Exploring Event Correlation for Failure Prediction in Coalitions of Clusters

Song Fu and Cheng-Zhong Xu
Department of Electrical and Computer Engineering
Wayne State University
Detroit, MI 48202
{song, czxu}@eng.wayne.edu

ABSTRACT

In large-scale networked computing systems, component failures become norms instead of exceptions. Failure prediction is a crucial technique for self-managing resource burdens. Failure events in coalition systems exhibit strong correlations in time and space domain. In this paper, we develop a spherical covariance model with an adjustable timescale parameter to quantify the temporal correlation and a stochastic model to describe spatial correlation. We further utilize the information of application allocation to discover more correlations among failure instances. We cluster failure events based on their correlations and predict their future occurrences. We implemented a failure prediction framework, called PREdictor of Failure Events Correlated Temporal-Spatially (hPREFECTS), which explores correlations among failures and forecasts the time-between-failure of future instances. We evaluate the performance of hPREFECTS in both offline prediction of failure by using the Los Alamos HPC traces and online prediction in an institute-wide clusters coalition environment. Experimental results show the system achieves more than 76% accuracy in offline prediction and more than 70% accuracy in online prediction during the time from May 2006 to April 2007.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Measurement, Algorithms, Experimentation, Performance.

Keywords

Failure prediction, Coalition clusters, Temporal correlation, Spatial correlation, System availability.

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. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SC'07, November 10-16, 2007, Reno, Nevada, USA
Copyright 2007 ACM 978-1-59593-764-3/07/0011 ...$5.00.

1 Introduction

Networked computing systems continue to grow in scale and in the complexity of their components and interactions. In these systems, component failures become norms instead of exceptions. For example, a recent system reliability study on a 512-node LLNL ASC White machine showed that the mean time to failure of a node was about 160 days [31]. If the same failure model is applied to the largest BlueGene/L machine, there would be more than 17 node failures per hour. As a result, a long running job on a large number of nodes may find it difficult to make progress due to the frequent failures. It was because of the system availability concern, BlueGene/L in LLNL had to disable L1 cache in each node when jobs larger than 4 hours was running because the cache was found prone to failure.

Checkpointing is a conventional approach for fault tolerance. Because checkpointing a job in a large-scale system could incur overhead as high as more than half an hour execution time, frequent periodic checkpointing often prove counter-effective. As the scale and complexity of high performance computing (HPC) systems continue to grow, research on failure management has recently shifted onto failure prediction and related proactive autonomic management technologies [29, 23, 41, 25, 22, 27, 13, 11, 10]. Failure prediction is a crucial technique for understanding emergent, system-wide phenomena and self-managing resource burdens. Based on the analysis of failure data in a system, a failure predictor aims to determine possible occurrences of fatal events in the near future and help develop more effective failure tolerant solutions for improving system availability.

To predict the trend in failure occurrence, we need an in-depth understanding of the cause of failures and their empirical and statistical properties. Past studies on component failures in production systems, such as IBM BlueGene/L supercomputer [23] and Los Alamos National Laboratory (LANL) HPC clusters [5, 29], revealed important patterns in failure distribution. Although the time-between-failure is highly non-linear, there exists the time-of-day and day-of-week patterns in long time spans [29, 28]. Temporal correlation aside, failure events, depending on their types, display strong spatial correlations: a small fraction of nodes may experience most of the failures in a coalition system [28] and multiple nodes may fail almost simultaneously [23]. These temporal and spatial correlation properties of failure events revealed by offline profiling provide important information for predicting the trend of failure dynamics.

There were recent works utilizing temporal and/or spatial correlations of failures for failure prediction and proactive management; see [27, 22, 31] for examples. They explored the temporal correlation via profiling the time-between-failure of different

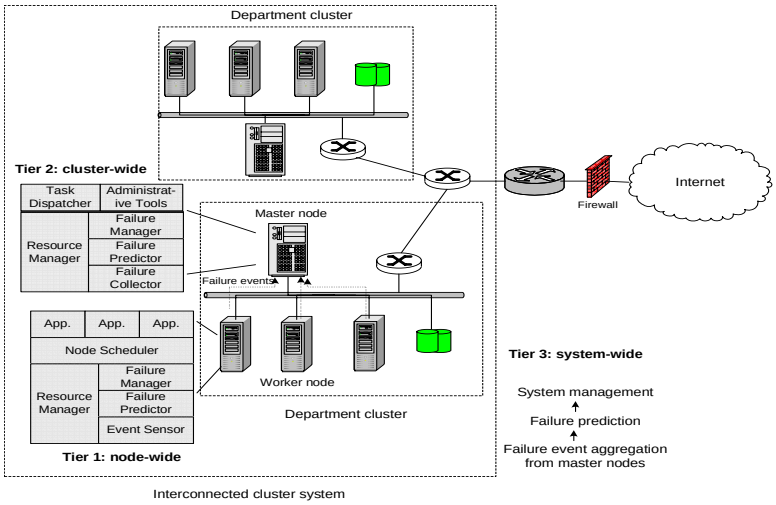


Figure 1: hPREFECTS: a hierarchical failure prediction system for a coalition system.

failure types. Liang *et al.* [22] also considered spatial correlation among failures. They found skewness in the distribution of network failures among the BlueGene midplanes.

We notice that most of today's failure prediction approaches are heavily empirical, applying heuristics to explore temporal and spatial correlation of failures based on profiling. There lack formal models to quantify the temporal correlation among failures in different timescales. A cluster coalition system is hierarchical in structure and failures may occur in multiple scopes: node, cluster and system. There are no models to quantify the spatial correlation of failures for predicting their future distribution and locations in different scopes. Because the co-existence of both spatial and temporal correlation may lead to failure propagation from node to node at different times, the models should also be able to facilitate the study of failure propagation and its impact on prediction accuracy. It is known that there is dependency between the workload (its type and intensity) and the failure rate [29, 20, 28]. However, there are few work on the impact of application allocation on failure prediction, either.

In this paper, we exploit both temporal and spatial correlations for failure prediction in coalition systems. We develop a covariance model with an adjustable timescale to quantify the temporal correlation and a stochastic model to describe spatial correlation. We utilize information of application allocation in a coalition system to discover more correlations among failure instances. Further, we investigate and model the failure propagation resulted from both temporal and spatial correlations. We cluster failure events in a coalition system based on their correlations and predict their future occurrences. We summarize our contributions as follows:

1. We define a *failure signature* representation to capture the system performance metrics associated with a failure event. It's effective for clustering and analyzing the temporal and spatial correlations among failure events. The construction of an effective signature requires to consider the hierarchical structure and interactions among components at different scopes of a coalition system.

2. We develop a spherical covariance model with an adjustable timescale parameter to quantify temporal correlation among failure events. We use the distance in time between two failures to calculate their covariance value, which specifies the extent of their correlation. The timescale in calculating the covariances is adjustable for different types of failures.

3. We propose a time-efficient aggregate stochastic model to quantify spatial correlations. We use the probabilistic distribution of failures to compute the spatial covariance among failures. We model the failure propagation phenomena by investigating failure correlations in both time and space domains.

4. We utilize the application allocation information to refine the possible correlations among failure occurrences, based on our observation in a real coalition system: 71.5% application I/O failures are clustered in space and their locations are determined by job scheduling decision.

5. We implemented a failure prediction framework, called PREdictor of Failure Events Correlated Temp-Spatially (hPREFECTS), which explores correlations among failures and forecasts the time-between-failure of future instances. We evaluate the performance of hPREFECTS node-wide and system-wide in both offline prediction of failure by using the LANL HPC traces and online prediction in an institute-wide computational grid.

A prototype of hPREFECTS, has been in operation since May 2006 on a production coalition environment: the Wayne State University Computational Grid (WSU Grid) [6]. The grid consists of three clusters located in three campus buildings and contains 40 high-performance compute servers in support of university-wide high-performance computing application programs. Online failure predictions were performed with observed failures and on production traces from more than one and a half years of operations. The prediction results show our prediction system can forecast the occurrence time of future failures with more than 70% of accuracy. We also evaluated the performance of hPREFECTS using the trace from the LANL HPC coalition system. Offline evaluation results show that we can forecast of the failure occurrences with higher than 76% accuracy, and predict the occurrences of failures on individual nodes with Bayesian networks.

