A Fault Diagnosis and Prognosis Service for TeraGrid Clusters

Zhiling Lan, Prashasta Gujrati, Yawei Li, Ziming Zheng
Rajeev Thakur and John White

Abstract—In this paper, we present an ongoing research effort on developing an automatic fault diagnosis and prognosis service for large-scale computing systems, such as TeraGrid clusters. By leveraging the research on system health monitoring, the proposed service aims at automatically revealing fault patterns from historical data by applying data mining and machine learning techniques. To address key challenges posted by fault diagnosis and prognosis, two integrated techniques are developed: a knowledge base to accumulate empirical and inferred fault patterns from historical data and a meta-learning mechanism to optimally combine separately learned classifiers for improved detection and prediction accuracy. We also present preliminary studies on failure logs from the BlueGene/L systems at SDSC and ANL.

Index Terms— Fault diagnosis, Fault Prognosis, Meta-learning, Knowledge base, Clusters

1 Introduction

In the field of high performance computing (HPC), one of the challenges facing the operation of large-scale computational infrastructure – such as TeraGrid clusters - is to understand and deal with failures. This involves not only examining the operational data found in logs, but also diagnosing trends and predicting reliability. Over the past decades, various technologies have been developed for improving fault resilience of applications and systems, such as failure-aware resource management and scheduling [22,35], checkpointing [2,7,9,12,28], and run-time resilience support [3,4,19]. The problem is that the fault management research has been hindered by the lack of automatic fault diagnosis and prognosis support in HPC. For example, many fault tolerance techniques require certain support of detecting and predicting potential failures to enable safer and more cost-effective fault handling operations. Even for reactive fault tolerance mechanisms such as checkpointing, an efficient fault prediction mechanism would substantially reduce their operational cost by telling when and where to perform fault tolerance actions, rather than blindly invoking actions periodically with an unwisely chosen frequency.

While an automatic fault diagnosis and prognosis support is clearly desirable for advancing fault management in HPC, it faces several key challenges. First, faults are very complicated phenomena, especially in large-scale HPC systems with tens-of-thousands to hundreds-of-thousands of components. A required service must be capable of capturing a variety of fault patterns and interactions in the system, even when these patterns are dynamically changing with time. Second, fault prediction subjects to some degree of uncertainty in practice, where unexpected failures may occur (i.e. false negative) and false alarms may be produced (i.e. false positive). False alarms may not be an issue in the situations where fault tolerance actions have relatively low cost if performed unnecessarily [6]. However, they can cause significant performance degradation in HPC [19]. An effective prediction service for HPC must be capable of minimizing both false positive and false negative rates.

This paper presents an on-going research project on developing an automatic fault diagnosis and prognosis service for large-scale computing systems, such as TeraGrid clusters. Specifically, we address the above challenges with two integrated approaches:

- Design a knowledge base for accumulating empirical and inferred knowledge of fault patterns and characteristics from historical data;
- Develop a meta-learning mechanism to optimally combine separately learned information for improved detection and prediction accuracy.

The next section discusses related work. Section 3 describes our design. Section 4 gives our preliminary studies on BlueGene/L RAS. Finally, we conclude the paper and discuss our future work in Section 5.

2 Related work

In HPC, many health monitoring tools have been developed for tracking the status of the underlying system and its components. From the hardware side, modern systems are deployed with various features (e.g. hardware sensors) that can monitor the degradation of an attribute over time for early detection of errors [1,14,15]. From the software side, a number of monitoring tools have been developed, including both publicly available packages and commercial tools [5,10,27,29]. Some leading supercomputers are even equipped with extensive error logging facilities, such as the CMCS service (Core Monitoring and Control System) in BlueGene/L [21]. In general, these tools can periodically collect performance- or health-related data from the underlying system by using different low-overhead tracking mechanisms. The collected data is usually stored in central data repositories or as system logs.

There are a number of research projects, from both aca-

- *Zhiling Lan, Prashasta Gujrati, Ziming Zheng are with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616. Email: {lan,gujrpra,liyawei,zzheng11}@iit.edu*
- *Rajeev Thakur is with Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439. Email: thajur@mcs.anl.gov*
- *John White is with San Diego Supercomputer Center, La Jolla, CA 92093. Email: whitej@sdsc.edu*

demia and industry, on the design of efficient fault diagnostic and prognostic techniques [17,25,26,31,32]. Existing predictive techniques can be broadly classified as either model-based methods or data mining based methods. The majority of research has focused on Internet services or business applications [6,25]. HPC applications significantly differ from business applications by using different programming and parallel paradigms, thereby resulting in distinct fault patterns and characteristics. To the best of our knowledge, there is no such a service in HPC that can comprehensively learn a variety of fault patterns from historical data and then use the learned knowledge to enhance fault management of applications and environments.

