

Navigating the Blue Waters: Online Failure Prediction in the Petascale Era

Ana Gainaru
University of Illinois at
Urbana-Champaign
againaru@illinois.edu

Mohamed Slim Bouguerra
INRIA/UIUC
mohamed-slim.bouguerra@imag.fr

Franck Cappello
Argonne National Laboratory
fci@lri.fr

Marc Snir
Argonne National Laboratory
snir@illinois.edu

William Kramer
National Center for Supercomputing Applications
wtkramer@illinois.edu

Abstract

At the scale of today's large scale systems, fault tolerance is no longer an option, but a necessity. As the size of supercomputers increases, so does the probability of a single component failure within a time frame. With a system MTBF of less than one day and predictions that future systems will experience delays of couple of hours between failures, current fault tolerance strategies face serious limitations. Checkpointing is currently an area of significant research, however, even when implemented in a satisfactory manner, there is still a considerable loss of computation time due to frequent application roll-backs. With the growing operation cost of extreme scale supercomputers like Blue Waters, the act of predicting failures to prevent the loss of computation hours becomes cumbersome and presents a couple of challenges not encountered for smaller systems.

In this paper, we present a novel methodology for truly online failure prediction for the Blue Water system. We analyze its results and show that some failures types can be predicted with over 60% recall and a precision of over 85%. The failures which represent the main bottlenecks are discussed in detail and possible solutions are proposed by investigating different prediction methods. We show to what extent online failure prediction is a possibility at petascale and what are the challenges in achieving an effective fault prediction mechanism for Blue Waters.

Categories and Subject Descriptors CR-number [subcategory]: third-level

General Terms resiliency, fault tolerance

Keywords failure prediction, torus spatial patterns, online prediction, fault tolerance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CONF 'yy, Month d-d, 20yy, City, ST, Country.
Copyright © 20yy ACM 978-1-nnnn-nnnn-ny/mm...\$15.00.
<http://dx.doi.org/10.1145/nnnnnnn.nnnnnn>

1. Introduction

The last years have been a fertile ground for the development of many scientific and data-intensive applications in all fields of science and industry. These applications provide an indispensable mean of understanding and solving complex problems through simulation and data analysis. As large-scale systems evolve towards post-petascale computing to accommodate applications increasing demands for computational capabilities, many new challenges need to be faced, among which fault tolerance is a crucial one [1, 2]. At the scale of today's large scale systems, fault tolerance is no longer an option, but a necessity. With failure rates predicted in the order of tens of minutes [26] for the exascale era and applications running for extended periods of time over a large number of nodes, an assumption about complete reliability is highly unrealistic. Because processes from scientific applications are, in general, highly coupled, even more pressure is put on the fault tolerance protocol since a failure to one of the processes could eventually lead to the crash of the entire application.

By far the most popular fault tolerance technique to deal with application failures is the Checkpoint-Restart strategy. Unfortunately, classical checkpointing, as used today, will be prohibitively expensive for exascale systems [13]. A complement to this classical approach is failure avoidance, by which the occurrence of a fault is predicted and proactive measures are taken. Failure avoidance uses the information received by a failure predictor to facilitate proactive fault tolerance mechanisms such as proactive job migration or proactive checkpoint. In general, failure prediction is based on the observation that there is a Fault-Errors-Failure propagation graph [24]. The fault generates a number of errors that could be observable at the system level, which represent either notifications in the log files or changes in performance metrics. The propagation chain ends with the failure which is observed at the application level and usually is represented by an application interruption. However, the error could propagate and generate other effects like performance degradation.

Over the years, different methods have been developed that deal with failure prediction in the HPC community [24], methods that have been used extensively on different HPC systems and that present a variety of results. In our previous work we introduced the concept of signal analysis in the context of event analysis, which allowed us to characterize the behaviour of different events and to analyze them separately depending on their individual behaviour [8]. All results show that failure prediction is a theoretical viable solu-

tion for future fault tolerance techniques. However, a large fraction of experiments and results for failure prediction methods have been the result of the analysis of past generation HPC machines in simulated online environments. The scale of today's systems has increased by two orders of magnitude, with forecasts for exascale showing an even higher increase [26]. Moreover, simulated online predictions, the method of choice for extracting prediction results, assumes tuning the parameters of all prediction modules in the offline phase in order to achieve the best possible results in the online phase. While these methods show prediction results that could theoretically be achieved in real scenarios, they do not reflect the reality of running in realtime and predicting failures using best local parameters.

In this paper, we introduce a methodology for truly online predictions and we show that using this model, prediction is possible and gives good results on small systems. Moreover, we present an analysis of the Blue Waters system by investigating the feasibility of online failure prediction methods on a petascale machine. With a sustained performance of 1 Petaflop on a range of real-world science and engineering applications, the Blue Waters supercomputer is representative for today's large scale systems and provides new insights into the performance of current fault predictors. Specifically, the contributions of the paper are the following:

1. We introduce a novel methodology for online failure prediction and present its results on different systems. Moreover, we show into detail what category of failures are predicted and what are the limitations imposed by each system.
2. We analyze the characteristics of failures from the Blue Waters system and study their effect on the results given by the online failure prediction. We make a couple of key observations about the difference in behaviour between Blue Waters and previously smaller systems and propose specific optimizations for this system that increase the results. A detailed analysis of the prediction results is also given.
3. We show the impact of failures on applications and investigate how the prediction method is influenced when looking from the application's perspective and not the system. We show that by taking the application into consideration into our methodology, the prediction becomes more useful for fault avoidance techniques.
4. We propose a novel methodology for predicting the locations on which a failure propagates by taking into consideration the topology of the investigated system which increases the coverage of a predictor.

The rest of the paper is organized as follows. Section 2 gives an overview of the related work, highlighting their limitations. Section 3 presents the methodology for a real online failure predictor and study its results when applied on a smaller system. In section 4, the focus moves on Blue Waters and on the differences in prediction results compared to smaller systems, from both the system's perspective and from the application level. Section 5 presents the impact of the novel location propagation methodology and section 6 gives a general view of different types of failures and how their characteristics influence the results. We conclude with a summary of our work and a discussion of future research directions.

2. Related work

Failure prediction has been extensively used over the years in numerous science and engineering fields [16, 18], from research in material science [5] and engine fault tolerance [19] to studying fatigue predictions for rails [3]. Failure prediction in computer science and specifically in HPC, followed two distinct directions: component level and system level failure prediction. The first level

includes methods that observe components (hard drives, mother boards, DRAM, etc) with their specific parameters and domain knowledge and that define different approaches that give best prediction results for each [4, 21]. In [10], the authors investigate the abilities of two Bayesian methods to predict disk drive failures based on measurements of drive internal conditions. More recently, in [12], the authors analyze DRAM failures and conclude that simple page retirement policies might be able to mask a large number of DRAM errors in production systems.