The rest of this paper is organized as follows: Section 2 describes the framework of hPREFECTS and its prediction methodology. Section 3 presents the algorithms to cluster failure signatures based on their spatial and temporal correlations. Offline

Table 1: Variables characterizing failure dynamics.

Variable	Description
fID	Failure identification number
$fLoct$	Location of a failure including cluster ID and node ID
$fType$	Classification of a failure based on its cause
time	Timestamp when a failure occurs
tbf	Time between successive failures in node, cluster or system
$fCount$	Number of failures in node, cluster or system for a time window
usrUtil	Percentage of CPU utilization that occurred while executing at the user level in a node
sysUtil	Percentage of CPU utilization that occurred while executing at the system level in a node
$frmUtil$	System frame utilization in a node
pktCount	Number of packets transmitted and received by a node for a time window
ioCount	Number of I/O requests to the physical disks of a node for a time window
alloc	Allocation information of nodes to application jobs
sptCorr	Spatial correlation among failures in cluster or system
tmpCorr	Temporal correlation among failures in node, cluster or system

prediction by hPREFECTS using the LANL HPC trace is evaluated in Section 4. Section 5 discusses the performance of online prediction in the WSU Grid. Section 6 presents the related work and Section 7 summarizes the paper.

2 Failure Prediction Architecture and Methodology

In this section, we address two key issues in failure prediction: (a) How do we design the architecture of failure prediction subsystem that explores failure dynamics in a coalition? (b) What representation should we use to describe failure instances and the associated system performance variables? An additional issue is how we cluster failure signatures to identify temporal and spatial correlations among failure occurrences. We leave this issue to the next section.

Without loss of generality, when we refer to a "failure" in the following discussion, we mean any anomaly caused by hardware or software defect, incorrect design, unstable environment or operator mistakes that makes services or compute nodes unavailable. We also analyze the properties of particular types of failures in Section 4.

2.1 Hierarchical Failure Prediction Framework

To analyze the correlations of failure instances in different scopes of a coalition system, we design a failure prediction framework with a multi-layer prediction architecture. Failure events along with the associated performance variables are described by a formal representation, which allow us to cluster correlated failures and to utilize this correlation information for prediction.

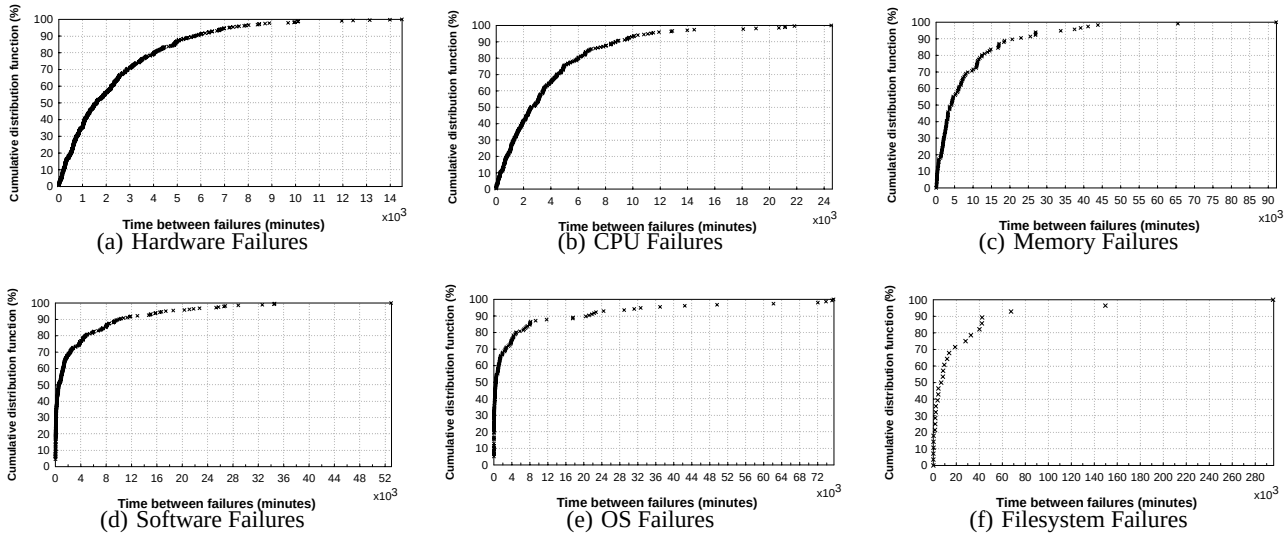
Figure 1 depicts the architecture of hPREFECTS. Failure prediction is invoked on a compute node or a master node upon its job scheduler receiving a job submission from a user. In compute node wide, *event sensor* keeps track of the new events recorded to the local event logs since its last operation, extracts failure records and creates formatted failure reports for the *failure predictor*. It also monitors the performance dynamics of executing applications and measures the resource utilization. Upon invocation, the failure predictor calculates the estimates of future failure occurrences based on information collected by the event sensor. To filter out scheduled events, such as system maintenance operations, from failure reports, the failure predictor extracts records of these scheduled events from received system activity reports maintained and sent by the master node. With the prediction results, *failure manager* generates failure alarms and system availability reports; and finally updates the system profiles. It also sends a copy of the nodal failure report to the master node for cluster-wide failure prediction and resource allocation.

In cluster wide, the master node collects failure reports from its managed compute nodes; statistically processes and analyzes the failure events; predicts prospective failures; and generates system availability reports for the resource scheduler and system administrator. Cluster-wide failure predictor analyzes spatially and temporally correlated failures and estimates cluster availability based on predicted failure dynamics. System-wide failure predictor receives failure reports from master nodes of clusters and forecasts the failure dynamics of the entire coalition environment for system management.

2.2 Failure Signatures

An important issue we address is that of extracting from a running system an indexable representation that distills the essential characteristic from a system state associated with a failure event. For this end, we define several performance variables of system characteristics in the face of failures. However, defining these variables is nontrivial. They should be able to present the difference between system states in normal execution and those in failures. They also need to capture the temporal and spatial correlations of failure events in multiple tiers of the system components.

By investigating the structure of coalition system and the correlations of failure occurrences in multiple scopes of a system, we define the performance variables used in our failure prediction framework, as listed in Table 1. They are raw data collected from the system event logs or derived from the raw data. The runtime states of a subsystem is characterized by its processor and memory utilization and the volume of communication and I/O operations. These performance metrics provide insightful information about the causes of failures. Variables, such as the number of failures in a time window, their types and intervals, are used to model the statistical characteristics of failure dynamics. Along with the nodal allocation information, these variables are utilized to establish the spatial and temporal correlations among failure events. Note that some of the variables, such as $fCount$, tbf, *sptCorr* and *tmpCorr*, can be used on a node, cluster or the entire system. We use the subscript to specify the corresponding system scope in which these variables are used in the following discussion.



(a) Hardware Failures (b) CPU Failures (c) Memory Failures

(d) Software Failures (e) OS Failures (f) Filesystem Failures

Figure 2: Temporal distribution of major hardware and software failure events in LANL Cluster 20 from September 1, 2003 to August 31, 2005.

To predict occurrences of future failures in a coalition system, we need a representation that provides essential information about system performance status associated with a failure event. By clustering these representations, we are able to capture the failure dynamics across different scopes of the system. We will call such a representation a *failure signature*. Based on the performance variables defined in Table 1, a failure signature is constructed as a tuple (fID, *time*, $fLoct$, $fType$, *util*, *pktCount*, *ioCount*), where *util* includes (*usrUtil*, *sysUtil*, $frmUtil$) of a compute node, and *pktCount* and *ioCount* measure the number of packets and I/O requests in the sampling period that immediately precedes the failure. With these failure signatures collected in a coalition system and the node allocation information *alloc*, we will analyze the failure distributions and correlations in both space and time domains.

