A Meta-Learning Failure Predictor for Blue Gene/L Systems

Prashasta Gujrati[1], Yawei Li[1], Zhiling Lan[1]
Rajeev Thakur[2] and John White[3]

Illinois Institute of Technology[1]
Argonne National Laboratory[2]
San Diego Supercomputer Center[3]

{gujrpra, liyawei, lan}@iit.edu[1]
thakur@mcs.anl.gov[2]
whitej@sdsc.edu[3]

Abstract

The demand for more computational power in science and engineering has spurred the design and deployment of ever-growing cluster systems. Even though the individual components used in these systems are highly reliable, the presence of large number of components inevitably increases the failure probability of such systems. Successful prediction of potential failures can greatly enhance various fault tolerance mechanisms used in large clusters, thereby mitigating the adverse impact of failures on system productivity and total cost of ownership. In this paper, we present a three-phase failure predictor to automatically process RAS events and further discover failure patterns for prediction in Blue Gene/L systems. In particular, this paper explores the use of meta-learning to adaptively integrate base methods with the goal to boost prediction accuracy. Experiments with two RAS logs collected from Blue Gene/L systems at ANL and SDSC demonstrate the effectiveness of the proposed failure predictor.

1. Introduction

In high performance computing (HPC), the insatiable demand for more computing power in science and engineering has driven the development of ever-growing supercomputers. Large-scale clusters with hundreds to thousands of processors are being designed and deployed [26]. For systems of such scales, *reliability* is becoming a major concern as the system-wide MTBF (mean time between failures) decreases with the increasing count of system components [18]. With the greater need for fault resilience in HPC systems, a variety of fault tolerance techniques have been proposed, such as failure-aware resource management and scheduling [25], checkpointing [8, 13], run-time resilience support [3, 20], etc. It is widely accepted that effective failure analysis and prediction can significantly enhance these techniques, thereby improving system resilience to failures and reducing the total cost of ownership.

Recognizing the importance of system reliability, Blue Gene/L - one of the leading HPC systems - is deployed with a sophisticated error checking service called CMCS [24]. This service periodically gathers RAS (Reliability, Availability, and Serviceability) data from system components, in a granularity of less than 1 millisecond. Due to the very fine granularity of error checking and monitoring, a substantial amount of data can be collected. For instance, a 15-month RAS log from ANL Blue Gene/L is more than 5 GB. The potential size of RAS logs makes it a daunting challenge to not only process the data, but discover fault patterns from these logs.

To address the above problem, in this paper we propose a three-phase failure predictor for Blue Gene/L systems (see Figure 1). During phase 1 "*event preprocessing*", the raw RAS log is cleaned and categorized. During phase 2 "*base prediction*", different base learning methods are applied on the preprocessed log to identify fault patterns and correlations. During phase 3 "*meta-learning prediction"*, meta-learning is explored to adaptively integrate multiple base predictors to boost prediction accuracy. The ultimate goal of our research is to provide a framework that can automatically process

RAS events collected by CMCS and further discover failure patterns for prediction in Blue Gene/L systems.

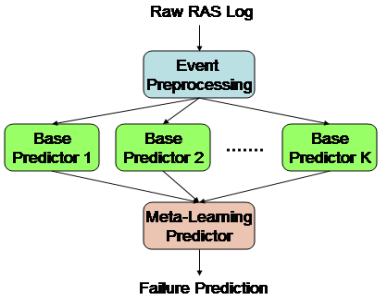


Figure 1. Three-Phase Failure Predictor

A key feature that distinguishes our work from existing failure prediction research is that we explore the use of meta-learning to improve prediction accuracy (i.e. reducing both the false negative and false positive rate). The proposed framework learns various fault patterns and correlations by combining the merits of different base predictors, thereby reducing both the false positive rate and the false negative rate. To the best of our knowledge, this is the first research on applying meta-learning to improve failure prediction in the context of high performance computing. In addition, we present a hierarchical mechanism to categorize RAS events in Blue Gene/L systems. More specifically, the paper makes the following major contributions:

- Present the use of meta-learning to boost prediction accuracy in Blue Gene/L, by combining the strengths of different base predictors;
- Develop a generic three-phase framework for end-to-end failure prediction;
- Evaluate the proposed framework with production RAS logs, and our preliminary studies show that it can effectively predict a number of failures in Blue Gene/L.