The second level is represented by system level failure prediction, in which monitoring daemons observe different system parameters (system log, scheduler logs, performance metrics, etc) and investigate the existence of correlations between different events. Initial research from this level was focused on applying failure prediction on networks [25], server environments [22] or computer clusters [23]. More recent methods for short-term failure prediction are typically based on runtime monitoring as they take into account the current state of the system. One example of this type of approach is [29] in which matrices are used to record system performance metrics at every interval. The algorithm afterwards detects outliers by identifying the nodes whose behaviour is far away from the majority. Failure prediction for HPC systems has been receiving increasingly attention in the last years, different methods being proposed for a wide variety of systems.

Current research is focusing on using system logs, scheduling logs, performance metrics or usage logs in order to extract a correlation between events generated by a system. There are numerous methods, starting with simple brute force extraction of rules between non-fatal events and failures [24] with more sophisticated techniques. In [27], the authors are using a meta-learning predictor to choose between a rule-based method and a statistical method depending on which one gives better predictions for a corresponding state of the system. Other research include SVM [6], hidden-Markov chains [17] or Bayesian networks [20]. A slightly different approach is given in [15] where the authors investigate parameter correspondence between different application log messages for extracting dependencies among components.

In our previous work we introduced a novel methodology for extracting correlations between events that uses different signal analysis modules [8]. By treating events as signals we were able to characterize the normal behaviour of each type of event and how failures affects them. Our experiments follow the workflow from previous studies by looking at historic log files divided in two parts, one for training and extracting the correlations and one for the actual prediction. We call this method simulated online because it assumes tuning the parameters of all modules in the training phase in order to achieve the best possible results in the online phase. Currently the best obtained results show a recall of 70% with a precision of 80% [14] for the Blue Gene/P system. Our results on Blue Gene/L shows 50% recall with 90% precision (or 60% with 70% depending on how the parameters are tuned) for a lead time between the prediction and failures of more than 10s [9]. While these results are theoretically possible, in real online scenarios the parameters cannot be tuned to offer best results for an unknown future and so the results are much lower than the theoretical peak. To the best of our knowledge, this paper is the first to offer a methodology for truly online failure predictions for HPC systems.

The paper also focuses on analyzing how online prediction can be achieved on both smaller and large-scale systems and how the characteristics of each system influence the results. In addition, we look at how jobs are affected by failures and how this information influences the application level failure prediction. In [28], the authors also make a difference between system and application failures by using RAS logs and job logs for filtering out the failures that do not have any effect on the running jobs. However, they fo-



Figure 1. Failure prediction: simulate online

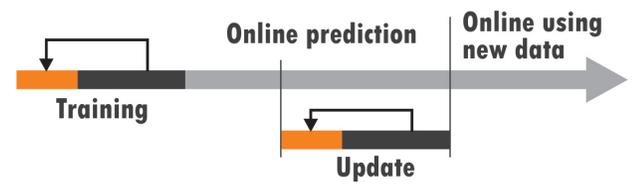


Figure 2. Online failure prediction

cus on an analysis on the failure distribution and do not investigate how this information affects the results of failure predictors.

3. Online failure prediction

In general, the prediction methods follow the workflow presented in Figure 1. The historic log files are divided in two parts. The first part is used for training and different methods are used to learn patterns and correlations between different events in the system. This part usually represents 10% of the whole log and can be anything between a couple of weeks [27] to months [14]. These patterns are then used in the second part of the analysis, by applying them on the second part of the log in order to predict failures. Based on the actual failures from this part, recall and precision values can be computed for the given method.

A prediction where the predicted failure occurred in the given time interval and on the given location is called a true positive. An incorrect prediction happens when the predicted failure does not occur in the given time frame, in which case it is called a false positive. Failures that occur without being predicted are false negatives. These three values define the two metrics that we use to measure the performance of a predictor: precision and recall. Precision defines the quality of all the predictions and it is equal to the ratio of true positives to false positives. Recall represents the coverage of a predictor and defines the ratio of failures predicted to the total failures in the system.

Every step in the training method contains parameters used to decide when a pattern is reliable enough to be used in the second phase. These parameters have a high influence on the final results of a predictor. For example, figure 3 shows how prediction and recall are influenced by different values of the parameters used by one of our prediction system. Depending on the results obtained on the second part of the log, the parameters can be tuned and the analysis on the first part of the log can be redone in order to increase the accuracy of the final results.

Choosing and tuning the parameters is usually a manual process and can be done using domain knowledge about the system or based on previous experience with other systems. Repeating the prediction for the same testing piece of log until obtaining the best possible results shows the best possible results that can be achieved with a predictor. While this is a good way of analyzing the limitations of a predictor it is not a valid methodology when deploying a predictor to work online on a real system.

The solution we propose is presented in figure 2 which is currently implemented in the ELSA toolkit. The historic log file is still divided into two parts, one for training and one for testing, however this time the training is using the whole methodology as before as

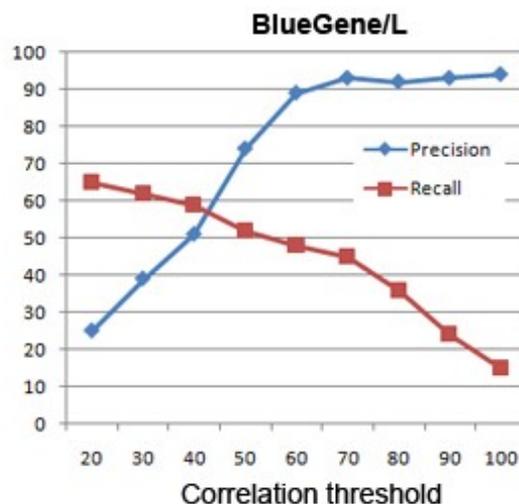


Figure 3. Precision and recall (in percentage) for different values for one of ELSA's parameters, the correlation threshold

described in figure 1. The only difference is that the training part is done automatically, either by implementing rules as to how parameters modify the precision and recall ratio or by a brute force strategy. After the best results have been reached for the simulate online part, the process stops and the parameters are used online on the incoming stream of events. The methodology can be tested on a historic log, by dividing the log into 3 parts, one for training, one for simulate online and the rest for online predictions. The best parameters are chosen based only on the training and simulate online parts either manual or automatic, then online predictions are made on the third part of the log only once and the results of this one time execution defines the recall and prediction values.

In time the learned patterns used for prediction become less and less accurate because of new updates in the system and so both precision and recall decrease over time. Figure 2 deals with this problem by triggering a new session of training and simulate online in parallel with the online prediction each time the precision or/and recall decrease below a threshold.

In our previous work we introduce ELSA (Event Log Signal Analyzer) [9], a failure prediction tool based on signal analysis concepts. ELSA is working on a two-phase algorithm, first offline by combining signal analysis with data mining algorithms to extract correlations between events and secondly online by using the correlations to predict future events. Signal analysis allows ELSA to characterize the behavior of events affecting the system, highlighting the differences between failures. Data mining is an efficient method for extracting patterns in high-dimensionality sets so ELSA is using them to provide accurate correlations between defined behaviours. ELSA was successfully applied on several HPC systems, being able to predict over 40% of the failures with a very high accuracy.