2.3 Design of Failure Predictor

The failure predictor of hPREFECTS uses the failure signatures together with the derived temporal and spatial correlations among failure signatures to forecast the probable patterns of failure occurrences in the future. (For simplicity in notation, we use δ to denote time-between-failure (tbf).) That is, given the failure occurrence in an observation instance δ_o, $F(\delta_o)$, estimate $F(\delta_o + \delta_p)$, where δ_p is the interval after which a prospective failure will appear with probability.

Occurrences of failures are quite dynamic in a large-scale coalition system. The number of failure events varies with time. Numerically, its value is related to some performance metrics of the system, *e.g.* the resource utilization, the volume of communication and I/O operation. We model this relationship by using function $\mathcal{F}$ in multiple scopes of a coalition, as

$$\begin{aligned} &\mathcal{F}_{node}(\delta_n, perf_n, tmpCorr_n) = 0 \\ &\mathcal{F}_{cluster}(\delta_c, perf_c, sptCorr_c, tmpCorr_c) = 0 \\ &\mathcal{F}_{system}(\delta_s, perf_s, sptCorr_s, tmpCorr_s) = 0 \end{aligned} \tag{2.1}$$

where δ denotes the time-between-failure in a node, cluster or system, $sptCorr$ and $tmpCorr$ are the spatial and temporal correlation among failure events in the corresponding scope. The performance state, $perf$, of a node can be represented by (*usrUtil*, *sysUtil*, $frmUtil$, *pktCount*, *ioCount*). The $perf_c$ and $perf_s$ are composed of the mean and variance values of these performance variables in a cluster and the system, respectively. To find the essential performance variables for a failure instance, we analyze their probabilistic dependency among them in experiment evaluation sections.

In essence, predicting failures in a coalition system is to find approximate function $\mathcal{F}$. Failure events are highly non-linear and it is difficult to find the relation between failure occurrences and performance states that fits various product systems. Instead of deriving function $\mathcal{F}$ directly, the predictor uses statistical learning approaches to perform failure prediction based on the current failure statistics and the resource utilization level. The prediction procedure can be expressed as follows,

$$x(\delta_{i+1}) = \mathcal{G}(x(\delta_i), x(\delta_{i-1}), \ldots, x(\delta_{i-k+1})) \tag{2.2}$$

where x denotes the measures of failure dynamics and $\mathcal{G}$ is the prediction function determined by a prediction mechanism with parameters' values in the k observations. The predictor's input layer is k consecutive measures $\delta_i, \delta_{i-1}, \ldots, \delta_{i-k+1}$, obtained with the aid of a tapped delay line.

In this way, failure correlations spanning across multiple observations are kept for failure prediction. In essence, for an average interval $\hat{\delta}$, we can maintain the correlation information of a period of $k * \hat{\delta}$ by using the order-k predictor, while being able to make failure predictions at a granularity of $\hat{\delta}$ at the same time. This scheme also increases the robustness of the failure predictor to noisy inputs because the noise effect of each measure fed to the predictor is suppressed by the multi-step looking back of the prediction mechanism.

3 Signature Classification based on Temporal and Spatial Failure Correlations

The objective of applying clustering to a database of failure signatures is to find the natural grouping of these signatures that characterizes correlations among failure instances. The output of clustering is a set of groups, plus a characterization of each

group. By inspecting the elements of failure signatures in each group, we can identify different regions of anomaly as well as a hint of the causing problems. In addition, the central signature of a group can be used as a syndrome of the failures, because it highlights the metrics that characterize a set of manifestations of the neighboring failure instances.

In order to render the description above operational, we specify distance metrics and clustering algorithms. We cluster failure signatures in two directions. One is to discover the causal dependency among failure instances in the space domain. The other is to explore their temporal locality in the time domain. In the following discussion, we use the availability and performance traces [5] from the LANL HPC system. The data was collected over the past 9 years and covers 22 high-performance computing systems, including a total of 4750 computers and 24101 processors. The data contains an entry for every failure that occurred during the 9-year time period. The associated performance traces recorded the resource consumption information by each user application and events occurred in the running of the system. For detailed information about the traces, please refer to [29].

3.1 Temporal Correlation

Studies in [29, 28, 23] found the skewness of failure distribution in time domain. Multiple failures may occur in a short time period. Liang [22] and Sahoo [27] used a fix time window to classify failure patterns for all types of failures. In reality, the time-between-failure (tbf) may follow various distributions for different types of failures. We profile time-between-failure in the LANL HPC system from its failure traces. Figure 2 presents the cumulative distribution functions of tbf for the major hardware and software failures. From the figure, we can see that tbf has a heavy tail distribution and its shape varies with the failure type. Software failures have a much more heavily tailed distribution in time than hardware failures. Even different types of software failures, such OS and file system failures, have distinct tbf patterns. This discovery suggests that we should use an adjustable timescale to model and measure the temporal correlation among failures of different types.

By closely inspecting the system event logs and performance logs, we found that the temporal locality of failure events was mainly due to two causes:

- (T1) some faults [1] cause several failure instances occurred on multiple compute nodes in a short interval;
- (T2) a failure event may appear multiple times on a node before its root problem is solved.

To cluster failure signatures in the time domain, we define the distance between two failure events f_i and f_j as the elapsed time between them, denoted by $d_{i,j} = \| f_i - f_j \| = |t_{f_i} - t_{f_j}|$. We develop a spherical covariance model, based on recent advance of Bayesian statistics [9], to quantify the temporal failure correlations. The model characterizes the relations of failure instances in time space based on their distance between each other, even when they occur on different nodes. We assume the timers of compute nodes in a cluster are synchronized. The spherical covariance, $C_T(d)$, for temporal correlation is defined as:

$$C_T(d) = \begin{cases} 1 - \alpha\frac{d}{\theta} + \beta(\frac{d}{\theta})^3 & \text{if } 0 \le d \le \theta \\ 0 & \text{if } d > \theta \end{cases} \quad (3.1)$$

[1] A fault is associated with incorrect state of a hardware or software component and it may cause a reduction in, or loss of, the capability of a component to perform a required function.

where θ is an adjustable timescale parameter for determining the temporal relevancy of two failure events, α and β are positive constants with $\alpha = 1 + \beta$. We use different values of *theta* to quantify temporal correlations of different types of failures. For example, with $\theta = 1$ hour for the LANL HPC system, we can capture more than 35% OS failures. For other coalition systems, θ can be determined by inspecting the cumulative distributions of the inter-failure time from their event logs. Two failures taken more than θ distance apart are considered as uncorrelated in time. $C_T(d)$ is nonnegative with limiting values of 1 at $d = 0$ and of 0 at $d = \infty$. After specifying the value of θ, we cluster the failure signatures of a compute node by comparing their $C_T(d)$ pair-wise. Failure signatures within a group is temporally correlated with high probability and likely to appear closely in time. The central signature is useful for investigating the root cause of the failure group and analyzing the distribution of inter-failure time among failure signatures in the same group.

In node-wide prediction, temporal correlations among failure events can also be utilized to remove duplicate failures in preprocessing event logs. To that end, we extend the spherical covariance model in (3.1) by dynamically setting θ using the current measure of the mean-time-to-repair (MTTR) based on the failure repair records in the administrative log, and calculating the temporal distances $d_{i,j}$ for failures of the same type only. Then, failure instances resulted from the same root problem are closely correlated with high values of their $C_T(d)$. The central signature is selected to represent the group. In this way, duplicated failure events are removed and the resulting set of events contains failures caused by distinct root problems. Algorithm 1 presents the pseudo-code of correlating failure signatures in time by a compute node and a master node in a cluster. The compute node constructs failure signatures and quantifies the temporal correlation among them. The master node first collects the failure signatures grouped by compute nodes in the cluster. It then inspects each pair of failure signature from different groups to calculates the temporal correlation. Although the algorithm needs to scan failure signatures of every compute node in a cluster, the total number of failure events occurred within a time window is quite limited.