The rest of the paper is organized as follows: Section 2 gives a brief overview of Blue Gene/L and its RAS logs. Section 3 describes the three-phase failure predictor – preprocessing, base classifiers and meta-learning. Section 4 describes the related work on failure prediction in large clusters. Finally, Section 5 summarizes the paper.

2. Background

2.1. Blue Gene/L Overview

In Blue Gene/L, the computational core consists of compute and I/O nodes, which are connected in a regular topology [9]. These cores are controlled from a service node through a *control network*. The I/O nodes are used exclusively for all I/O and this is done through *functional network* whereas the compute nodes are interconnected through a *torus network*.

The Cluster Monitoring and Control System (CMCS) service is implemented on the service nodes for the purpose of system monitoring and error checking. The service node, which is available in each midplane, acquires specific device information, such as fan speeds and power supply voltages, directly through the control network. Runtime information is collected from computer and I/O nodes by a polling agent running on each BLC, reported to the CMCS service, and finally stored in a centralized DB2 repository. This system event logging mechanism works in a granularity of less than 1 millisecond. More details of the system architecture can be found in published literature [9].

2.2. RAS Event Logs

Obtaining realistic fault-related data is one of the key roadblocks to the fault prediction research. Toward this end, we have acquired RAS (Reliability, Availability and Serviceability) logs from the Blue Gene/L systems at ANL and SDSC. The major reason of using multiple RAS logs is to ensure our framework is not bias to any specific system and thus produces representative results expected in other systems as well.

The Blue Gene/L system at *SDSC (San Diego Supercomputer Center)* has a single rack with I/O rich configuration. It includes 1024 compute nodes (2048 processors) and 128 I/O nodes [27]. The Blue Gene/L system at *ANL (Argonne National Laboratory)* has 1024 compute nodes (2048 processors) and 32 I/O nodes [2]. Both systems are mainly used for scientific computing. Table 1 summarizes these logs.

	ANL	**SDSC**
Start Date	1/21/2005	12/6/2004
End Date	4/28/2006	2/21/2006
No. of Records	4,172,359	428,953
Log Size	5 GB	540 MB

Table 1: Summary of RAS Logs at SDSC and ANL

The entries in the log are records of all the RAS-related events that occur across the machine. These events include hard errors, soft errors, machine checks, and software problems. Information about scheduled maintenance, reboot, and repair is not included. Each record of the logs has a number of attributes which are described in Table 2.

The *SEVERITY* attribute can be one of the following levels - INFO, WARNING, SEVERE,

ERROR, FATAL, or FAILURE - which also denotes the increasing order of severity. INFO events are for the purpose of general information to administrators about the reliability of various hardware/services components in the system. WARNING events report unusual events in node cards, link cards, service cards or related services. SEVERE events provide more information about the reasons causing problems in node cards or service cards etc. ERROR events indicate problems that are occurring more frequently and require further attention of administrators.

Attribute	Attribute Description
Event Type	Specifies the mechanism through which the event is recorded, mostly RAS
Event Time	Time stamp associated with the reported event
Job ID	Job that detects the event
Location	Place of the event (i.e. chip/node-card/service-card/link-card)
Entry Data	Gives a short description of the event
Facility	Indicates the services/hardware component that has experienced the event
Severity	Denotes the level of severity of the reported event

Table 2: Description of attributes in the RAS log

An event with any of the above SEVERITY attributes is either informative in nature, or is related more to the initial configuration errors, and is thus relatively transparent to the applications/runtime environment. However, FATAL or FAILURE events (such as "uncorrectable torus error", "communication failure socket closed", "uncorrectable error detected in edram bank", etc.) are more severe, and usually lead to application/software crashes. Our primary focus in this study is to predict FATAL and FAILURE events (denoted as *fatal events,* while other events are denoted as *non-fatal events*). In the paper we use "failure" and "fatal event" interchangeably.

