J. Parallel Distrib. Comput. 70 (2010) 1100–1109

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Quantifying event correlations for proactive failure management in networked computing systems

Song Fu [a,*], Cheng-Zhong Xu [b]

[a] Department of Computer Science and Engineering, University of North Texas, United States
[b] Department of Electrical and Computer Engineering, Wayne State University, United States

ARTICLE INFO

Article history:
Received 19 August 2008
Received in revised form
17 April 2010
Accepted 29 June 2010
Available online 13 July 2010

Keywords:
Failure characterization
Temporal correlation
Spatial correlation
System availability
Networked computing systems
Autonomic management

ABSTRACT

Networked computing systems continue to grow in scale and in the complexity of their components and interactions. Component failures become norms instead of exceptions in these environments. Moreover, failure events exhibit strong correlations in the time and space domains. In this paper, we develop a spherical covariance model with an adjustable timescale parameter to quantify the temporal correlation and a stochastic model to characterize spatial correlation. The models are further extended to take into account 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. Experimental results on a production coalition system, the Wayne State Computational Grid, show the offline and online predictions made by our predicting system can forecast 72.7–85.3% of the failure occurrences and capture failure correlations in a cluster coalition environment.

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. Failure occurrences as well as their impact on system performance and operation costs are becoming an increasingly important concern to system designers and administrators.

The growing complexity of hardware and software mandates autonomic management of failures in production systems. 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 can help determine possible occurrences of fatal events in the near future and help develop more effective failure-tolerant solutions for improving system availability.

To achieve self-management of component failures in a system, 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 [22] and LANL high-performance computing clusters [29], revealed important patterns in failure distribution. Although the time-between-failure is highly nonlinear, there exist time-of-day and day-of-week patterns in long time spans [29, 27]. 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 [27] and multiple nodes may fail almost simultaneously [22]. These temporal and spatial correlation properties of failure events revealed by offline profiling provide important information for predicting the trend of failure dynamics.

Most of today's failure characterization approaches are heavily empirical, applying heuristics to explore the temporal and spatial correlation of failures based on profiling. There is a lack of formal models that quantify the temporal correlation among failures in different timescales. Most networked computing systems are 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. It is known that there is dependency between the workload (its type and intensity) and the failure rate [29,27]. However, there are few works on the impact of application allocation on failure correlation.

In this paper, we analyze and quantify both temporal and spatial correlations for proactive failure management in networked computing systems. We develop a covariance model with an adjustable timescale to quantify the temporal correlation and a stochastic model to describe the spatial correlation. We utilize information of application allocation in a coalition system to discover more

* Corresponding author.
E-mail addresses: songfu@unt.edu, song@nmt.edu (S. Fu), czxu@wayne.edu (C.-Z. Xu).

0743-7315/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2010.06.010

Table 1
Variables characterizing failure dynamics.

Variable	Description
fID	Failure identification number
fLoct	Location of a failure including compute node ID
fType	Classification of a failure based on its cause
time	Timestamp when a failure occurs
tbf	Time between successive failures
fCount	Number of failures in a subsystem 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 a subsystem
tmpCorr	Temporal correlation among failures in a subsystem

correlations among failure instances. We cluster failure events in a 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 is effective for clustering and analyzing the temporal and spatial correlations among failure events. The construction of an effective signature requires us to consider the hierarchical structure and interactions among components at different scopes of a 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 failure.
(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 decisions.

As a proof of concept, a prototype failure predictor based on quantified failure correlation was developed and has been in operation since May 2006 on a production coalition environment: the Wayne State Computational Grid [38]. The WSU Grid consists of three clusters located in three campus buildings of Wayne State University (WSU) and contains 40 high-performance compute servers in support of university-wide high-performance computing application programs. Offline and 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 that our prediction system can forecast 85.3% of the failure occurrences cluster-wide and system-wide, 72.7% in node-wide prediction, and capture the failure correlations in coalition clusters.