A condensed figure of ELSA's workflow is presented in figure 4. The tool works in two distinct phases: offline and online. In the offline phase the input is represented by the training part of the log and the output is formed by the correlation chains between events. The algorithm executed by ELSA is the following: extract all possible event types from the log, identify anomaly moments for each of the events and then correlate the anomaly moments of non-fatal events between each other and with failures. The rules or chains of correlations between events are then used in the online phase in order to trigger predictions. The online module classifies

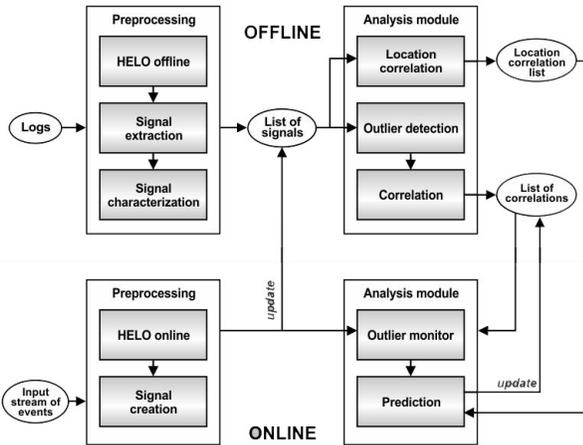


Figure 4. ELSA workflow

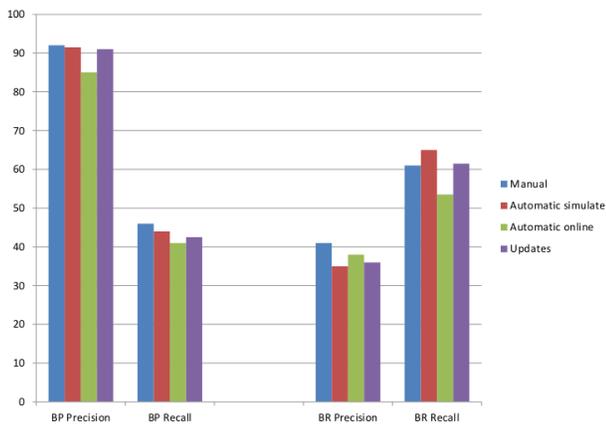


Figure 5. Precision and recall for simulate online and online

the incoming event, detects if the event indicates an anomaly and then based on the event's type, it searches the correlation chain and triggers predictions if necessary.

We implemented the online prediction methodology described above and integrate it into the ELSA toolkit. We tested our method, first onto the same Blue Gene/L log used in our previous studies and then on the Blue Waters system. For the training phase, we chose to tune the parameters in an automatic way with a hybrid version of domain knowledge and brute force. There are three categories of parameters in ELSA's testing phase, one in the classification phase, two in the outlier identification phase and two in the correlation extraction phase. The last two parameters have a direct relation with both precision and recall (for example as shown in figure 3 as the correlation threshold parameter increases, the recall decreases and precision increases). However, for the second type of parameters there is no straightforward relation. This is the motive of choosing to implement the hybrid method of brute force (for the second phase) and guided search (for the first and third phase) for finding the parameters that give either best precision, or best recall in the simulate online part of the log.

The first experiment was done on the Blue Gene/L log which was downloaded from [?]. In our previous studies we focused on offering the best precision possible because we desired to have as few wrong predictions as possible. There are fault avoidance

techniques for which the overhead of a misprediction is higher than the benefit of covering a large set of failures in which case highest precision is the best solution. One example is the charm++ framework in which migration is very cheap compared to the time wasted restarting an application because of an unpredicted failure.

In one of our previous studies, we investigated the impact of combining our prediction method with a multi-level checkpointing strategy and we observed that a decrease in recall has a higher impact on the benefit of this hybrid fault tolerance method than precision ???. For this reason, in this paper we remade our previous experiments on Blue Gene/L focusing on both best precision and best recall. Figure 5 shows the precision and recall for different scenarios. The left part of the figure shows the results when focusing on obtaining the best precision possible and the right side focuses on best recall.

The Blue Gene/L log used contains events generated between June 2005 and January 2006. We divided the logs in three phases, 3 months for training, 3 months for simulate online and 2 months for online.

The first bar in each column (the blue bar) shows the results for the manual simulate online method. We tuned the parameters and re-executed the training phase until we obtained the best results in the online part of the log. The second bar from the left (the red bar) in each column represents the results when using the automatic script for finding the best parameters. For computing the first two bars, the first part of the log is used for extracting patterns and the last part is used for computing the recall and precision value. The other two experiments compute these values also using the last part of the log but using the first two parts for training as described in figure 2. This way all four experiments compute the recall and precision by predicting failures only on the online part of the logs and by using only data from the training phase. The online methodology uses the simulate online part of the log only to tune the parameters and so optimizing the correlations found in the training part. This means that if a correlation cannot be found in the training part and its present in the simulate online part our predictor will not see it. Because of this the comparison is fair.

The results using the automatic script are very similar to the ones obtained by manual tuning. For the best recall values, the automatic method gives a better recall, however, paying the price of having a lower precision. The variations happens because the automatic algorithm goes through a larger set of parameter combinations and, particular case it managed to go a step further than what we did in the manual analysis. However, overall the results are very similar.

The third bar (green bar) presents recall and precision values for the online methodology when no updates are being made and the last bar when there is one update. When no update is made, the recall and precision values both are much smaller than in the simulate online case. However, after one update the values become very similar.

The results on Blue Gene/L shows that the online methodology gets similar results by choosing automatically the parameters that otherwise should be manually tuned and updated every couple of months. However, when we applied the same strategy on the Blue Waters systems we observed a couple of limitations. We will focus on analyzing these limitations and proposing solutions in the next section.

4. Blue Waters

4.1 System characteristics

By providing a sustained performance of 1 Petaflop on a range of real-world science and engineering applications, the Blue Waters supercomputer is currently one of the most powerful supercomput-

ers. Because the size and complexity of the system our analysis modules needed to be adapted in order to generate useful information. The NCSA (National Center for Supercomputing Application) has launched for production the Blue Water system on March 28, 2013.

The Blue Waters system is a Cray XE/XK hybrid machine composed of AMD 6276 "Interlagos" processors and NVIDIA GK110 "Kepler" accelerators all connected by the Cray Gemini torus interconnect. Divided into 237 Cray XE6 and 32 Cray XK7 cabinets, Blue Waters contains over 25 thousand computing nodes, reaching a peak performance of 11.6 Petaflops and offering a total system memory of over 1.4 PetaBytes. The online storage gives 26.4 PB of total usable storage with an aggregate I/O bandwidth of over 1 TB/s. In August 2013, Blue Waters was upgraded with 12 additional Cray XK racks, each with 96 nodes. This boosts the systems peak performance to over 13 petaflops. Our analysis was mainly done on the system before the extra nodes have been added since the system is still stabilizing after the upgrade. However, a couple of experiments were made after the racks were added in order to see the impact of using logs generated by an unstable system on the training modules.