3. Three-phase Predictor

3.1. Phase 1 – Event Preprocessing

The raw logs contain many repeated or redundant entries. This is because each compute chip runs a polling agent which collects the errors reported by the chip. As each job is assigned to multiple compute chips in a midplane, any failure of the job will get reported multiple times - once from each of the assigned compute chips. Thus multiple components may report the same failure. Also, the CMCS logging mechanism records the events at a very fine granularity (in millisecond), but the recorded event time is generally in seconds leading to multiple entries of an event with the same timestamp. Therefore, before a RAS event log can be used for failure prediction in Blue Gene/L, it is essential to identify unique RAS events by preprocessing the raw RAS log, which is the focus of Phase 1. Event preprocessing consists of three steps: (1) event categorization, (2) temporal compression at a single location, and (3) spatial compression across multiple locations.

We develop *a hierarchical mechanism for event categorization* in Blue Gene/L. First, all the events are categorized based on the subsystem in which they occur, according to the LOCATION field, the FACILITY field, and the description listed in the ENTRY DATA field. The high-level categories include (1) *application* indicating events related to application instruction failures, (2) *iostreams* indicating events related to socket read/write calls and I/O procedure calls, (3) *kernel* indicating events related to instructions and alignment of data, (4) *memory* indicating events related to memory hierarchy, (5) *midplane* indicating events related to midplane configuration and switches, (6) *network* indicating events related to torus when compute chip exchange messages , (7) *node card* indicating events related to the operation and configuration of node cards, and (8) *other*. Each of them is further grouped at a finer granularity. Table 3 lists the resulting RAS categories in Blue Gene/L, where there are totally 101 subcategories.

Main Category	subcategories	Examples
Application	12	loadProgramFailure, loginFailure, nodemapCreateFailure,…
Iostream	8	socketReadFailure, streamReadFailure,…
Kernel	20	alignmentFailure, dataAddressFailure, instructionAddressFailure, …
Memory	22	cachePrefetchFailure, dataReadFailure, dataStoreFailure, parityFailure,…
Midplane	6	linkcardFailure, ciodSignalFailure, midplaneServiceWarning,…
Network	11	ethernetFailure, rtsFailure, torusFailure, torusConnectionErrorInfo,…
NodeCard	10	nodecardDiscoveryError, nodecardAssemblyWarning,…
Other	12	BGLMasterRestartInfo, CMCScontrolInfo, linkcardServiceWarning,…

Table 3. Event Categorization

Next, temporal compression and spatial

compression are used to remove duplicate entries by applying a threshold based technique [17, 23]. With *temporal compression at a single location,* events from the same location with identical values in the JOB_ID and LOCATION fields are coalesced into a single entry if reported within the threshold duration of 300 seconds. Results from the compression show that the amount of compression of FAILURE events achieved, is not significant when threshold values greater than 300 seconds is used for temporal compression. Additionally, as RAS events are logged at a sub-second frequency, taking a higher threshold value will increase the chances of different events being clustered together. With *spatial compression across multiple locations,* we remove those entries that are close to each other within time duration of 300 seconds, with the same ENTRY_DATA and JOB_ID, but from different locations.

This three-step event preprocessing provides a list of unique events which can then be used for the purpose of generating a prediction model as described in the following subsections.

Results. Table 4 summarizes the number of compressed fatal events from ANL and SDSC logs, which are divided into eight high-level event categories. As has been studied by Oliner et al., some of these failures are not true/actual failures from the perspective of applications and such failures do not result in abnormal termination of user jobs [6]. Our future work will incorporate filtering out this ambiguity of failures and analyze only those failures which will impact user jobs.

Main Category	**ANL**	**SDSC**
Application	762	587
Iostream	1173	905
Kernel	224	182
Memory	52	25
Midplane	102	97
Network	482	366
Node Card	20	17
Other	8	3
TOTAL	**2823**	**2182**

Table 4. Distribution of Compressed *Fatal* Events

3.2. Phase 2 – Base Prediction

While a number of predictive methods have been developed to date, in this study we will examine the use of two methods (i.e. statistical based method and association rule based method) as base predictors. We first describe these base prediction methods, followed by a discussion of their strengths and drawbacks.