The rest of this paper is organized as follows. Section 2 presents the algorithms to explore temporal and spatial correlation among failure events. Section 3 describes the failure traces from the WSU Grid. The performance of offline and online prediction by our implemented prototype in the WSU Grid is evaluated in Section 4. Section 5 presents the related work and Section 6 summarizes the paper.

2. Quantifying temporal and spatial failure correlation

In this section, we address two key issues in characterizing failure correlation: (a) What representation should we use to describe failure instances and the associated system performance variables? (b) How do we cluster failure signatures to identify temporal and spatial correlations among failure occurrences and to utilize the correlations for proactive failure management?

Without loss of generality, when we refer to a "failure" in the following discussion, we mean any anomaly caused by a hardware defect, incorrect design, unstable environment or operator mistakes that makes services or compute nodes unavailable.

2.1. 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. Toward 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 system components.

By investigating the structure of networked computing systems and the correlations of failure occurrences in a system, we define the performance variables used in our failure management framework, as listed in Table 1. They are raw data collected from the system event logs or derived from the raw data. The runtime state 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.

To quantify the correlation among failure events in a networked computing 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 and correlation. 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.

2.2. Exploring temporal and spatial correlation

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 and obtain an idea of the underlying root 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 temporal locality among failure instances in the time domain. The other is to explore the causal dependency in the space domain. We design the clustering algorithms based on our observation from the Wayne State Computational Grid.

2.2.1. Temporal clustering

Studies in [29,27,22] found the skewness of the failure distribution in the time domain. Multiple failures may occur in a short time period. Liang [21] and Sahoo [26] used a fixed time window to classify failure patterns for all types of failure. In reality, the time-between-failure (*tbf*) may follow various distributions for different types of failure. We profile time-between-failure in the WSU Grid from its failure traces. Fig. 1 presents the cumulative distribution functions of *tbf* for the hardware, software and all 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. 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 cause several failure instances to occur 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.

Here, 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.

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 [2], 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 that the timers of compute nodes in a cluster are synchronized. The spherical covariance, $C_T(d)$, for temporal correlation is defined as

$$C_T(d) = \begin{cases} 1 - \alpha\dfrac{d}{\theta} + \beta\left(\dfrac{d}{\theta}\right)^3 & \text{if } 0 \le d \le \theta, \\ 0 & \text{if } d > \theta, \end{cases} \tag{2.1}$$

where θ is an adjustable timescale parameter for determining the temporal relevancy of two failure events, and α and β are positive constants with $\alpha = 1 + \beta$. We use different values of θ to quantify temporal correlations of different types of failure. For example, with $\theta = 3$ h for the WSU Grid, we can capture more than 40% of the software failures. For different types of failure, the value of θ varies. For other 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 are 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. Algorithm 1 presents the pseudo-code of correlating failure signatures in time by a master node of a cluster. 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 calculate the temporal correlation. Although the algorithm needs to scan failure signatures of every compute node in a cluster, the total number of failure events occurring within a time window is quite limited.

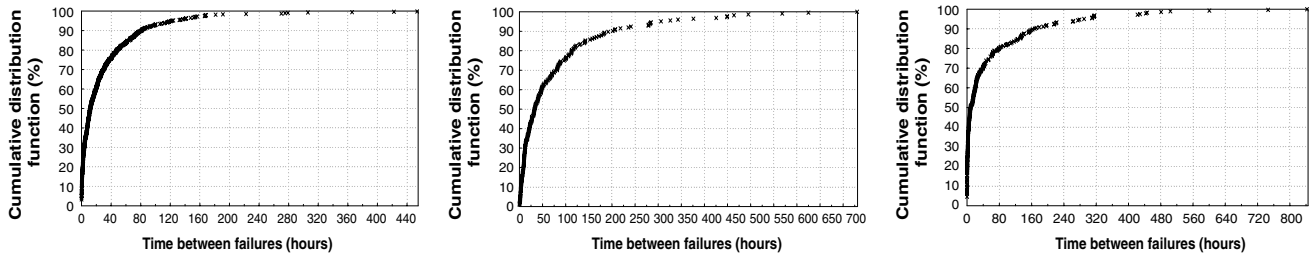
Algorithm 1 Temporal clustering of failure signatures

```
/* 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 the master node of a subsystem */
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:}
```