There are numerous applications that are running on Blue Waters. Examples include earthquake engineering for which simulations want to capture seismic waves in 1Hz range which is 256 times more computationally demanding than current simulations; cosmology applications that desire to model the first billion years after the Big Bang; epidemiology applications that model local and global disease outbreaks; tornado simulations where forecasters can identify conditions that make a tornado likely, they can pinpoint when and where they start, their path, and strength.

All modules in our analysis use events generated by the system in their process. For the Blue Water systems, several events are gathered into different logs summing up to more than 15GB of data per day (on peak days the number exceeds 120GB). By the time of this paper, we did not have access to environmental data so all our modules use only log events. There are 5 major sources used: syslogs that contain usual RAS information, HPSS (High Performance Storage System) which is the near-line storage designed for moving large files and large amounts of data, Sonexion storage system used for storing the Luster filesystem, Moab job scheduler and ESMS, the data system manager. Table 1 presents these sources and their characteristics compared to some smaller systems.

The number of events generated by the Blue Waters system is two order of magnitude larger than Blue Gene/L. Combined with the complexity of the analysis modules it is no longer feasible to run the training phase of the online methodology to get updates periodically. Moreover, the increase number of types of events, create complex patterns that need longer training phases to be discovered. Figure 6 shows the results of applying the same prediction algorithm used for Blue Gene/L on the Blue Waters system. Overall, both the recall and precision values are significantly lower than for the Blue Gene/L system. In the next section we will analyze the reason for this decrease in both recall and precision values.

Information about failures is kept into a distinct failure log where system administrators write down the approximate timestamp for each failure and the possible cause. We analyzed 5 months of activity before the new cabinets were added and a couple more after the system was expanded. The second phase of the analysis is done before the system had a chance of becoming stable and so we were able to compare the results for the two different phases of the machine. The first 5 months were divided into three parts: 2 months for training, 2 for the simulate online part and 1 month for online testing. For the second analyzed period we used only one month for training, 3 weeks for simulate online and 1 week for online testing.

Table 1. Frequency of Special Characters

Source	Events/Day	Total Event Types
Syslog	8GB (50mil events)	3,852
HPSS	1MB (900,000 events)	358
Sonexion	3.5GB (10mil events)	3,112
Moab	500 MB (15mil events)	725
ESMS	3GB (12mil events)	2,452
System	Events/Day	Total Event Types
BlueGene/L	5.76MB (25,000 events)	186
BlueGene/P	8.12MB (120,000 events)	252
Mercury	152.4MB (1.5mil events)	563

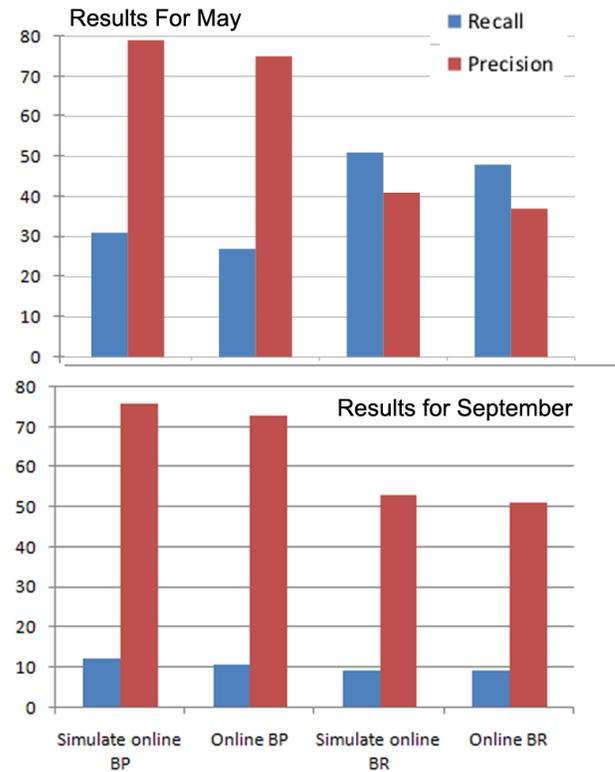


Figure 6. Precision and recall for the Blue Waters

Our first analysis on the failures gave us a general view of the entire system. We observed that software errors are 65.7% of all problems and hardware 34.3%. For software failures, the most representative were related to the Luster filesystem. For hardware, AMD Opteron problems and DIMM failures have the majority. Moreover, 13.5% types of the failures affect more than one node, which translates in more than 25% of total failures from the analyzed time frame.

4.2 Prediction from the system's perspective

Because of the high number of events in the system and the stress they put on the analysis modules, we were forced to change the offline training methodology to make it more light. We transformed the correlation extraction process in a hierarchical one, starting with correlations between only failures and recursively adding new events in the analysis. This way we are extracting patterns only for events that might lead to failures and ignore the rest. Also, in the initial step we no longer divide the failures separately into their

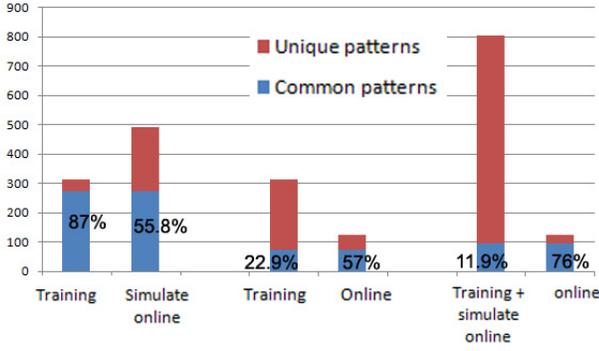


Figure 7. Number of common correlation patterns between the training, simulate online and online phases

corresponding types (and so extracting patterns independently for each type). We make a breakdown on failure types only during the online phase when we are recording the true positives.

Figure 6 shows the precision and recall obtained for the Blue Waters system for both manual tuning of the parameters and the online methodology. The experiments were made twice by focusing first on obtaining the best possible precision and the second for best recall. The results are much lower than for the Blue Gene/L ones. However, when looking just at the results for Blue Waters, the manual and automatic methods both give similar results which shows that the online methodology is a valid strategy on larger systems as well. For the second analysis period the recall value is much lower even than before. A closer analysis of the patterns used showed that more than 60% of the failures from the online phase were not encountered in either the training nor the simulate online phase. It seems that the constant changes that happen in a system after a recent upgrade make it very difficult for the prediction modules to learn patterns and adapt. For the rest of the paper we focused out attention on the first 5 months of analysis.