ALGORITHM 1. ***Temporal clustering of failure signatures***

```
/* Temporal clustering on each compute node */
NodePredictor.TemplClustering() {
1:  collect time, location, Util and nodeAlloc information
      of failure instances in the current time window;
2:  construct failure signatures into a set S;
3:  mttr = current measure of MTTR;
4:  θ = inter-failure time with p cumulative distribution;
5:  for any pair of failure signatures f and g (t_f ≤ t_g) in S do
6:    d_{f,g} = t_g − t_f;
7:    if f and g are of the same type and d_{f,g} < mttr then
8:      remove g from S;
9:    else if d_{f,g} ≤ θ then
10:     c_{f,g} = 1 − α * d_{f,g}/θ + β * (d_{f,g}/θ)^3;
11:     if c_{f,g} ≥ C then
12:       Group_f = Group_f ∪ {g};
13:       remove g from S;
14:     end if
15:   end if
16: end for
17: return Groups;
18:}
```

```
/* Temporal clustering on master nodes */
MasterPredictor.TemplClustering() {
1:   collect signature groups from compute nodes in the cluster;
2:   for any pair of nodes i, j in nodelist do
3:     T[i, j] =0;
4:     for any failure signature f on node i and g on j (t_f ≤ t_g) do
5:       d_{f,g} = |t_f − t_g|;
6:       if d_{f,g} ≤ θ then
7:         c_{f,g} = 1 − α ∗ d_{f,g}/θ + β ∗ (d_{f,g}/θ)^3;
8:         T[i,j] = T[i, j] + c_{f,g};
9:         if c_{f,g} ≥ C then
10:          Group_f = Group_f ∪ {g};
11:          remove g from the signature set;
12:        end if
13:      end if
14:    end for
15:  return T, Groups;
16:}
```

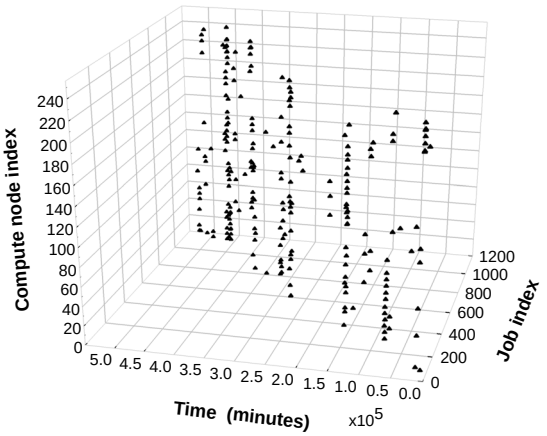


Figure 3: Spatial clustering of failures in Cluster 20 of the LANL HPC system. There are 256 nodes in total.

3.2 Spatial Correlation

In cluster coalition environments, performance-hungry applications exploit computational power of available nodes and execute their tasks in parallel. For example, in a typical cluster of the LANL HPC system, say Cluster 20, 21.6% jobs consume 85.4% processor time. They were run on at least 4 nodes and some were allocated 52 nodes out of the total 256 nodes. In such a computing environment, a software defect or bug in a running job will cause multiple nodes to fail. In the LANL HPC system, 31.6% nodes experienced 70.7% failures and these nodes serviced 85.2% application jobs. The distribution of failures among compute nodes is uneven. Figure 3 illustrates the spatial clustering among failures in Cluster 20. The figure presents that failures occurred in groups. For example, 12 out of 16 nodes allocated to Job 1089 experienced OS failures between late January 28, 2004 and early January 29, 2004. The figure also shows the strong correlation between failure distribution and the application allocation among nodes.

By inspecting the system failure and performance traces, we found

- (S1) a failure may (nearly) simultaneously occur on multiple nodes in a cluster or across its border;
- (S2) a failure on a node may cause another failure happening on a different node.

The first case is common in parallel computing, where a single-program-multiple-data (SPMD) application runs on a set of nodes and a fatal software bug in the application will make multiple nodes come to failure. The second case happens among cooperative nodes. For example, a processor failure on one node may cause its running program to send wrong data to another node, which leads to an overflow and system dump. We refer to Case (S1) as *failure multiplication correlation* and to Case (S2) as *failure propagation correlation*. Note that in this paper we consider those propagation correlations that are caused by communication between failing nodes.

We develop an aggregate stochastic model to cluster failure signatures in the space domain and use these groups for failure prediction. The model analyzes the probabilistic dependency among failure instances of different nodes, and combines the nodal failure statistics in a cluster into an aggregated state, which is further combined with failure states of other clusters into an aggregated system state. Modeling based on hierarchical decomposition and aggregation makes it possible to treat a large scale system by considering a reduced one with essentially the same features but with reduced complexity.

The analysis of failure multiplication correlation is based on the occurrence relations of failure events. Let set $F = \{f_1, f_2, \ldots, f_m\}$ denote all possible types of failures that may occur in a coalition clusters system, and $N = \{n_1, n_2, \ldots, n_r\}$ be the set of all compute nodes in the system. Random variables $\hat{n}_i$ and $\hat{f}_j$ are defined as: So, n_i and f_j indicate whether a node fails or a failure happens. A

$$\hat{n}_i = \begin{cases} 1 & \text{if node } n_i \text{ fails in a unit interval,} \\ 0 & \text{otherwise.} \end{cases} \qquad \hat{f}_j = \begin{cases} 1 & \text{if failure } f_j \text{ occurs in a unit interval,} \\ 0 & \text{otherwise.} \end{cases}$$

unit interval is a small period of time when only one failure event can appear on a node. Time window is measured in unit intervals. Based on failure statistics, we measure the conditional probabilities $p(\hat{f}_j \mid \hat{n}_i)$, that is if node n_i fails, the probability that the failure is f_j, for $1 \le i \le r$ and $1 \le j \le m$.

Now, let's first consider the failure multiplication correlations among two nodes, say n_1 and n_2. The number of failures counted in a time window is $nodeFCount_i = \hat{n}_i \cdot w$. If we fix the window size in measurements, then the expected number of failures becomes,

$$E[nodeFCount_i] = w \cdot E[\hat{n}_i] = w \cdot p(\hat{n}_i) = w \sum_j p(\hat{n}_i \mid \hat{f}_j) \cdot p(\hat{f}_j).$$

We can further calculate the covariance of $nodeFCount_i$ of different compute nodes to analyze the correlations of these variables. Assume the failure dynamics of the two nodes are monitored independently. Then according to the Bayesian theorem,

$$p(\hat{n}_1\hat{n}_2 \mid \hat{f}_j) = \frac{p(\hat{f}_j \mid \hat{n}_1) \cdot p(\hat{f}_j \mid \hat{n}_2) \cdot p(\hat{n}_1) \cdot p(\hat{n}_2)}{p(\hat{f}_j)^2}.$$

According to the inclusion-exclusion principle, the number of failure events after considering the failure multiplication correlation becomes

$$\begin{aligned} E[clusterFCount] = \; & w \cdot \Big(\sum_i p(\hat{n}_i) - \sum_{i,k} \sum_j p(\hat{n}_i\hat{n}_k \mid \hat{f}_j) p(\hat{f}_j) + \\ & \cdots + \sum_j (-1)^r p(\hat{n}_1 \cdots \hat{n}_r \mid \hat{f}_j) p(\hat{f}_j) \Big). \end{aligned} \tag{3.2}$$

By using these probabilities of failure distributions along with the temporal correlation among failure signatures, predictors in node, cluster and system wide calculate the number of failures

that will occur in the prediction window with certain probability. Since $E[nodeFCount]$, $E[clusterFCount]$ and $E[sysFCount]$ are correlated, we use their corresponding prediction results to cross-verify each other. Then we refine the temporal and spatial correlations among the predicted failure instances by using the node allocation information in the system.

For failure propagation correlations, we define *propagation groups* to cluster failure signatures.