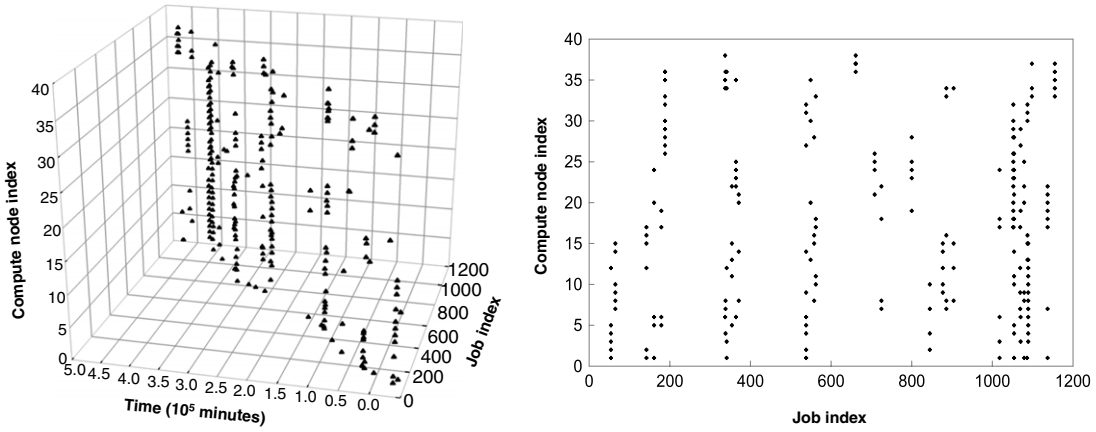
2.2.2. Spatial clustering

In addition to temporal locality, failure events are correlated in the space domain. Multiple failures occur on different compute nodes and intersect with each other.



(a) All failures. (b) Hardware failures. (c) Software failures.

Fig. 1. Temporal distribution of hardware and software failure events in the Wayne State Computational Grid from January 2, 2005 to January 6, 2006.



(a) Spatial distribution of application I/O failures. (b) Node allocations among grid jobs.

Fig. 2. Spatial distribution of failures in the Wayne State Computational Grid from January 2, 2005 to January 6, 2006.

Fig. 2(a) shows the spatial distribution of application I/O failures in the WSU Grid. From the figure, we can see some failure occurrences are clustered in space. By inspecting the system failure and performance traces, we found that

- (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 to occur 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 to fail. 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.

To find out what kinds of node are likely to experience spatially correlated failures, we inspected the job-scheduling information, the Portable Batch System (PBS) logs in the WSU Grid. From these logs, we found that grid nodes had been allocated to grand applications in groups. This also indicates that faults triggered by parallel and distributed applications may cause multiple nodes to fail almost simultaneously. For example, six out of ten nodes allocated to Job 538 failed on April 20, 2005. This job was to perform fast Fourier transformation (FFT) in an image-processing application. The bugs in the program caused the six nodes to experience endless loops. Reboot events were later found in their event logs. This indicates that failures in coalition clusters are spatially correlated, and the node allocation information is useful for predicting failure occurrences in the future.

We developed an aggregate stochastic model to cluster failure signatures in the space domain and used 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 the set $F = \{f_1, f_2, \ldots, f_m\}$ denote all possible types of failure that may occur in a coalition cluster 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

$$\hat{n}_i = \begin{cases} 1 & \text{if node } n_i \text{ fails in a unit interval,} \\ 0 & \text{otherwise;} \end{cases}$$

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

So, n_i and f_j indicate whether a node fails or a failure happens. A unit interval is a small period of time when only one failure event can appear on a node. The time window is measured in unit intervals. Based on the failure statistics, we measure the conditional probabilities $p(\hat{f}_j|\hat{n}_i)$; 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 us 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 $nodeFCount_i = \hat{n}_i \cdot w$, where w denotes the size

of the time window. If we fix the window size in measurements, then the expected number of failures becomes