Before presenting our prediction methodology, let's first describe performance metrics to measure prediction accuracy. A standard way of measuring the effectiveness of failure prediction is by calculating *precision* and *recall*. *Precision* is defined as the proportion of correct predictions to all the predictions made, i.e. $T_p / (T_p + F_p)$, and *Recall* is the proportion of correct predictions made to all the predictions that are possible, i.e. $T_p / (T_p + F_n)$. Here, T_p is number of correct predictions (i.e. *true positive*), and F_p is number of false alarms (i.e. *false positive)*, and F_n is number of incorrect non-failure predictions (i.e. *false negative*). A good prediction engine provides a high value (closer to 1.0) for both *precision* and *recall*.

In the rest of the paper, to evaluate the effectiveness of prediction methods, we use a standard *n-fold cross-validation* technique for the learning and testing. That is, the log is divided into *n* folds of equal size and then the (*n-1)* folds are used as training set for learning and the last fold is used for prediction and testing. As a result, there are *n* such results, which are then averaged to calculate the prediction accuracy. This technique provides a fair evaluation of prediction methods. In our experiments we have used 10-fold cross-validation.

3.2.1. Statistical-based Method

Statistical based methods emphasize on discovering probabilistic characteristics among failure events and then using the obtained characteristics for failure prediction. Similar to the work done earlier [22], our *statistical based method* utilizes the statistical characteristics of fatal events for failure prediction. More specifically, the statistical-based predictor works as follows:

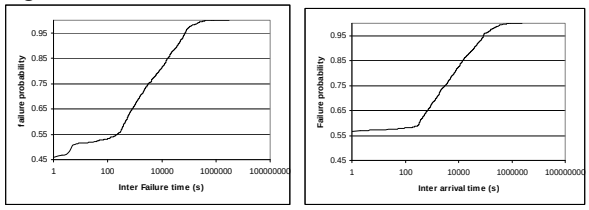
Step 1: On the learning set, obtain and verify statistical characteristics of failures (e.g. temporal correlations) from the training data;

Step 2: On the testing set, produce a warning if statistical patterns are observed in a fixed time window before the occurrence of the failure on the testing data.

Results. With both RAS logs, we investigate statistical correlations among fatal events in the training set, i.e. how often and with what probability will the occurrence of one failure influence subsequent failures. Figure 2(a) and 3(b) show the cumulative distribution function (CDF) of compressed failures for the ANL and SDSC logs respectively. We observe that a significant number of failures happen in close proximity, and our further analysis indicates that network and I/O stream related failures form a majority of such failures.

Such a temporal correlation between fatal events is then used for failure prediction in the testing set. That is, *if a network or I/O stream failure is reported, it is predicted that another failure is possible within a time*

period of 5 minutes to 1 hour. The reason for choosing this duration is that a time window smaller than 5 minutes becomes too small for taking preventive action based on the prediction, whereas a time window larger than 1 hour will induce an increased monitoring load on the system if such a scheme is implemented in an online system as it will require maintaining the history of all the events for the duration of 1 hour after a failure has been reported. Also, the processing/analysis cost of these events for failure prediction may not be trivial. The values of *precision* and *recall* for both the logs are shown in Table 5.



(a) ANL (b) SDSC

Figure 2.CDF of Failure Probability

Log Name	Precision	Recall
ANL	0.5157	0.4872
SDSC	0.2837	0.3117

Table 5. Prediction Results by Using Statistical Correlation between Fatal Events

Discussion. A drawback with this prediction strategy is that *precision* is low as the number of failures which do not have subsequent failures occurring in the next time window is substantial. Another drawback is that apart from I/O stream and network failures, none of other categories of failures has such a temporal correlation. As shown in Table 4, other types of failures constitute 42% of the failures in both the logs, hence a substantial number of failures cannot be predicted by using this method. Further, as can be observed in Table 5, the *precision* and *recall* values may vary significantly for different Blue Gene/L systems.

3.2.2. Rule-based Method

Next, we examine *causal correlations* between non-fatal and fatal events and then use the correlations for failure prediction. A widely used technique for extracting such a causal correlation is to build association rules.