Definition 1 (***Propagation relation***). *Let $\diamond$ be a relation on the set of failures F. It satisfies:*

1. *For any f_i and f_j in F, if f_i can cause f_j on another node, then f_i and f_j have relation $\diamond$, denoted as $f_i \diamond f_j$;*
2. *For any f_i, f_j and f_k in F, if $f_i \diamond f_j$ and $f_j \diamond f_k$, then $f_i \diamond f_k$.*

□

Relation $\diamond$ formulates the failure propagation dynamics between nodes. According to its definition, $\diamond$ is irreflexive [2] and asymmetric [3], and transitive. Thus, $(F, \diamond)$ is a strict partial order set. The propagation relation can be represented by Hasse diagrams. For strict partial order set $(F, \diamond)$, we calculate the transitive closure of relation $\diamond$ as $\mathcal{D}_\diamond$. The members of $\mathcal{D}_\diamond$ are groups of failure signatures that have possible propagation relations. Then we treat each member in $\mathcal{D}_\diamond$ as a unit and calculate its occurrence probability to compute nodes in N. After this transformation, the failure propagation correlation can be reformulated by the stochastic models that we use to analyze the failure multiplication correlation. Thus, we consider both of the spatial correlations in failure prediction.

In implementation, predictor estimates the probabilities $p(\hat{f}_j \mid \hat{n}_i)$ and constructs the propagation relation $\diamond$ based on the failure statistics derived from event logs. For example, the predictor in node n_i counts the number of f_j occurrences and the total number of all failures in all past time windows and calculates the ratio between them as an estimate of $p(\hat{f}_j \mid \hat{n}_i)$. We set the length of an observation interval based on the measured mean-time-to-failures (MTTF) so that only one failure event can occur on a node in an interval. We also calculate the ratio of the number of intervals in which failures are observed on node n_i to the total number of intervals as the value of $p(\hat{n}_i)$. The master node of the system collects failure signatures from all compute nodes and estimate $p(\hat{f}_j)$ by (number of f_j instances)/(total number of failures) in the system. The master node also mines failure signatures to establish the propagation relations between failures. Failures f_j and f_k follow $f_j \diamond f_k$, if f_k occurs on a node, say A, after node A receives a message from another node B which suffers failure f_j. The probability of sending such a message equals to the inverse of the total number of messages sent by node B in the interval between occurrence time of failures f_j and f_k. In runtime, the predictor updates these probabilities and relation information using newly generated failure measures as system runs on. Then the correlations among failure signatures are analyzed. Node allocation information is utilized to refine failure correlations for prediction.

Algorithm 2 presents the pseudo-code of correlating failure signatures in space. Although the algorithm needs to scan failure events of every compute node in a cluster, the total number of failure events occurred within a time window is quite limited. The system wide predictor finds the failure correlations, utilizes cluster wide results and makes predictions in a similar way. The aggregate stochastic model reduces the state space of failure statistics and computational complexity, which facilitates online failure prediction in a coalition system.

[2] We treat the case in which failure f_i causes f_i on another node in a small interval as an instance of the failure multiplication correlation.

[3] If failure f_i causes f_j and failure f_j causes f_i in a small interval, then according to the transitive property f_i causes f_i, which is an instance of the failure multiplication correlation.

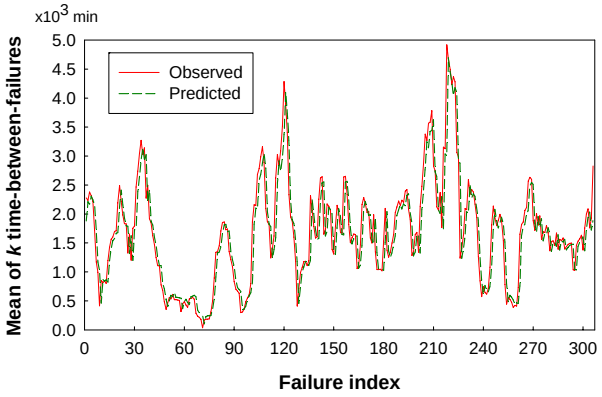


Figure 4: Performance of failure prediction in Cluster 20 using order-8 neural network predictor. Training samples are based on failure records from September 2003 to August 2004 and prediction is for September 2004 - August 2005.

ALGORITHM 2. ***Spatial correlating of failure signatures***

MasterPredictor.SpatioCluster() *{*
1: ***for*** *i = 1 upto nodelist.size* ***do***
2: $f_{j,i}$ *= number of failure signature of type j on node i;*
3: $n_i = f_{1,i} + f_{2,i} + \ldots + f_{m,i}$*;*
4: $p_{j,i} = f_{j,i}/n_i$*;*
5: $p_{f_j} = (f_{j,1} + f_{j,2} + \ldots + f_{j,r})/(n_1 + n_2 + \ldots + n_r)$*;*
6: p_{n_i} *= number of intervals with failures on i / total number of intervals;*
7: ***end for***
8: ***for*** *any pair of nodes i, j in nodelist* ***do***
9: $p_{n_{ij},f_k} = p_{k,i} * p_{k,j} * p_{n_i} * p_{n_j} / p^2_{f_k}$*;*
10: $S[i, j] = p_{n_{ij},f_1} * p_{f_1} + \ldots + p_{n_{ij},f_m} * p_{f_m} + p_i * p_{m_{ij}}$*;*
11: ***if*** *node i and j are allocated to the same job in nodeAlloc* ***then***
12: $S[i, j] = S[i, j] + 1$*;*
13: ***end if***
14: ***end for***
15: ***return*** *S;*
16:}

4 Offline Prediction Performance Evaluation

The hPREFECTS provides a general failure prediction framework, to which many prediction algorithms can be applied. As a proof of concept, we implemented a prototype of hPREFECTS using several illustrating algorithms. In this section, we evaluate the performance of hPREFECTS for offline failure prediction using traces from the LANL HPC system. We will discuss the results of online prediction in the WSU Grid in the next section.

We used traces from the LANL HPC system in the experiments of offline prediction, because the scale of the system is large for analysis of failure dynamics in high-performance computing systems. In addition, the detailed records of failures and performance information allow us to have a deep understanding of the causes of failure events.

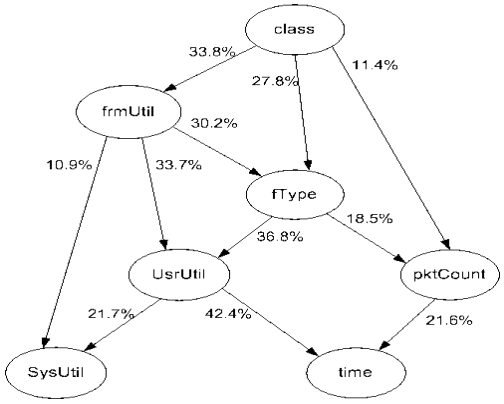


Figure 5: Bayesian network model for Node 1 in Cluster 20.

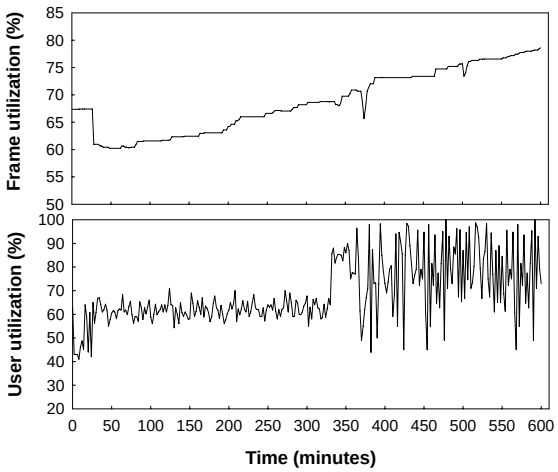


Figure 6: A case study of failure prediction in Node 1. (Time starts from September 6, 2004 22:20:00)

In our prototype, we implemented a group of illustrating prediction algorithms, including a neural network approach, to learn and forecast failure dynamics based on the temporal and spatial data among the failure signature clusters. We selected Cluster 20 as the testing system in our experiments, due to the availability of its performance traces. It is a typical HPC cluster in the LANL computing system and has 256 compute nodes.