3 SERVICE DESIGN

Figure 1 presents the structure of our proposed knowledge-based fault diagnosis and prognosis service. The goal is to automate the process of analyzing massive quantities of historical data (e.g. those collected by health monitoring tools and stored in central data repositories or as system logs) for understanding fault characteristics in HPC. Such a service can be used by existing fault tolerance tools to enhance fault resilience of HPC applications or by system administrators to reduce management cost of HPC systems. For instance, it can be integrated with fault-aware scheduling [22,35] to improve system resilience. It can also be utilized by checkpointing libraries [2,9,28] or adaptive fault management [19] to improve fault resilience of HPC applications. The service consists of four major components, which are detailed in the following.

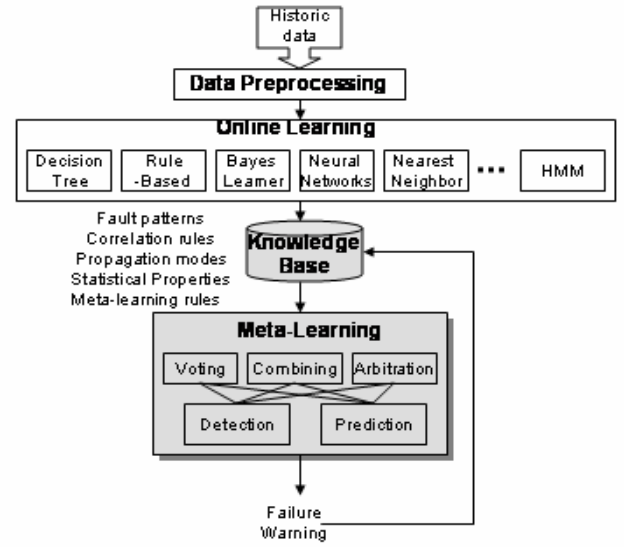

Figure 1. Overview of Knowledge-based Fault Diagnosis and Prognosis Service

The first component is *data preprocessing*. The data collected by various health monitoring tools are generally in arbitrary formats, while different data mining methods require different input formats. Hence, before applying any learning algorithm, data preprocessing is required. Necessary data preprocessing includes removing duplicated or unnecessary attributes, adding or removing data entries, creating consistency across similar entries based on a standard model, integrating separate logs or data sets from different components, and preparing desired input formats for different learning algorithms. It provides interfaces to various health-related data repositories and enables data translation for online learning.

The second component is *online learning*. In a large-scale system comprising hundreds-of-thousands components, causes of abnormal behaviors may be directly observable or be inferred from other normal behaviors that are observed during system operation. This component is responsible of dynamically learning fault characteristics across the system by applying an extensive set of data mining and machine learning techniques, such as those in Weka and R [24,33]. There are a great number of well-proven learning techniques, such as decision tree, Bayesian networks, neural networks, association rules, hidden Markov Model, etc. These learning techniques are separately explored to capture cause-and-effect relations from the preprocessed data. The learned patterns and relations are collected and stored in a knowledge base described below.

The third component is *a knowledge base*. Research studies have shown that health data is highly valuable for understanding failure modes and possible predicting potential failures in HPC. Traditionally, human operators are responsible to check the collected data for possible problems. Manual processing is time consuming, error-prone, and requires much expertise. More importantly, it does not scale. The knowledge base is intended to accumulate empirical and inferred knowledge from historical data for fault detection and prediction. It includes a collection of statistic properties, correlation rules, propagation modes, and fault patterns that are learned from historical data. The knowledge base is provided as a relational database with user interfaces for easy accessing, such as GUIs for system managers and APIs for fault tolerance middlewares. The information in the knowledge base may be used at two different aspects: one is to generate long-term failure distributions (e.g. the statistical properties of the time between failures MTBFs and the time to repair MTTRs) and the other is to assist online failure prediction for runtime fault management.