Association rules were first introduced by Agrawal et al. to analyze customer habits in retail databases [1]. Association rule is an implication of the form $X \rightarrow Y$, where the rule *body* X and *head* Y are subsets of the set I of *items* ($I = \{I_1, I_2,...,I_n\}$) within a set of itemsets D and $X \cap Y = \Phi$. A rule $X \rightarrow Y$ states that the transactions T that contain the items in X are *likely* to contain also the items in Y. Association rules are characterized by two measures: the *support*, which measures the percentage of transactions in D that contain both items X and Y; the *confidence*, which measures the percentage of itemsets in D containing the items X that also contain the items Y. The problem of mining association rules from a set of itemsets D consists of generating all the association rules from a set of items that have *support* and *confidence* greater than user-defined thresholds.

Lower value of *support* and *confidence* will generate larger amount of rules, thereby requiring longer time and more memory space to build the rules. Higher value of *support* and *confidence* will reduce the number of frequent itemsets and thereby reduce the number of generated rules, and consequently reduce the time and memory required for rule generation. However, the lesser number of rules, in turn, reduces the opportunities of capturing causal relationships among items, thereby reducing the rate of discovering fault patterns. We have set the minimal value for *support* as of *0.04* and *confidence* of *0.2* in our experiments. The low values for the parameters ensure that even if a failure event is reported very infrequently but it leads to a rule which is very strong, then it gets generated. This avoids the problem of infrequent items which may happen if higher values of *support* are used. Parameter values lower than these lead to exhaustion of compute resources because of generation of too many rules.

Our rule based method works as follows:

Step 1: On the learning set, for each fatal event identify the set of non-fatal events frequently preceding it within a fixed time window (i.e. *rule generation window*). The set, including the fatal event and their precursor nonfatal events, is called an *event-set*.

Step 2: Apply the standard association rule algorithm to build rule models for event-sets that are above the minimum user-defined support [1, 15].

Step 3: Combine rules as we focus on predicting whether there is an imminent failure. For example, if $\{e_1,e_2,...,e_k\} \rightarrow f_1$ and $\{e_1,e_2,...,e_k\} \rightarrow f_2$ are generated by Step 2, we combine them as $\{e_1,e_2,...,e_k\} \rightarrow \{f_1,f_2\}$.

Step 4: Sort the generated rules in descending order of their confidence values.

Step 5: Evaluate rules generated with different rule generation windows, and select the window size that can best capture a variety of fault patterns between non-fatal and fatal events.

Step 6: On the testing set, use the rules generated to produce a warning if an association rule is observed within a fixed time window (i.e. *prediction window*)

before the occurrence of the failure. If multiple rules are observed, select the rule with the highest confidence.

To determine the optimum size of the *rule generation window*, we conducted experiments with window size ranging from 5 minutes to 1 hour. From the observed values of *precision* and *recall* for each of the *rule generation window*, we chose the window size which gives the best *precision* with highest *recall*. Thus, the *rule generation window* is 15 minutes for ANL log and 25 minutes for SDSC log. These rule generation windows were used for subsequent failure prediction in the testing set as described in Step 6.

```
nodeMapFileError ==> nodeMapCreateFailure: 1
nodeMapError ==> nodeMapCreateFailure: 0.947368
controlNetworkHWCSError ==> nodeConnectionFailure: 0.708333
ddrErrorCorrectionInfo maskInfo ==> socketReadFailure: 0.697674
ciodRestartInfo midplaneStartInfo controlNetworkInfo ==> rtsLinkFailure: 0.696629
ciodRestartInfo midplaneStartInfo ==> rtsLinkFailure: 0.688889
nodecardVPDMismatch nodecardAssemblySevereDiscovery nodecardFunctioanlityWarning ==> linkcardFailure: 0.636364
nodecardVPDMismatch nodecardFunctioanlityWarning midplaneLinkcardRestartWarning ==> linkcardFailure: 0.6
coredumpCreated ==> loadProgramFailure: 0.583333
midplaneStartInfo controlNetworkInfo BGLMasterRestartInfo ==> cacheFailure: 0.555556
nodecardDiscoveryError nodecardFunctioanlityWarning endServiceWarning midplaneLinkcardRestartWarning ==> linkcardFailure: 0.545455
```

Figure 3. Partial List of Generated Association Rules with Their Confidence Values

Results. Figure 3 shows a partial list of generated rules along with their confidence values. Figure 4 presents the results of *precision* and *recall* for ANL and SDSC logs. As we can see, the *precision* value is in the range of 0.7 – 0.9, while *recall* is not as satisfying (ranging between 0.22 and 0.55). The figures also show that as the prediction window increases, *recall* improves without a substantial loss in *precision*.