In the next paragraphs we will open the hood and look at how this results are obtained. We used ELSA's trainig phase on all log parts: the training, the simulate online and the online testing parts and then compared the patterns extracted for each. Figure 7A shows the quantity of patterns found in both training and simulate online (the bottom part of each bar) and the number of patterns that are encountered in one phase but not in the other (the top part of each bar). Figure 7B shows the same metrics but for the training and online phases and figure 7C looks at the pattern count when combining the first two phases together compared to the online phase. All results in this figure are obtained in the experiments focused on obtaining the best recall.

The patterns from the training phase that appear also in the simulate online phase are the only ones that are optimized to give best results on the simulate online phase. The rest of the patterns even though might be used in the testing phase, are left in their default state. We believe this is the reason for the small difference between the manual and automatic results. Another interesting find is the fact that only 55.8% of patterns found by the training method on the online phase appear in the training phase. Longer training phase in the prediction methodology would help increase this percentage, as shown in figure 7C where 76% of patterns from the online phase are covered by including the simulate online in the training phase. Of course the actual recall values depend on how frequent each pattern is used, but larger coverage should give better recall values in all cases.

In our second experiments we used the entire log for all three stages: we training the method on the entire log, run a simulate

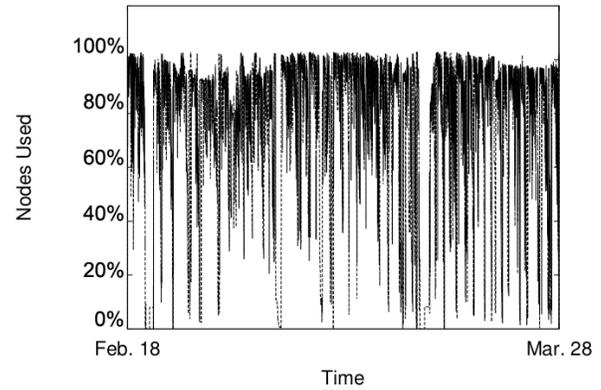


Figure 8. Blue Waters usage

online phase and automatically tune the parameters in the training phase until we get the best results, then run the online phase on the same log. In the ideal case we would expect to get 100% recall and precision, assuming all failures have precursors in the log and they appear frequent enough to pass the correlation thresholds. The results show a recall of 67.2% with a precision of 78% when looking at best recall. The values are lower than expected either because some failure patterns were not frequent enough to become correlation chains for our model, in the analyzed months, or simply because not all failures have precursors. This experiment gives us a higher bound for our predictor in the sense that we cannot expect to obtain better results than 67.2% with any methodology.

Another experiment shows that the location prediction used before for extracting patterns is not accurate enough for a system of Blue Waters' size and complexity. Over 90% of our predictions for multi-node failures do not succeed in discovering all the nodes in the fail set. A more detailed inspection reveals that the benefit of a good location propagation could increase the recall with more than 10%. We decided to implement a couple of novel methodologies for extracting propagation patterns for correlation chains and failures and test their results. The next section of the paper presents the details of this study.

There are many untracked or untrackable variables that can play an important role in the correlation of performance analysis. For example, network monitoring is crucial to understanding the health of a supercomputer, but is barely on the radar of hardware manufacturers. The last section of the paper deals with analyzing the recall for different type of failures which will make the lack of precursors more visible.

4.3 Prediction from the application's perspective

Figure 8 shows the usage of the system for the analyzed period. In general, we would expect higher failure rates with heavier workload. However, our first results does not show any correlation between jobs exited with an error code and average load of the system. Moreover, a failure in the system most of the time does not seem to impacted massive numbers of jobs. At a closer analysis, we observed that only around 44% of failures lead to at least one application crash. The same analysis shows that 62% of the failure types predicted by ELSA refer to failures that lead to application crashes. This corresponds to an increase in the recall of over 5% when we filter out from the analysis all failures that have no effect on any of the running applications.

When analyzing the prediction results from the application’s perspective, in addition to the recall and precision, the online methodology is highly sensitive to the lead time offered by each prediction. The lead time represents the time interval between when the prediction is triggered and when the failure actually occurs. Normally, during faulty moments, there is a spike in the number of notifications in the logs to over 20,000 messages per second. After a high number of optimizations, our analysis modules manage to cope with 500 messages per second. During these peak times the analysis slowly falls behind having lags of over 30 minutes for some of the more aggressive failures. As long as the precursor events appear before this peak of messages the prediction module will not be influenced by the failure’s behavior. However, over 80% percentage of a failure’s precursors appear in the system after the notifications have started to peak. If we define true positives so that they considered the lead time, the recall now only considers failures for which the prediction is done before its occurrence. This decreases the total recall value with 8%.

Another change in the methodology of extracting results is that location prediction get a slightly new meaning when application crashes need to be predicted instead of pure failure messages. If the prediction method predicts a failure correctly in time, but the predicted location is wrong and so the failure actually occurs on a different node, with our previous method this will give a false negative and a false positive in the final results. However, if an application was running on both nodes, the predicted one and the one that will fail, if the application takes into consideration the prediction and takes preventive actions, the failure could cause minimal damage. Depending on the fault avoidance strategy, a predictor that only looks at applications as a whole and not as a set of running nodes could increase the recall significantly.

By taking the lead time and the new definition of location prediction into consideration we recomputed the results for best possible recall and obtained 35% recall and 75% precision.

5. Location prediction

One of the main problems brought by the complexity of Blue Waters is the fact that the patterns between precursor events and failures are now more complicated and often precursors from one node location indicate a failure on a completely different node. Moreover, some problems, like the majority of Luster failures, although they have precursors on the same set of nodes, they influence applications running on other sets of compute nodes.

In this section we will analyze the relation between the location of precursors and their corresponding failures. We will also look at how the same failure propagates on multiple nodes and how this influence the applications running on the machine. In this paper we will focus on two parts of this architecture since they cover the large majority of events: the compute nodes and Sonexion nodes.

On Blue Waters the locations for compute nodes have the following format: c0-0c0s0n0 which represents the cabinet id, cage id, slot id and node id; and for the Sonexion nodes snx11003n0 we consider cabinet id and node id. Our first attempt in finding patterns between different locations used only the location id without any information as to what is the network of the system. We investigated for each of the correlation chains that contain events appearing on multiple location if the are propagating beyond the same node/slot etc. If, for example a precursor and its error are always on the same slot but not necessary on the same node, the prediction engine could use this information and predict that the whole slot will fail.