Last but not least, *a meta-learning mechanism* is provided to improve diagnostic and prognostic accuracy in large-scale computational infrastructures [11]. In a large HPC system, the sources of failures are many and complex; therefore it is unrealistic to expect a single method to detect and capture all of them alone. For example, some techniques are only effective for a specific device/component or for particular types of failures, and different methods have different strength and drawbacks. The proposed meta-learning approach seeks to compute a "meta-classifier" that integrates the separately learned classifiers (denoted as "base classifiers") to boost overall predictive accuracy. In particular, it learns to identify preferable combinations of base classifiers as well as their quantitative performance effects from previous results [23]. Possible meta-learning strategies include voting, arbitration, and combining. Voting means that each predictor gets one vote, and the majority (or plurality) wins. Arbitration entails the use of an "ob-

jective" judge who may choose a final outcome based upon its own prediction but cognizant of the other models. Combining refers to the use of knowledge about how predictors behave with respect to each other. The meta-learner obtains base classifications from the knowledge base, applies a preferable combination of these classifications and then produces a final result. Effective meta-learning strategies are also part of the knowledge base, and will be dynamically adjusted according to the actual observation during system operation.

4 PRELIMINARY RESULTS

As pointed out in numerous failure analysis studies, an appropriate error checking or monitoring system is of critical importance for effective failure prediction. BlueGene/L is such a system which is deployed with an extensive error checking service CMCS [21]. We have acquired two RAS (Reliability, Availability, and Serviceability) logs from the BlueGene/L systems at ANL (Argonne National Laboratory) and SDSC (San Diego Supercomputing Center) for initial evaluation of the proposed service. Both systems consist of 1024 compute nodes (2048 processors) with 32-128 I/O nodes. Table 1 summarizes these logs.

TABLE 1: SUMMARY OF FAILURE LOGS

	SDSC	ANL
Start Date	12/6/04	1/21/05
End Date	2/21/06	4/28/06
No. of Records	428,953	4,172,359
Log Size	540 MB	5 GB

First, a *three-step data prepreprocessing* is conducted to obtain a unique list of events from the logs. The preprocessing includes event categorization, temporal compression at a single location, and spatial compression across multiple locations [11]. We are currently working on the design of a stream processing engine for online more data preprocessing [20].

Next, considering the nature of RAS logs, we have examined the use of two base prediction methods (i.e. statistical based method and association rule based method) for base prediction. The *statistical based method* emphasizes on discovering probabilistic characteristics (e.g. how often and with what probability will the occurrence of one failure influence subsequent failures) and then using the obtained characteristics for failure prediction. Statistical properties are in the form of $\{f_i, f_j, t, k\}$, meaning that if f_i occurs, then f_j may occur in a time window of t with the possibility of k. *The rule-based method* builds association rules to capture causal correlations between non-fatal events occurring before each fatal event and then use them for failure prediction. Rules are in the form of $\{x_1,x_2,\ldots,x_k\}$=>y, meaning that if an occurrence of $\{x_1, x_2,\ldots,x_k\}$ is found then there is a good chance of finding a failure y in the near future (i.e. in the order of minutes).

In the rest of the paper, to evaluate the effectiveness of prediction methods, we use the 70-30 data split of the logs for learning and testing. The learned patterns or correlations, along with their accuracy, by using base predictive methods are collected and stored in a database.

Finally, a simply meta-learning strategy is explored for fault prediction. Specifically, it adaptively combines the statistical based method and the rule based method as follows:

> "Observe the events within a fixed time window before the occurrence of a failure: (1) if there exist non-fatal events, apply the rule-based method for the discovery of fault patterns and produce a warning in case of matching rules; (2) if no nonfatal event is observed, examine the occurrence of fatal events and apply the statistical based method for failure prediction; (3) if both fatal and non-fatal events are presented, use the base method that produces a prediction with higher confidence."

Figure 2-3 present the preliminary results obtained by using the proposed meta-learner on both logs, where the x-axis represents the size of prediction window. Here, *Precision* is defined as $T_p/(T_p+F_p)$, and *Recall* is defined as $T_p/(T_p+F_n)$, where T_p is number of correct predictions (i.e. true positive), and F_p is number of false alarms (i.e. false positives), and F_n is number of false negatives. A good prediction engine should provide a high value (closer to 1.0) for both metrics.