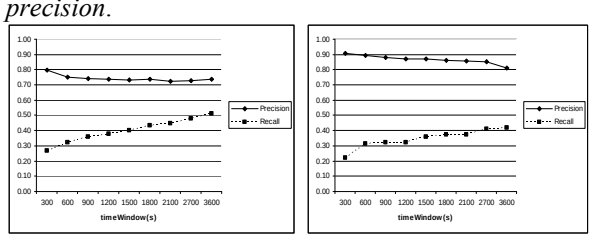


Figure 4. Prediction Results (left ANL, right SDSC)

As observed in Figure 4, the recall value is always smaller than 0.55 even when a larger prediction window is used. By further analysis, we find that a substantial number of failures (i.e. 31%-66% of failures from the ANL log and 47%-75% of failures from the SDSC log) do not have any precursor non-fatal events.

Discussion. In short, the rule-based predictor can effectively capture causal relationships among RAS events (e.g. high *precision* values); however, it is limited by the proportion of fatal events without any precursor warnings (e.g. low *recall* values).

3.3. Phase 3 – Meta-Learning Prediction

As shown in Section 3.2, it is unlikely to produce an effective failure prediction by using either of base predictors alone. The statistical based method is effective in leveraging the temporal correlation among fatal events, but suffers from low coverage of failures and cannot capture causal relationships between non-fatal events and fatal events, thereby ending up with low prediction accuracy. The rule based prediction method is good at discovering causal relationships between non-fatal and fatal events, but its effectiveness is limited by the portion of fatal events without any precursor events. To address the problem, a meta-learning mechanism is explored in our framework to boost failure prediction.

Meta-learning or *ensemble-learning* can be loosely defined as learning from learned knowledge [7]. It is a technique that deals with the problem of computing a "global" predictor from the separately learned "base predictors" to boost overall predictive effectiveness [4]. In particular, it learns to identify preferable combinations of based classifiers as well as their quantitative performance effects from previous results. A widely used approach is called stacked generalizations, in which three metrics, namely conflict, coverage, and diversity, are developed to measure its accuracy [4]. Another important approach is called reinforcement learning which attempts to take the right bias according to the type of input-output distributions [21].

Our *meta-learner* uses the *coverage* based stacked generalization which adaptively integrates the statistical based method and the rule based method. More specifically, it works as follows:

- On the learning set, (1) obtain the statistical characteristics of failures with corresponding confidence values as described in §3.2.1 (Step 1); (2) generate association rules between nonfatal and fatal events with corresponding confidence values as presented in §3.2.2 (Step 1- 5);
- On the testing set, observe the events within a fixed time window before the occurrence of a failure: (1) if there are nonfatal 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.

Results. The prediction results obtained by using the proposed meta-learner are presented in Figure 5. As we can see, with both RAS logs, the prediction accuracy is significantly improved: (1) with the ANL log, the precision decreases from 0.88 to 0.65, while the recall increases from 0.64 to 0.78 as the prediction window increases from 5 minutes to 1 hour; (2) with the SDSC log, the precision decreases from 0.99 to 0.89, whereas the recall is always around 0.65 as the prediction window increases from 5 minutes to 1 hour. Further, the precision decrease is more pronounced for results in ANL log as compared to SDSC log. This is so because the number of high confidence rules generated in ANL is less when compared to the rules learnt in SDSC resulting in higher *false positive* values which affect the *precision* of prediction.