4.1 Failure Prediction Accuracy

In the first experiment, we predicted the time between failures using a neural network based predictor. The predictor network has 3 input neurons to receive temporal and spatial correlation data and failure measures; 1 output neuron for prediction result; 3 hidden layers and 4 neurons in each hidden layer. We used the Weka machine learning software [7] in the implementation. To mitigate the measurement noises, we applied an order-k predicting approach, where k failure measures were tapped into the predictor. First, we calculated the mean of the k measures, as $m_i = (\sum_{j=i-k+1}^{i} \delta_j)/k$, where $\delta_j = time_j - time_{j-1}$ being the time between the jth and the $(j-1)$th failures. Then, m_i was used as an input along with the temporal and spatial correlation data to the predictor. After the predictor outputs the predict, $\hat{m}_{i+1}$, we retrieved the predict of δ of the next failure by $\hat{\delta}_{i+1} = \hat{m}_{i+1} * k - \sum_{j=i-k+2}^{i} \delta_j$.

Figure 4 presents the prediction performance as we used the neural network predicting algorithm with 8 tapped failure measures. The predictor was trained by the failure measures from September 2003 to August 2004. We plotted the prediction results. The average error rate in predicting (m_i), in Figure 4, is $err_{\hat{m}} = \overline{|m_i - \hat{m}_i|/m_i} = 14.9\%$. After retrieving the predicts of δ, the average error rate is $err_{\hat{\delta}} = \overline{|\delta_i - \hat{\delta}_i|/\delta_i} = 23.5\%$ and the average prediction accuracy is 76.5%. From the figure, we also find that the prediction error rate is high, when values of the δ measures are relatively small. Due to the temporal correlation among failure events, the mean values may remain small for a period of time. A small difference in the predict from the observed makes the error rate relatively high. Actually, the inclusion of spatial and temporal correlation information in our predictor has already reduce the prediction error. Their effects will be discussed in the following experiment.

Among the 450 failure events occurred in Cluster 20 within September 2003 – August 2004, 252 (56%) were caused by hardware faults, 175 (38.9%) by software faults, 17 (3.8%) by undermined problems and the remaining 6 (1.3%) by power outage and network breakdown. We extracted the two dominant types of failures, hardware and software failures, from the traces and evaluated the performance of predicting these two sets of failures respectively. By using an order-8 neural network based predictor, we predicted software failures with the average accuracy of 81.6%, and hardware failures with 72.9% accuracy. The difference in prediction performance of these two major types of failures is resulted from the different distribution patterns of them. Software failures have stronger correlations among each other. As an illustration, Figure 2 shows that software failures are distributed a much more heavy tail than hardware failures.

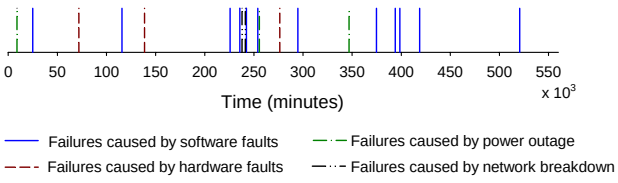
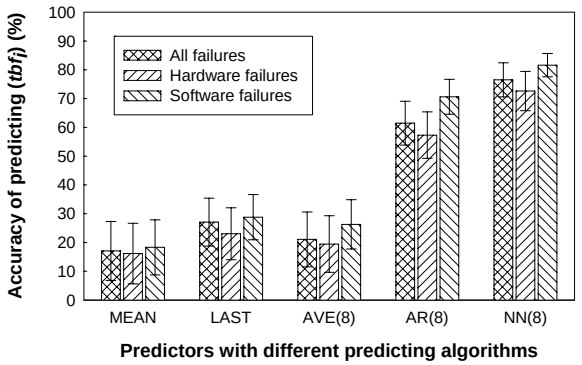
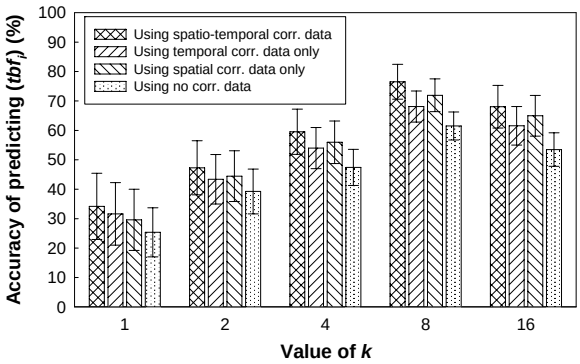


Figure 8: Failure events of Node 1 in Cluster 20.

In addition to analyzing the failure behavior of Cluster 20, we conducted experiments to predict failure events in individual nodes. Figure 8 plots the 20 failures experienced by Node 1 in Cluster 20 from September 2003 to August 2004. Because of the limited number of failure events occurred in a node, we can not apply the supervised statistical learning approaches to approximate the failure dynamics with a small training set. Instead, we extract low-level performance data, such as the processor and memory utilization, communication and I/O operation status, number of processes, file handles and network sockets, from the failure signature for each failure event and from the performance traces for normal execution. We then consider a Bayesian network to present the probabilistic dependency among these fine-grained performance metrics for failure prediction and diagnosis. Due to the availability of these performance data in the LANL HPC traces, we used the first 3 metrics in our experiments. Figure 5 presents the dependency graph obtained by jBNC [3] for Node 1 in Cluster 20, using the failure and performance data in



(a) Prediction of hardware and software failures in different algorithms



(b) Effects of k and correlation data on prediction performance using order-k neural network predictor

Figure 7: Failure prediction in Cluster 20. Training samples are based on failure records from September 2003 to August 2004 and prediction is for September 2004 - August 2005.

September 2003 - August 2004. From the Bayesian network, we found that among the failure events occurred in Node 1, 84% occurred at the time with high user utilization and 61% occurred at the time with inter-node communication. Between the user and system utilization, the former affects the nodal availability more than the latter.

Based on the preceding Bayesian network, we predicted the occurrences of failures in Node 1. Figure 6 presents a case in our prediction. The upper figure plots the frame utilization ($frmUtil$) of Node 1 between September 6, 2004 22:20 pm and September 7, 2004 08:20 am, and the lower figure shows the user utilization ($usrUtil$) of Node 1 during the same period. At 03:50 am on September 7, 2004, the $usrUtil$ jumped to more than 86% and became fluctuated; at about the same time, the $frmUtil$ had an increase which was followed by a drop, and then continued to increase. The hPREFECTS predicted a failure occurred. After we checked the failure traces, we found a record of scheduler software failure recorder at September 7, 2004 08:20 am, after which the node was shut down for repair.

4.2 Sensitivity to Prediction Algorithms

In the last experiment, we evaluated the sensitivity of our predictor to different prediction algorithms. In the hPREFECTS prototype, we implemented four additional time-series predictors by using the gretl GNU time series library [2]: MEAN takes the average of previous measurements as prediction; LAST uses the last measurement; AVE(k) uses the average of last k measurements; AR(k) is autoregressive algorithm. Figure 7(a) presents the results in predicting hardware and software caused failures with different prediction algorithms. According to the figure, NN(8) and AR(8) performed the best among them. This is because of the learning capacity of these two approaches. They can adapt to the change of time-between-failures and approximate their trend. For the two major causes of failures, hardware faults and software faults, failures resulted from the latter source are more predictable compared with those from the former. This can be explained by looking into the distribution of failures of these two categories. Software-based failures are better clustered in time and space than hardware-based failures (see Figure 2 as an example).

Recall that because the time-between-failure varies dramatically from one measure to another, we use the k tapped measures to smooth its variance and make it predicable by the NN(k) predictor. The choice of k is critical. Figure 7(b) shows the effect of different k's on the prediction performance. When k is small, the fluctuation of measured δ's makes failure prediction difficult. As k increases, the mean value sequence (m_i) leads to improving the prediction accuracy. However, as k increases further, the variance of (δ_i) is smoothed out in (m_i). Although the predictor can predict the sequence (m_i) with less error rate, the retrieved sequence $(\hat{\delta}_i)$ from (m_i) is a distorted predict of the observed time-between-failure sequence (δ_i). As a result, the prediction accuracy decreases instead.