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

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

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

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

$$E[clusterFCount] = w \cdot \left(\sum_i p(\hat{n}_i) - \sum_{i,k} \sum_j p(\hat{n}_i\hat{n}_k|\hat{f}_j)p(\hat{f}_j) + \cdots + \sum_j (-1)^r p(\hat{n}_1 \cdots \hat{n}_r|\hat{f}_j)p(\hat{f}_j) \right). \quad (2.2)$$

By using these probabilities of failure distributions along with the temporal correlation among failure signatures, predictors in node, cluster-wide 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 spatio-temporal 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 the following.

(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 (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.) and asymmetric (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) 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, the predictor estimates the probabilities $p(\hat{f}_j|\hat{n}_i)$ 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 $p(\hat{f}_j|\hat{n}_i)$. We set the length of an observation interval based on the measured mean-time-to-failure (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 is equal 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 the 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 the failure events of every compute node in a cluster, the total number of failure events occurring 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.

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: $n_i = f_{1,i} + f_{2,i} + \cdots + f_{m,i}$;
4: $p_{j,i} = f_{j,i}/n_i$;
5: $p_{f_j} = (f_{j,1} + f_{j,2} + \cdots + f_{j,r})/(n_1 + n_2 + \cdots + n_r)$;
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: $p_{n_{ij},f_k} = p_{k,i} * p_{k,j} * p_{n_i} * p_{n_j}/p_{f_k}^2$;
10: $S[i, j] = p_{n_{ij},f_1} * p_{f_1} + \cdots + p_{n_{ij},f_m} * p_{f_m} + p_i * p_{m_{ij}}$;
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:}

2.3. Proactive failure management

Occurrences of failures are quite dynamic in networked computing systems. 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 different scopes of a system, as

$$\begin{aligned} &\mathcal{F}_{node}(w, fCount_n, perf_n, tmpCorr_n) = 0, \\ &\mathcal{F}_{cluster}(w, fCount_c, perf_c, sptCorr_c, tmpCorr_c) = 0, \\ &\mathcal{F}_{system}(w, fCount_s, perf_s, sptCorr_s, tmpCorr_s) = 0, \end{aligned} \quad (2.3)$$

where w is the look-ahead window size of interest, and *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*). $perf_c$ 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 Section 4.

In essence, predicting failures in a coalition system means finding the approximate function $\mathcal{F}$. Failure events are highly nonlinear 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 quantified temporal and spatial correlations (as presented in Section 2.2) are inputs to a statistical learning algorithm, which forecasts the failure dynamics in the next time window by exploring the correlation information. The prediction procedure can be expressed as follows:

$$x(w_{i+1}) = \mathcal{G}(x(w_i), x(w_{i-1}), \ldots, x(w_{i-k+1})), \tag{2.4}$$

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

In this way, failure correlations spanning across multiple windows are kept for failure prediction. In essence, for a window size w, we can maintain the correlation information of a period of $k * w$ by using the order-k predictor, while being able to make failure predictions at a granularity of w 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 feature of the prediction mechanism.

3. Trace collection and characterization

In this study, we use event/failure trace files collected from the Wayne State Computational Grid. First, we will briefly describe the architecture of the WSU Grid and the collected failure traces.

3.1. WSU computational grid

The WSU Grid consists of three Linux clusters, maintained by Computing and Information Technology (CIT), the Institute for Scientific Computing (ISC), and the Department of Chemistry (CHM), in three separate buildings. It contains 40 high-performance compute servers dedicated to computational research. The CIT and ISC clusters each consist of 16 nodes, and there are 8 nodes in the CHM cluster. Within each cluster, nodes are interconnected by gigabit Ethernet switches. Connections between clusters are through 100 Mbps 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 WSU students to execute their sequential and parallel programs.