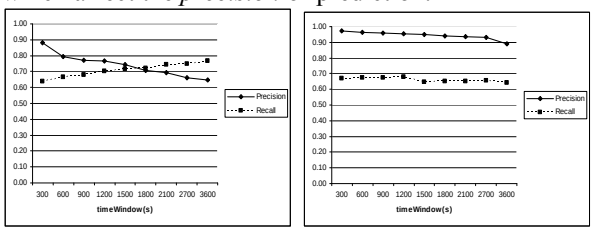


Figure 5. Meta-learning Results (left ANL, right SDSC)

As discussed in the previous sub-sections, *recall* was ranging from 0.22 to 0.55 for both the RAS logs by using either of base prediction methods. We also observed a high *precision* for rule based method but low for the statistical method (as given in Table 5). Compared to either of these predictors, the combined meta-learner has *recall* which is consistently more than 0.65 for all prediction windows along with a consistently high value for *precision*. Thus, the meta-learner based predictor can significantly boost the prediction accuracy.

Discussion. The main reason for such a significant improvement is that in a large cluster such as Blue Gene/L, the sources of failures are many and complex, thus it is improbable for a base prediction method to capture all of them alone. Instead, a meta-learning based approach can combine the strengths of multiple base predictors and discover various failures to give better failure prediction results.

The meta-learning strategy is simple and time efficient. Its overall cost is about the same as the rule-based method: the rule generation process varies from 35 seconds for a 5-minute prediction window to 167 seconds for a 1-hour prediction window; and the rule matching process is trivial. Therefore, it is practical to deploy the meta-learner as an online prediction engine.

4. Related Work

Generally speaking, fault prediction can be approached from two different angles: *model-based* or *data-driven*. A *model-based* approach derives a probabilistic or analytical model of the system [12]. A warning is triggered when a deviation from the model is detected [16]. Examples include an adaptive statistical data fitting method called MSET developed by Gross et al. [24], a Semi-Markov reward model [30], a neutral-network based classification for forecasting hardware failures [29], a naive Bayesian based algorithm for predicting disk drive failures [14], etc. Most of them either focus on specific types of failures or target small scale systems, thus not sufficient for large clusters.

Data-driven approaches, such as using data mining in combination with intelligent systems, focus on learning and classifying occurring faults from historical data without assuming a priori model ahead of time. There are several recent research efforts on failure prediction in large clusters. Vilalta and Ma apply frequent itemset mining for failure prediction in a networked system comprising 750 hosts [19]. Sahoo et al. present several methods, including a rule-based data mining method, to predict a set of target failure events in a 350-node IBM cluster [28]. Perhaps, the work by Liang et al. is the most closely related work to ours [22]. The paper focuses on utilizing statistical characteristics among RAS events (e.g. spatial or temporal correlation) for prediction in a Blue Gene/L system. Different from these studies, our work emphasizes on exploiting meta-learning to boost failure prediction by combining statistical based method and rule based method. In a recent work [6], Oliner et al. has also pointed out the importance of using ensemble learning for failure prediction in large-scale clusters.

In [23], the authors give a detailed filtering process for a Blue Gene/L log. Our proposed framework utilizes a similar filtering process for log preprocessing. The major difference is that we categorize RAS events at a much finer granularity, which is useful in capturing more detailed fault patterns.

5. Summary

In this paper, we have presented a three-phase framework for failure prediction in Blue Gene/L, namely *event preprocessing, base prediction and meta-learning prediction*. In particular, we have proposed the use of meta-learning for improving

failure prediction in large scale clusters such as Blue Gene/L. The proposed framework adaptively integrates and combines two widely used base prediction methods (i.e. statistical based method and rule-based method) for discovering various fault modes (e.g. temporal correlations among failures and causal correlations among non-fatal and fatal events). Our preliminary results are promising. As compared to using a base predictor alone, the proposed meta-learning prediction can significantly improve failure accuracy by up to three times. Our primary goal is to open up further research on designing effective failure analysis and prediction systems for large-scale clusters, in particular those used in high performance computing. Further, the proposed meta-learning mechanism should be further examined for advancing failure prediction in large clusters.

Although in this paper we focus on Blue Gene/L, we believe the proposed three-phase framework can be extended for general failure analysis and prediction in other large-scale clusters. For large-scale clusters which do not have a CMCS type of facility, the key issue is how to develop a monitoring tool which is capable of gathering fault-related information from low-level devices and archive the information in a centralized repository. This data can then be used by the three-phase framework for failure analysis and prediction.