The failure signatures are clustered according to their correlations in time and space domains. Then, these correlation data is used to predict the intervals of future failures. To analyze the effects of the correlation data on the performance of failure prediction, we conducted a series of experiments to measure the prediction accuracy using different order-k algorithms with/without correlation data. The results are presented in Figure 7(b). From the figure, we can see that the existence of correlation data influences the prediction performance. For example, when $k = 8$, by using the temporal and spatial correlation data, the prediction accuracy is creased by 6.6% and 10.4% respectively, compared with that of the correlation-unaware one. Between these two correlations, the spatial one makes more contribution to the improvement of prediction performance for large k.

5 Online Prediction Performance

To evaluate the prediction performance in real system at runtime, we performed online failure prediction in an institute-wide computational grid. The WSU Grid consists of three Linux clusters, denoted by C_1, C_2 and C_3, located in three separate buildings on campus. It contains 40 high-performance compute servers dedicated to computational research. Cluster C_1 and C_2 consist each of 16 nodes, and there are 8 nodes in cluster C_3. Within each cluster, nodes are interconnected by gigabit Ethernet switches. Connections between clusters are through 100M fast Ethernet.

Typical applications running on the grid includes the molecular dynamics simulations, gene analysis, fluid dynamics simulation and more. These parallel applications ran on 8 to 30 nodes and some of them lasted for more than 10 days. The grid is also open to institute students to execute their sequential and parallel programs.

We installed our predictors on compute nodes of each clusters and their master nodes in the WSU Grid. By making on-line predictions, our failure predictors provide useful information for

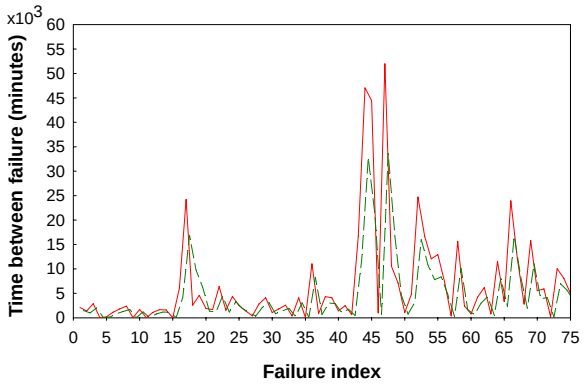


Figure 9: Online failure prediction in comparison with observed failure events in the WSU Grid.

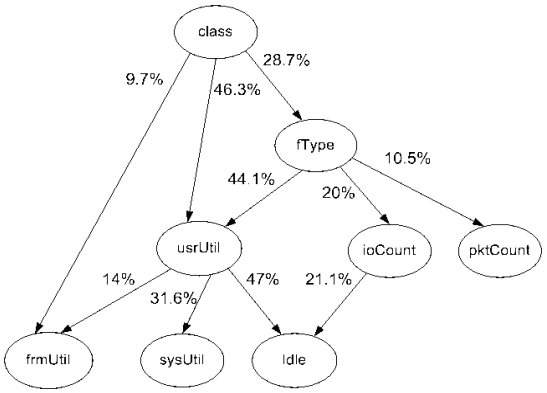


Figure 10: Bayesian network model for Node 7 in cluster C_1.

resource management, load distribution. In this experiment, first, we trained the predictors using failure event records between May 2004 and April 2006. Then, we evaluated the on-line prediction performance from May 12, 2006 to April 2, 2007. We record the failure predictions and compare them with the observed failure events later mined from the event logs.

Figure 9 depicts predicted and observed δ of failure events during the online prediction based on the NN(8) algorithm in Grid A. Temporal and spatial correlations among failure occurrences were utilized for prediction. The node allocation information was used to refine the correlations among failure signatures. From the figure, we can see the predictor can capture the trends of failure dynamics. The average accuracy of prediction is 70.3%. To make a prediction, it took 2.26 seconds for the master node (a Pentium Xeon computer with 2.6 GHz processor and 2.5 GB of memory) to analyze the system wide failure events, find the failure correlations and make a prediction, after receiving the failure event data from the three clusters.

We conducted online prediction of failure occurrence for individual nodes. From a total of eighteen low-level performance variables, such as the processor utilization, percentage of idle time, frame utilization, volume of communication and I/O operations, and swap utilization, we found seven of them were the dominant factors in node-wide failure prediction. Figure 10 presents the Bayesian probabilistic dependency network of these variables. We used the failure event records between May 2004 and April 2006 in Node 7 to construct this network. From the figure, we can see the I/O operation is also important factor in failure prediction, and $frmUtil$ is not so important as that in Figure 5 due to the different memory capacity and different operating systems in the two compute nodes. In comparing Figure 10 with Figure 5, we can see that some dependency relations are the same in both nodes, such as that between *usrUtil* and *sysUtil*. We also noticed the difference between the two networks. This is caused by the different hardware/software configurations and applications in the two nodes. Based on the Bayesian network (Figure 10), we successfully predicted 5 out 6 software failures and 1 out 4 hardware failures occurred in Node 7 between May 2006 and April 2007.

6 Related Work

As the complexity of computing systems increases, failure management tasks require significantly higher levels of automation. Examples include diagnosis and prediction based on realtime streams of computer events, and performing continuous monitoring of the runtime services. The core of *autonomic computing* [17, 21] is the ability to analyze data in realtime and to predict potential problems. The goal is to avoid catastrophic failures through prompt execution of remedial actions.

Failure prediction provides a vehicle for autonomic computing in coalition systems. To predict the occurrences of failures, it is imperative to understand the characteristics of failure behaviors. Research in [29, 23, 28, 41] studied event traces collected from clusters and supercomputers. They found that failures are common in large-scale systems and their occurrences are quite dynamic, displaying uneven inter-arrival time. Sahoo et al. [28] found the correlation of failure rate with hour of the day and the distribution of failures across nodes. They reported that less than 4% of the nodes in a machine room experience almost 70% of the failures and found failure rates during the day to be four times higher than during the night. Similar result was observed by Schroeder and Gibson [29]. several studies [8, 37, 38] have examined system logs to identify causal events that lead to failures. Correlation between the workload intensity and the failure rate in real systems was pointed out in many studies [14, 24, 26, 20, 12].

Tang et al. [35, 34] studied the failure log collected from a small VAX-cluster system and showed that failures on different machines are correlated. Xu et al. [40] performed a study of error logs collected from a heterogeneous distributed system consisting of 503 PC servers. They showed that failures on a machine tend to occur in bursts, possibly because common solutions such as reboots cannot completely remove the problem causing the failure. They also observed a strong indication of error propagation across the network, which leads to the correlation between failures of different nodes. A recent study [18] collected failure data from three different clustered servers, and used Weibull distribution to model time-between-failure. Both these studies [40, 18] found that nodes which just failed are more likely to fail again in the near future. At the same time, it has also been found [36] that software related error conditions can accumulate over time, leading to system failing in the long run.

There were recent works utilizing temporal and/or spatial correlations of failures for failure prediction and proactive management. Sahoo et al. [27] inspected the eventset within a fixed time window before a target event for repeated patterns to predict the failure event of all types. Later, Liang et al. [22] profiled the time-between-failure of different failure types and applied a heuristic approach to detect failures by using a monitoring window of preset size corresponding to event type. Mickens and Noble [25] assumed the independency of failures among compute nodes and

used the per-node uptime data to predict whether a failure might occur on that node in the next time window of fixed size. In building classification rules, Sahoo et al. [27] took the ordering of events into consideration. They utilized a Bayesian network to analyze the causes of failures in a node individually. The spatial correlation among failure was considered by Liang et al. [22]. The authors analyzed the number of failures in every midplane of IBM BlueGene/L supercomputer. They found skewness in the distribution of network failures only, among the midplanes. Fu and Xu [16] exploited the failure correlation information to predict the number of failures in a prediction time window in a coalition environment. Besides, failure prediction aside, there are many works on failure detection [39, 30, 15, 32]. These techniques are complementary to our failure prediction approaches in constructing a comprehensive failure management infrastructure.