The new methodology is called a hierarchical prediction of locations and it basically over-predicts the number of nodes that fail in order to increase the true positive base. We implemented

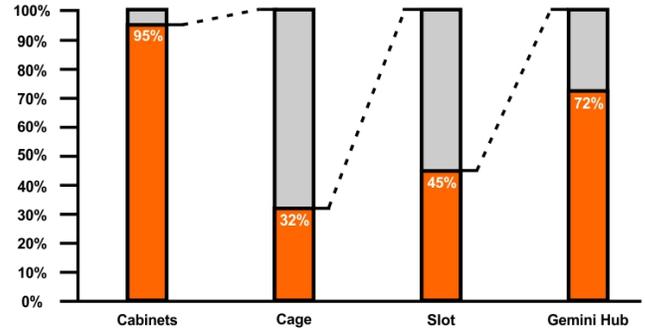


Figure 9. Location propagation results

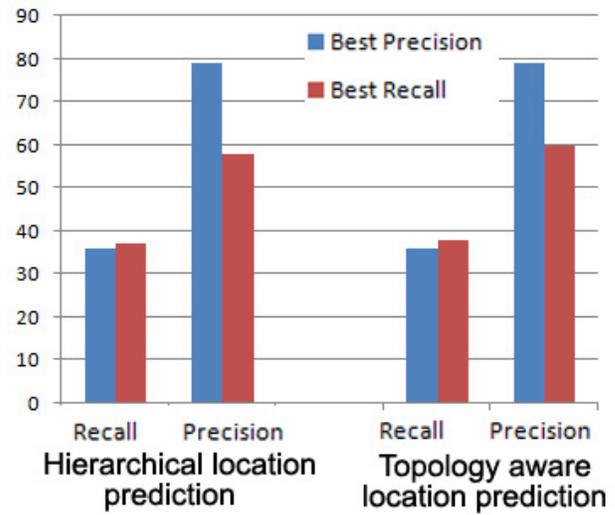


Figure 10. Precision and recall for different location predictors

this methodology in ELSA and after running it on Blue Waters we obtained the results from figure 9.

We observed that 95% of the correlation chains contain events that appear in the same cabinet, 32% appear in the same cage (from the same cabinet), 45% in the same slot and 72% on the same node. A slot has 4 nodes, a cage has 8 slots (32 nodes) and a cabinet has 3 cages (96 nodes). In total there are 276 cabinets. The prediction module is using the hierarchical location method by predicting different location classes (node level, slot level, etc.) depending on the chains used to trigger the future failure event. A true positive in this scenario represents a prediction for which, firstly, the failure occurred and also the set of locations affected by the failures is a subset of the predicted locations.

Figure 10A shows the precision and recall values when using the hierarchical strategy compared to our initial results. The results show an increase in recall from 28% with a precision of 72% to 36% with 79%. When looking for the best possible precision the increase in both recall and precision corresponds to the percentage of unpredicted failures because of the wrong location choice. This indicates that the majority of the correlations for which precursors are not on the same location as the failures they predict, are captured correct. Hierarchy prediction overcomes this problem by widening the set of locations used for prediction when patterns are not strong enough.

The main advantage of this method is that now ELSA captures a larger set of the failures with a higher precision. However, when predicting a cabinet failure, not necessary all 96 nodes from the cabinet will actually fail. We made an analysis on the online testing phase and we observed that out of all true positives: 26% of predictions where slot prediction and 72.2% of the times only one node was affected and only two nodes in the rest of the cases; 8.7% of cases were cage predictions with an average of 3 node failures; and 20% were cabinet failures with an average of 4 node failures in 81% of the cases and full cabinet failures in the rest.

For incorrect predictions, 20% were slot prediction, 3.4% were cage predictions and 6.6% were cabinet predictions. If the usage of the system is 100%, the hierarchical location prediction adds waste with both true positives and false negatives by forcing applications that are running on nodes that will not fail to take preventive actions.

Before analyzing into detail the waste, we focus on optimizing our location prediction so that the waste is decreased. Our main problem is that the method is predicting at multiple node level for a failure that occurs on only a small number of nodes. This happens only because our pattern extraction algorithm could not find patterns at smaller levels (node compared to slot, slot instead of cage and so on) by just examining the location id. In order to find better patterns, we will take the analysis a step further by looking at the network topology and how location ids map on the network.

5.1 Network topology

In Blue Waters compute nodes are connected by a 23x24x24 3D torus network and the Sonexion nodes in a fat tree connecting to the compute nodes through Infiniband.

We mapped the location ids of each compute node on the torus network and looked at patterns at this level (rather than patterns in the location ids). When there are no patterns the algorithm will still use the hierarchical prediction method described above. In general, mapping location ids on the torus on Cray systems follows the algorithm described in figure 11. In the figure every cube in the torus represents a Gemini hub. Consecutive nodes in the same slot are connected to the same Gemini hub so in the figure every cube represents two consecutive nodes (nodes 0 and 1 and node 2 and 3 from every slot). Two neighbour Gemini hubs on the OY axes create a slot, multiple consecutive slots on the OZ axes form a cage (in the figure two slots create a cage) and multiple consecutive cages on the OX axes create a cabinet (in the figure two cages create a cabinet). The cabinets are divided in two dimensions, first on the YOZ plane and then on the OX. For the Blue Waters system, there are 2 Gemini hubs in each slot, 8 slots form a cage and 3 cages create a cabinet. Each cabinet is as wide as the OZ portion of the torus, so the entire Gemini hub set is divided into 23 cabinets on the OX axes and 12 cabinets on the OY like in figure 12.

The algorithm used for extracting patterns in the torus looks at both relations between precursors in a chain and the predicted failure but also at location propagation patterns for the same failure. For this paper we implemented a simple neighbour regression algorithm that looks only for mathematical relation between nodes on each of the three axes. Basically after eliminating outlier nodes, the algorithm extracts a cube of minimal dimension that captures all the active nodes (nodes where failures are occurred). For the future we plan to investigate more complex spatial data mining techniques.

Figure 10B presents the results of using the extracted patterns with the new method for prediction on the online testing part of the log. The new method has similar recall and precision values as with the hierarchical method from the previous section. The main advantage of this method is that it reduces the number of times we over-estimate the number of nodes that might fail. The method uses prediction at cabinet/cage/slot level less frequent. The analysis on