References

[1] R. Agrawal, R. Srikant, "Fast Algorithms for Mining Association Rules", *VLDB*. Sep 12-15 1994, Chile, 487-99

[2] ANL Blue Gene/L Homepage.. www.bgl.mcs.anl.gov

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

[4] P. Chan and S. Stolfo, "Metalearning for Multistrategy and Parallel Learning", *Proc. of Multistrategy Learning Workshop*, Center for Artificial Intelligence, 1993.

[6] A. Oliner and J. Stearly, **"**What Supercomputers Say: A Study of Five System Logs", *Proc. of DSN* 2007.

[7] R. Polikar, "Ensemble Based Systems in Decision Making", *IEEE Circuits and Systems Magazine*, vol.6, no. 3, pp. 21-45, 2006.

[8] E. Elnozahy and J. S. Plank, "Checkpointing for Peta-Scale Systems: A Look into the Future of Practical Rollback-Recovery", *IEEE Transactions on Dependable and Secure Computing*, Volume 1, Number 2, 2004, pp. 97-108.

[9] A. Gara, M. A. Blumrich et al., "Overview of the Blue Gene/L System Architecture", *IBM J. Res. & Dev.* 49, No. 2/3, 195–212, 2005.

[10] M. Goldszmidt, I. Cohen, A. Fox and S. Zhang, "Three research challenges at the intersection of machine learning, statistical induction, and systems", *HOTOS 2005*.

[11] Hoffmann, Salfner et al, "Advanced Failure Prediction in Complex Software Systems", *Proc. of SRDS,* 2004.

[12] A. Goyal, S. Lavenberg, and K. Trivedi, "Probabilistic Modeling of Computer System Availability", *Annals of Operations Research,* 1987.

[13] 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.

[14] G. Hamerly and C. Elkan, "Bayesian Approaches to Failure Prediction for Disk Drives", *Proc. of ICML*, 2001.

[15] J. Han, J. Pei, Y. Yin, R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach", *Data Min. Knowl. Discov*. 8 (1): 53-87 2004.

[16] J. Hellerstein, F. Zhang, P. Shahabuddin, "A Statistical Approach to Predictive Detection", *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 2001.

[17] R. K. Iyer, L. T. Young, V. Sridhar, "Recognition of error symptoms in large systems", *Proceedings of 1986 ACM Fall joint computer conference*, 1986.

[18] D. Kerbyson, A. Hoisie, and H. Wasserman, "Use of Predictive Performance Modeling During Large-scale Systems Installation", *Proc. of the 1st International Workshop on Hardware/Software Support for Parallel and Distributed Scientific and Engineering Computing*, 2002.

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

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

[21] L. Lanzi, W. Stolzmann, S. Wilson, "Learning Classifier Systems, From Foundations to Applications", *Lecture Notes In Computer Science,*Vol. 1813, 2000.

[22] Y. Liang, Y. Zhang, M. Jette, A. Sivasubramanium, R. Sahoo, "BlueGene/L Failure Analysis and Prediction Models", *Proc. of DSN,* 2006.

[23] Y. Liang, Y. Zhang, A. Sivasubramanium, R. Sahoo, J Moreira, M. Gupta, "Filtering Failure Logs for a BlueGene/L Prototype", *Proc. of DSN,* 2005.

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

[25] A. Oliner, Ramendra K. Sahoo, José E. Moreira, Manish Gupta, Anand Sivasubramaniam, "Fault-Aware Job Scheduling for BlueGene/L Systems", *IPDPS,* 2004.

[26] The TOP500 Supercomputer Sites. www.top500.org

[27] SDSC Blue Gene/L Homepage. www.sdsc.edu/us/resources/bluegene

[28] R.K. Sahoo, A.J. Oliner et al., "Critical event prediction for proactive management in large-scale computer clusters", *Proc. of KDD*, 2003, pp. 426-435.

[29] D. Turnbull, N. Alldrin, "Failure Prediction in Hardware Systems", *UCSD CSE221 Project*, 2003.

[30] K. Trivedi and K. Vaidyanathan, "A Measurement-based Model for Estimation of Resource Exhaustion in Operational Software Systems", *Proc. of the 10th Int'l Symposium on Software Reliability Engineering*, 1999.