There has been prior work on monitoring and predicting failures for specific components in computer systems. Storage is one such subsystem which has received considerable attention because of its higher failure rates. S.M.A.R.T. is a recent technology, that disk drive manufacturers now provide, to help predict failures of storage devices [19]. SIGuardian [4] and Data Lifeguard [1] are utilities to check and monitor the state of a hard drive, and predict the next failure, to take proactive remedies before the failure. More recently, a Reliability Odometer [33] has been proposed for processors to track their wear-and-tear and predict lifetimes.

7 Conclusions

In this paper, we present a failure proactive prediction framework, which exploits the temporal and spatial correlations among failure events in coalition systems. Failure events are formally represented by failure signatures. By clustering signatures in the time and space domains, we explore the temporal and spatial correlations among failure occurrences. Node allocation information is utilized to refine the predicted correlations. Experimental results of offline and online prediction on production coalition systems present the feasibility of applying failure prediction to autonomic management for high-availability network computing.

There are some limitations and potentials in our work. First, the spherical covariance model quantifies temporal correlation of failures with an adjustable timescale. However, the temporal distribution of failures of different types may interfere with each other, which leads to a hybrid temporal correlation. More advanced model is needed to analyze such correlation. Second, we model failure propagation that is caused by inter-node communication. There are other causes of propagation. For example, busy waiting in synchronous computation among multiple nodes may make one node unavailable, even though there is no message passing to or from that node. To detect and predict such failures, we need to analyze the detailed structure of application programs and model their runtime behavior. Third, we use a limited number of performance variables to analyze the cause of failures. In order to obtain an in-depth understanding of failure behavior and trend, more performance information and non-performance information, such as temperature and power, are needed for thorough analysis.

Acknowledgments We would like to thank the anonymous reviewers for their constructive comments and suggestions. We would also like to thank Philip Sokolowski and Michael Thompson for their kind help in data collection from the WSU Grid. This research was supported in part by U.S. NSF grants CCF-0611750, DMS-0624849, CNS-0702488 and CRI-0708232.

References

[1] Data lifeguard. Available at: http://www.wdc.com/en/library/2579-850105.pdf.

[2] gretl: GNU Regression, Econometrics and Time-series Library. Available at: http://gretl.sourceforge.net/.

[3] jbnc: Bayesian network classifier toolbox in java. Available at: http://jbnc.sourceforge.net/.

[4] Siguardian. Available at: http://www.siguardian.com/.

[5] System availability, failure and usage data sets. Los Alamos National Laboratory (LANL). Available at: http://institutes.lanl.gov/data/fdata/.

[6] Wayne State University, Grid computing. Available at: https://www.grid.wayne.edu/.

[7] Weka: The university of waikato. machine learning software in java. Available at: http://www.cs.waikato.ac.nz/ml/weka/.

[8] H. Berenji, J. Ametha, and D. Vengerov. Inductive learning for fault diagnosis. In *Proceeding of IEEE International Conference on Fuzzy Systems*, 2003.

[9] J. O. Berger, V. D. Oliveira, and B. Sansó. Objective Bayesian analysis of spatially correlated data. *Journal of the American Statistical Association*, 96(456):1361–1374, 2001.

[10] G. Candea, A. B. Brown, A. Fox, and D. Patterson. Recovery-oriented computing: Building multitier dependability. *IEEE Computer*, 37(11):60–67, 2004.

[11] G. Candea, S. Kawamoto, Y. Fujiki, G. Friedman, and A. Fox. Microreboot–a technique for cheap recovery. In *Proceeding of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2004.

[12] X. CastiUo and D. P. Siewlorek. Workload, performance and reliability of digital computing systems. In *Proceeding of IEEE Symposium on Fault-Tolerant Computing (FTCS)*, 1981.

[13] R. Christodoulopoulou, K. Manassiev, A. Bilas, and C. Amza. Fast and transparent recovery for continuous availability of cluster-based servers. In *Proceeding of ACM Symposium on Principles and practice of parallel programming (PPoPP)*, 2006.

[14] B. Chun and A. Vahdat. Workload and failure characterization on a large-scale federated testbed. Technical Report IRB-TR-03-040, Intel Research Berkeley, 2003.

[15] J. Dunagan, N. J. A. Harvey, M. B. Jones, D. Kostic, M. Theimer, and A. Wolman. FUSE: Lightweight guaranteed distributed failure notification. In *Proceeding of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2004.

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

[17] A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, 2003.

[18] T. Heath, R. P. Martin, and T. D. Nguyen. Improving cluster availability using workstation validation. In *Proceeding of ACM International Conference on Measurement and modeling of computer systems (SIGMETRICS)*, 2002.

[19] G. Hughes, J. Murray, K. Kreutz-Delgado, and C. Elkan. Improved disk-drive failure warnings. *IEEE Transactions*

on Reliability, 51(3):350–357, 2002.

[20] R. K. Iyer, D. J. Rossetti, and M. C. Hsueh. Measurement and modeling of computer reliability as affected by system activity. *ACM Transactions on Computer Systems*, 4(3):214–237, 1986.

[21] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.

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

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

[24] J. Meyer and L. Wei. Analysis of workload influence on dependability. In *Proceeding of IEEE International Symposium on Fault-Tolerant Computing (FTCS)*, 1988.

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

[26] S. Mourad and D. Andrews. On the reliability of the IBM MVS/XA operating system. *IEEE Transactions on Software Engineering*, 13(10):1135–1139, 1987.

[27] R. K. Sahoo, A. J. Oliner, I. Rish, and et al. Critical event prediction for proactive management in large-scale computer clusters. In *Proceeding of ACM International Conference on Knowledge Discovery and Data Dining (SIGKDD)*, 2003.

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

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

[30] E. Schuchman and T. N. Vijaykumar. BlackJack: Hard error detection with redundant threads on SMT. In *Proceeding of IEEE International Conference on Dependable Systems and Networks (DSN)*, 2007.

[31] H. Song, C. Leangsuksun, and R. Nassar. Availability modeling and analysis on high performance cluster computing systems. In *Proceeding of International Conference on Availability, Reliability and Security (ARES)*, 2006.

[32] N. Sridhar. Decentralized local failure detection in dynamic distributed systems. In *Proceeding of IEEE International Symposium on Reliable Distributed Systems (SRDS)*, 2006.

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

[34] D. Tang and R. K. Iyer. Impact of correlated failures on dependability in a VAXcluster system. In *Proceeding of IFIP Working Conference on Dependable Computing for Critical Applications*, 1991.

[35] D. Tang, R. K. Iyer, and S. S. Subramani. Failure analysis and modelling of a VAXcluster system. In *Proceeding of IEEE International Symposium on Fault-Tolerant Computing (FTCS)*, 1990.

[36] K. Vaidyanathan, R. E. Harper, S. W. Hunter, and K. S. Trivedi. Analysis and implementation of software rejuvenation in cluster systems. In *Proceeding of ACM International Conference on Measurement and modeling of computer systems (SIGMETRICS)*, 2001.

[37] R. Vilalta and S. Ma. Predicting rare events in temporal domains. In *Proceeding of IEEE International Conference on Data Mining (ICDM)*, 2002.

[38] G. M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In *Proceeding of ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 1998.

[39] M. Wiesmann, P. Urban, and X. Defago. An SNMP based failure detection service. In *Proceeding of IEEE International Symposium on Reliable Distributed Systems (SRDS)*, 2006.

[40] J. Xu, Z. Kallbarczyk, and R. K. Iyer. Networked Windows NT system field failure data analysis. Technical Report CRHC 9808, UIUC, 1999.

[41] P. Yalagandula, S. Nath, H. Yu, P. B. Gibbons, and S. Sesha. Beyond availability: Towards a deeper understanding of machine failure characteristics in large distributed systems. In *Proceeding of USENIX WORLDS*, 2004.