Communication Engineering from Northeastern University, China, in 2005, and the MS degree in Information Engineering from Myongji University, South Korea, in 2008. He was a Ph.D. student in Computer Science at Mexico Institute of Mining and Technology from January 2010 to July 2010. His research interests include failure modeling and management, dependable assurance, resource management, and virtual machines in distributed and cloud computing systems.

Ziming Zhang is currently a Ph.D. candidate in Computer Science and Engineering at the University of North Texas. He received the BS degree in Computer Science from Beihang University, China, in 2009. He was a Ph.D. student in Computer Science at Mexico Institute of Mining and Technology from August 2009 to July 2010. His research interests include energy efficiency, power profiling, power-aware resource management, dependable computing, and virtual machines in distributed and cloud computing systems.

Song Fu is currently an Assistant Professor in the Department of Computer Science and Engineering and the Director of the Dependable Computing Systems Laboratory at the University of North Texas. He was an Assistant Professor in Computer Science and Engineering at New Mexico Institute of Mining and Technology from August 2008 to July 2010. He received his Ph.D. degree in Computer Engineering from Wayne State University in 2008, M.S. degree in Computer Science from Nanjing University, China, in 2002, and B.S. degree in Computer Science from Nanjing University of Aeronautics and Astronautics, China, in 1999. His research interests include distributed, parallel and cloud systems, particularly in dependable computing, self-managing and reconfigurable systems, power management, energy-efficient systems, system reliability and security, resource management, and virtualization. His research projects have been sponsored by the U.S. National Science Foundation, Amazon, Los Alamos National Laboratory, Xilinx Inc., and the University of North Texas. He is a member of the IEEE and a member of the ACM.