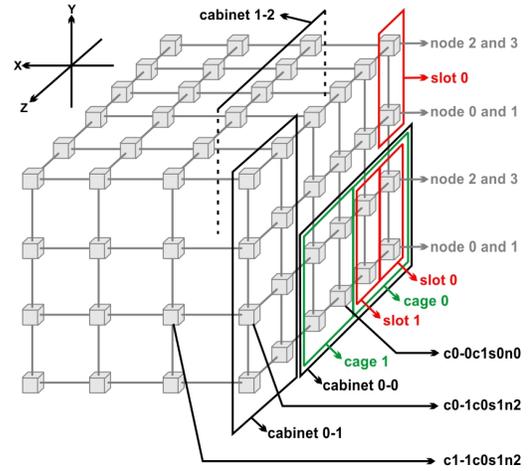


Figure 11. Failure prediction: simulate online

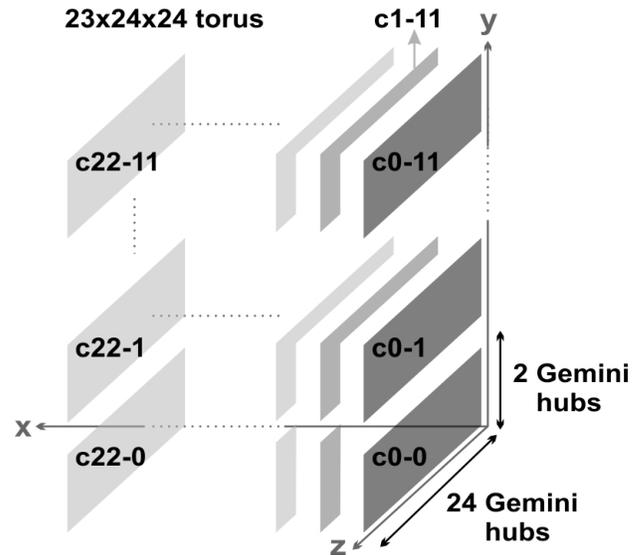


Figure 12. Failure prediction: simulate online

the online testing phase shows that out of all true positives: 21% of predictions where slot prediction with only one or two nodes failing, 2.3% were cage predictions with the same average of 3 node failures; and 12.5% were cabinet failures with an average of 4 node failures. For incorrect predictions the numbers decreased as well having only 18% slot predictions, 2% cage predictions and 2% cabinet predictions.

Assuming a 100% system utilization the percentage of nodes that do not have to take useless preventive actions because of our hierarchical prediction decreases with the new method by 15%.

6. Overall results

Until now failures did not have a type and were all analyzed together into two metrics: precision and recall. Since the results are lower than what smaller systems present, we broke down the predicted failures into their type. Table 2 presents the most frequent failures, and the ones with the best and worst recall from the ana-

lyzed timeframe. The examples cover cases when the failures are never seen by our prediction module, when there are limitations in the extracted correlation chains or location propagation patterns, and also cases when the prediction works well. We will discuss each case in the following paragraphs.

In general, we observed that hardware failures can cause software failures, software failures can cause other software failures and that software fault tolerance overall is poor. There are cases when one hardware failure can in practice bring down or cripple the performance of the entire machine. After discussions with system administrators at NCSA we observed that many of the major outages in Blue Waters have been caused by environmental issues and could have been avoided. As the system progresses we expect to see more stability in the correlation patterns extracted by ELSA and so we expect to see higher values for the recall and precision.

Table 2 present the precision and recall of a few failure types when considering all failures occurrences or only the ones that crash applications. The results show that the recall and precision depend greatly on the failure type. For example, the Luster Metadata failures have very few precursors and since most of them occur at the same time with the actual failure our method disregards the majority. Metadata servers for the Luster filesystem store namespace metadata, such as filenames, directories, access permissions, and file layout. When applications detect an MDT failure, they connect to the the backup MDT and continue their execution. Just in some cases, applications having trouble connecting to the back-up MDT fail so in general it is expected to have a higher value for precision and recall when looking from the application's point of view. This is not the case, however, because of location prediction.

The failures concerning the Luster Object Storage Targets behave in a very similar matter with the difference that there is a slightly larger set of precursors. During an OST failure, when applications attempts to do I/O to a failed Lustre target, these are blocked waiting for OST recovery or failover. An application does not detect anything unusual, except that the I/O may take longer to complete. Rarely, when an OST is marked as inactive, the file operations that involve the failed OST will return an IO error and the application might be finished. The results when computing the recall just for the failures that crash an application are similar as to the MDT failures. Interestingly, the number of applications crashed because of OST are higher than because of the metadata server failovers even though their raw number is lower. We plan to study this into more detail in the future.

Luster MDTs and OSTs are stored on the Sonexion storage system and so they use different location ids than the compute nodes. The Sonexion nodes are using a fat tree network and communicate with the compute node through Infiniband. When extracting the location patterns we only used ids for the Sonexion nodes and so far the correlations chains that have Luster information use only Sonexion nodes. This means that both precursors and failures use these nodes. This is enough to predict some of the OST failures. However, since applications are running on compute nodes, the prediction does not have enough information to manage to predict the exact applications that might suffer from a Luster failover. Information about what files are used by an application is necessary in order to manage to predict application crashes. This explains why ELSA was unable to predict any of the application failures caused by Luster failovers. We encountered similar problems with Moab failures as well.

DIMM failures are one of the most frequent hardware failures in the Blue Waters system. There are a couple of accurate chains extracted for this type of failure that give large lead times and that were used in the online testing phase. Moreover, when a pair of DIMMs is disabled due to faults, in some cases other DIMMS are disabled in order to maintain a valid configuration. Even if the first

failure is not predicted, subquential DIMM failures are captured. Our results on memory failures, in general, are similar to the ones observed on the Blue Gene/L system in our previous studies [9].

Compute Blade failures represent less than 3% of total failures. However, the patterns extracted for them have a high confidence and lead times of more than one hour. Our previous experience with Blue Gene/L shows similarities, with node card failures being the type with the best results. The recall and precision values for Blue Waters are smaller. However, looking at the patters needed to be used for prediction we believe the numbers would increase with a larger training phase. Another example are Gemini failures which are successfully predicted and for which the location prediction works flawless.

Since Sonexion and Moab failures represent the majority of problems on the Blue Waters system, we believe that increasing the results for this type would benefit the overall prediction. For this we filtered out all failures related to Luster and the scheduler and redid the analysis focusing on best recall. The results show a recall of 48% with a precision of 63%. The coverage in this case is close to the 62% maximum targeted recall which suggests that the main problem of our predictor are Luster and Moab failovers. We will work in the future in finding better precursors for these types.

Overall, the conclusion of our detailed analysis is that we need better precursors in order to increase our results on the Blue Waters, either from the system level or at the application level. For example, monitoring the I/O patterns of an application could help with location prediction for Luster errors which at this point represents the majority of the software failures. The best result obtained shows a recall of 62% with a 85% precision and has good location forecast for both predicting all failures and predicting failures that cause application crashes. This category of failures would greatly benefit from a larger training phase by achieving results very similar to the ones obtained on smaller systems.

7. Conclusion

Failure prediction has made substantial progress in the last 5 years and current studies have shown that failure avoidance techniques could give high benefits when combined with classical fault tolerance protocols. Understanding the properties of a prediction module and exploiting them for enhancing fault tolerance approaches and scheduling decisions is crucial for providing scalable solutions to deal with failures on future HPC systems. In this paper, we proposed a methodology for online failure predictions and tested it on previous generation large scale system and on the Blue Waters system. We described the problems and limitations faced in applying failure prediction on a petascale system and proposed a couple of solutions that will help evolve our predictor into a viable solution for failure avoidance approaches. The results obtained are greatly dependent on the type of failures analyzed with some results showing that our method can predict over 60% of occurrences of a certain type and other not being able to forecast more than 10%. Moreover, the analysis becomes even more complex when looking from the application's point of view increasing the recall in some cases or making the prediction almost impossible without context information in others. For the future, we plan to focus on analyzing the difference between all types of events and improve the results of our predictor by inspecting different precursor detectors, especially for Luster and Moab failures, to include in ELSA. Also, we will study to a wider extent, the impact of combining our predictor with various fault tolerance protocols over the execution time of applications.