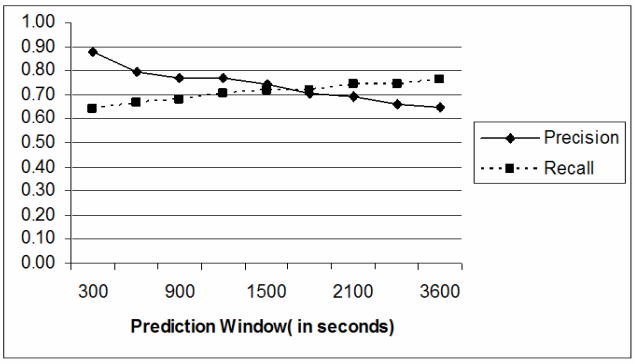

Figure 2. Fault Prediction Results with ANL BGL

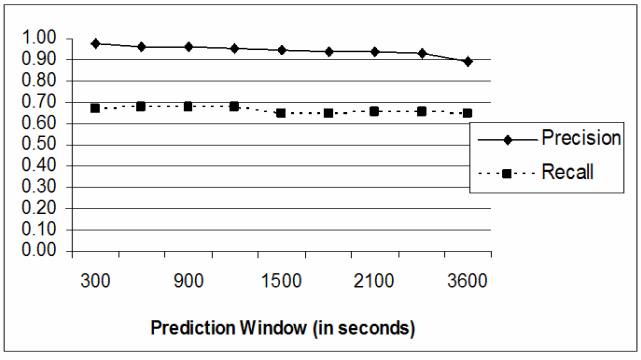

Figure 3. Fault Prediction Results with SDSC BGL

Compared to the results obtained by using a base classifier, where the statistical based method produces a 60% false positive rate and the rule-based method gives a 25%-69% false negative rate, the meta-learning approach can

boost prediction accuracy with both *Precision* and *Recall* higher than 65%. This represents up to three times improvement over using either of the base methods.

6 CONCLUSIONS

This paper presents our knowledge-based fault diagnosis and prognosis service for high performance computing systems, such as TeraGrid clusters. By leveraging existing research on system health monitoring, the proposed service can dynamically capture fault patterns and characteristics from the collected data by applying data mining and machine learning techniques. In particular, a knowledge base is used to accumulate learned patterns and modes during system operation, and the meta-learning approach is explored to improve prediction accuracy. Preliminary studies have been conducted on failure logs collected from BlueGene/L systems at SDSC and ANL.

While the knowledge based design in combination with the meta-learning approach provides an important foundation for automatic fault diagnosis and prognosis, more work are needed. Currently, we are implementing a stream processing system for continuous data flows between health data repositories and online learning by using the open-source package TelegraphCQ [20]. We continue to explore the possibilities for more comprehensive fault diagnosis and prognosis service. These include the investigation of a variety of data mining techniques and meta-learning strategies. Finally, we plan to fully implement the designed service for the use on TeraGrid.

ACKNOWLEDGMENT

The authors would like to thank TeraGrid, Argonne National Laboratory, and San Diego Supercomputer Center for the use of computing resources.

REFERENCES

[1] B. Allen, "Monitoring Hard Disk with SMART", Linux Journal, January, 2004.

[2] A. Bouteiller, T. Herault, G. Krawezik, P. Lemarinier, F. Cappello, "MPICH-V: A Multiprotocol Automatic Fault Tolerant MPI", International Journal of High Performance Computing and Applications, 2005

[3] R. Castain, T. Woodall, "The Open Run-Time Environment (OpenRTE): A Transparent Multi-Cluster Environment for High-Performance Computing", Euro PVM/MPI 2005, Italy, 2005.

[4] S. Chakravorty, C. Mendes and L. Kale, "Proactive Fault Tolerance in Large Systems", Proc. of HPCRI Workshop in conjunction with HPCA 2005, 2005.

[5] Clumon Performance Monitor. http://clumon.ncsa.uiuc.edu/

[6] I. Cohen and J. Chase, "Correlating Instrumentation Data to System States: A building Block for Automated Diagnosis and control", Proc. of OSDI'04, 2004.

[7] J. Duell, P. Hargrove, and E. Roman, "Requirements for Linux Checkpoint/Restart", Berkeley Lab Technical Report (publication LBNL-49659)

[8] E. Elnozahy et al., "A Survey of Rollback-Recovery Protocols in Message-Passing Systems", ACM Computing Surveys, 34(3), 2002.

[9] E. Gabriel, G. Fagg, et al., "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation", Proc. of The 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary, September 2004.