Table 2. Frequency of Special Characters

Failure type	Percentage	Recall	Application Crashes	Application Crash Recall
Luster MDT Failure	39.6%	7%	5%	0%
Luster OST Failure	16.3%	15%	13%	0%
DIMM Failure	15.7%	38%	11%	58%
Compute Blade	2.9%	62%	21%	64%
PBS Out-of memory	3.6%	44%	0%	0%

Acknowledgments

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. This research was done in the context of the INRIA-Illinois Joint Laboratory for Petascale Computing. The work was also supported by the U.S. Department of Energy, Office of Science, under Contract No. DE-AC02-06CH11357.

References

- [1] Inter-Agency Workshop on HPC Resilience at Extreme Scale. <http://institute.lanl.gov/resilience/docs/Inter-AgencyResilienceReport.pdf>, 2012. [Accessed on July 2013].
- [2] U.S. Department of Energy Fault Management Workshop. <http://shadow.dyndns.info/publications/geist12department.pdf>, 2012. [Accessed on July 2013].
- [3] S. B. A.F. Patra and U. Kumar. Failure Prediction of Rail Considering Rolling Contact Fatigue. *International Journal of Reliability, Quality and Safety Engineering*, 3, 2010.
- [4] S. U. C. P. B. M. Z. Agarwal, M. and S. Mitra. Circuit Failure Prediction and Its Application to Transistor Aging. *The 25th IEEE VLSI Test Symposium*, pages 277–286, May 2007.
- [5] A. G. Evans and S. M. Wiederhorn. Crack propagation and failure prediction in silicon nitride at elevated temperatures. *Journal of Materials Science*, 9:270–278, February 1974.
- [6] E. W. Fulp, G. A. Fink, and J. N. Haack. Predicting computer system failures using support vector machines. In *WASL*, 2008.
- [7] A. Gainaru, F. Cappello, S. Trausan-Matu, and B. Kramer. Event log mining tool for large scale hpc systems. In *Proceedings of the 17th international conference on Parallel processing - Volume Part I, Euro-Par'11*, pages 52–64, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23399-9. URL <http://dl.acm.org/citation.cfm?id=2033345.2033352>.
- [8] A. Gainaru, F. Cappello, and W. Kramer. Taming of the shrew: Modeling the normal and faulty behavior of large-scale hpc systems. In *Proceedings of IEEE IPDPS 2012*. IEEE press, 2012.
- [9] A. Gainaru, F. Cappello, M. Snir, and W. Kramer. Fault prediction under the microscope: A closer look into hpc systems. In *Proceedings of 2012 International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE press, 2012.
- [10] G. Hamerly and C. Elkan. Bayesian approaches to failure prediction for disk drives. 2001.
- [11] E. Heien, D. Kondo, A. Gainaru, D. LaPine, B. Kramer, and F. Cappello. Modeling and tolerating heterogeneous failures in large parallel systems. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 45. ACM, 2011.
- [12] A. A. Hwang, I. A. Stefanovici, and B. Schroeder. Cosmic rays don't strike twice: understanding the nature of dram errors and the implications for system design. *SIGARCH Comput. Archit. News*, 40(1):111–122, Mar. 2012. ISSN 0163-5964. URL <http://doi.acm.org/10.1145/2189750.2150989>.
- [13] W. M. Jones, J. T. Daly, and N. DeBardeleben. Application monitoring and checkpointing in HPC: looking towards exascale systems. *Proceedings of the 50th Annual Southeast Regional Conference*, pages 262–267, 2012.
- [14] Z. L. S. C. L. Yu, Z. Zheng. Practical Online Failure Prediction for Blue Gene/P: Period-based vs Event-driven. *IEEE Conference on Dependable Systems and Networks Workshops*, pages 259–264, 2011.
- [15] J. Lou. Mining dependency in distributed systems through unstructured logs analysis. *ACM The Special Interest Group on Operating Systems (SIGOPS)*, 44, 2010.
- [16] N. T. T. K. Y. K. H. Mitsunaga, Y. and Y. Ishida. Failure prediction for long length optical fiber based on proof testing. *Journal of Applied Physics*, 53, July 1982.
- [17] T. Muthumani, N. and A. Selvadass. Optimizing Hidden Markov Model for Failure Prediction - Comparison of Gaijes optimization and Minimum message length Estimator. *International Journal on Computer Science and Engineering*, 3, 2011.
- [18] N. E. E. H. N. Bolander, H. Qiu and T. Rosenfeld. Physics-based Remaining Useful Life Predictions for Aircraft Engine Bearing Prognosis. *Conference of the Prognostics and Health Management Society*, 2009.
- [19] NASA. Rocket engine failure prediction using an average signal power technique. <http://engineer.jpl.nasa.gov/practices.html> (accessed on August 2013), 1994.
- [20] Z. Z. Qiang Guan and S. Fu. Ensemble of bayesian predictors for autonomic failure management in cloud computing. *20th International Conference on Computer Communications and Networks*, pages 1–6, 2011.
- [21] J. L. H. S. M. R. Vilalta, C. V. Apte and S. M. Weiss. Predictive algorithms in the management of computer systems. *IBM Systems Journal*, 41:461–474, 2002.
- [22] R. Sahoo, M. Squillante, A. Sivasubramaniam, and Y. Zhang. Failure data analysis of a large-scale heterogeneous server environment. In *International Conference on Dependable Systems and Networks*, pages 772–781, 2004.
- [23] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, and S. Ma. Critical event prediction for proactive management in large-scale computer clusters. In *In KDD*, pages 426–435. ACM Press, 2003.
- [24] F. Salfner, M. Lenk, and M. Malek. A survey of online failure prediction methods. *Computing Surveys*, 42:1–42, 2010. .
- [25] S. V. Sasisekharan, R. and S. Weiss. Data mining and forecasting in large-scale telecommunication networks. *IEEE Expert*, 11:37–43, 1996.
- [26] M. Snir, W. Gropp, and P. Kogge. Exascale Research: Preparing for the Post-Moore Era. *Computer Science Whitepapers*.
- [27] Z. Z. R. T. S. C. Z. Lan, J. Gu. Dynamic meta-learning for failure prediction in large-scale systems: A case study. *Journal of Parallel and Distributed Computing*, 6:630–643, 2010.
- [28] Z. Zheng and L. e. a. Yu. Co-analysis of ras log and job log on blue gene/p. *Proceedings of the 2011 IEEE International Parallel and Distributed Processing Symposium*, pages 840–851, 2011.
- [29] Z. Zheng, Y. Li, and Z. Lan. Anomaly Localization in Large-Scale Clusters. *IEEE International Conference on Cluster Computing*, pages 322–330, 2007.

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne").

Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.