[10] Ganglia Monitoring System. http://ganglia.sourceforge.net/

[11] P. Gujrati, Y. Li, Z. Lan, R. Thakur, and J. White, "A Meta-Learning Failure Predictor for BlueGene/L systems", *Technical Report*, Dept. of Computer Science, Illinois Inst. of Tech, submitted to ICPP07, 2007.

[12] R. Gioiosa, J. Sancho, S. Jiang, F. Petrini, K. Davis, "Transparent Incremental Checkpointing at Kernel Level: A Foundation for Fault Tolerance for Parallel Computers", Proc. of SC2005, 2005.

[13] J. Han and M. Kamber, "Data Mining: Concepts and Techniques", 2nd ed. March 2006. ISBN 1-55860-901-6

[14] Hardware monitoring by lm sensors http://secure.netroedge.com/-lm78/info.html.

[15] Intelligent Platform Management Interface. http://www.intel.com/design/servers/ipmi/

[16] I. Lee, R. Iyer, and D. Tang, "Error/Failure Analysis Using Event Logs from Fault Tolerance Systems", Proc. of FTCS-21, 1991.

[17] Y. Liang, Y. Zhang, et al., "BlueGene /L Failure Analysis and Models ", Proc. of DSN'06, 2006.

[18] M. Lizkow, T. Tannenbaum, et al., "Checkpoint and Migration of UNIX Processes in the Condor Distributed Processing System", University of Wisconsin-Madison Computer Science Technical Report #1346, 1997.

[19] Y. Li and Z. Lan, "Exploit Failure Prediction for Adaptive Fault-Tolerance in Cluster Computing", Proc. of IEEE CCGrid'06, 2006.

[20] S. Madden, M. Shah, et al, "Continuously Adaptive Continuous Queries Over Streams", Proc. of SIGMOD'02, 2002.

[21] J. E. Moreira,G. Almási, etal, "Blue Gene/L programming and operating environment", IBM journal of Research and Development, Vol 49, Nov. 2005

[22] A. Oliner, R. Sahoo, J. Moreira, M. Gupta, A. Sivasubramaniam, "Fault-Aware Job Scheduling for BlueGene/L Systems", Proc. Of IPDPS'04, 2004.

[23] J. Keller, I. Paterson, H. Berrer, "An Integrated Concept for Multi-Criteria-Ranking of Data-Mining Algorithms," Eleventh European Conference on Machine Learning, Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination, Barcelona, Spain, 2000.

[24] The R Project for Statistical Computing. http://www.r-project.org/

[25] RAD Lab: Reliable Adaptive Distributed Systems Laboratory. http://radlab.cs berkeley.edu/

[26] R. Sahoo, A. Oliner, et al., "Critical Event Prediction for Proactive Management in Large-scale Computer Clusters", Proc. of KDD 2003: 426-435

[27] M. Scottile and R. Minnich, "Supermon: A High-Speed Cluster Monitoring System", Proc. IEEE Cluster, 2002.

[28] M. Schulz, G. Bronevetsky, R. Fernandes, D. Marques, K. Pingali, and P. Stodghill. "Implementation and Evaluation of a Scalable Application-level Checkpoint-Recovery Scheme for MPI Programs", Proc. of Supercomputing, November 2004.

[29] S. Smallen, C. Olschanowski, K. Ericson, P. Bechman, and J. Schopf, "The Inca test harness and reporting Framework", Proc. of SC04, 2004.

[30] J. Squyres and A. Lumsdaine,"A Component Architecture for LAM/MPI", Proc. of 10th European PVM/MPI Users' Group Meeting, 2003

[31] K. Vaidyanathan and K. Gross, "MSET Performance Optimization for Detection of Softtware Aging", Proc. of ISSRE, 2003.

[32] R. Vilalta and S. Ma, "Predicting Rare Events in Temporal Domains", Proc. of IEEE Intl. Conf. On Data Mining, 2002.

[33] Weka Sftware. http://www.cs.waikato.ac.nz/ml/weka/

[34] R. Wolski, N. Spring, and C. Peterson, "Implementing a Performance Forecasting System for Metacomputing: The Network Weather Service", Proc. of SC97, 1997.

[35] Y. Zhang et al., "Performance Implications of Failures in Large-Scale Cluster Scheduling", Proc. of 10th Workshop on Job Scheduling Strategies for Parallel Processing, held in conjunction with SIGMETRICS 2004.