3.2. Failure traces

The event log files constitute a continuous 369-day trace between January 2, 2005 and January 6, 2006 from the three clusters. In the raw trace files, there are 35,547 event entries recorded and gathered in the system. The event logs have a total size of 596 MB and the PBS logs occupy 4.29 GB. To extract relevant event entries for our failure analysis, we first screened out events with lower severity levels, such as the informational and warning entries, because they did not affect the availability of services and nodes. This step removes 97.5% of entries from the raw logs. Then, we eliminated the duplicate adjacent entries from the result logs. This results in a 18.3% reduction in the number of entries of data, leaving only 726 failures, which include the hardware failures of compute nodes and network, the fatal failures of application programs and operating systems, operator misoperation and undetermined failures.

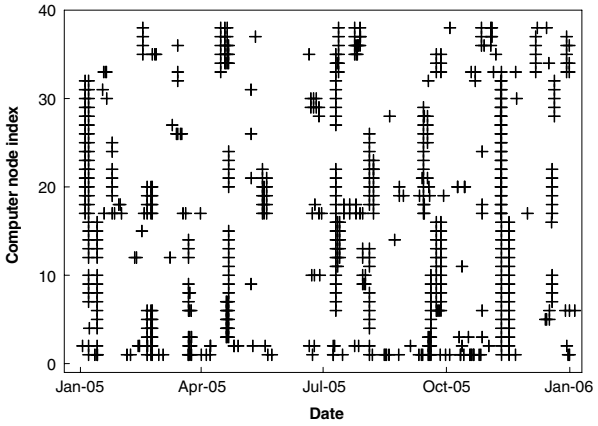


Fig. 3. Failure events as a function of time on the WSU Grid.

Among the remaining failure events, we observed four system maintenance operations conducted on June 3, 7, 9, and December 27, 2005, according to the administration logs. The first three were due to installing and debugging the PBS job-scheduling system, and the last one was for a system upgrade. After removing the events corresponding to these maintenance actions, we have the failure event distribution as shown in Fig. 3. Time is represented on the X-axis and the compute node index is on the Y-axis. A point is plotted each time a failure event occurs. The figure presents the burstiness of event occurrence and correlations of failures among compute nodes.

4. Experimental results

As a proof of concept, we developed a prototype system for failure correlation quantification and management in the WSU Grid. We implemented several illustrating prediction algorithms by applying four time-series algorithms and a specific statistical learning algorithm, the neural network, to learn and forecast failure dynamics. We used the gretl GNU time-series library [16] and the Weka machine learning software [40] in the implementation.

The primary metric we used for evaluation is the relative prediction error, which quantifies the discrepancy between the actual and predicted values.

$$err = \frac{|PredictedValue - ActualValue|}{ActualValue} * 100\%,$$

where *err* is the absolute value of the relative prediction error, *PredictedValue* is the number of failures predicted for the prediction window and *ActualValue* is the number of failures observed for that window, where the window size is a configurable parameter. The mean error which we used for our computations is calculated by averaging all of the relative prediction errors. We conducted failure predictions in the Wayne State Computational Grid and analyzed the accuracy of the system-wide, cluster-wide and node-wide predictions, by utilizing the temporal and spatial correlations among failure signatures quantified by the approaches presented in Section 2.2.

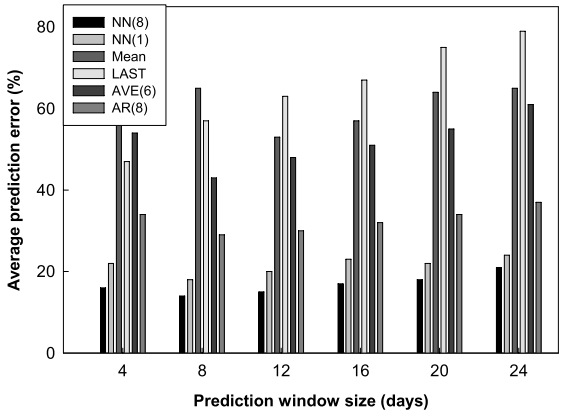


Fig. 4. System-wide prediction accuracy using a trace from the WSU Grid.

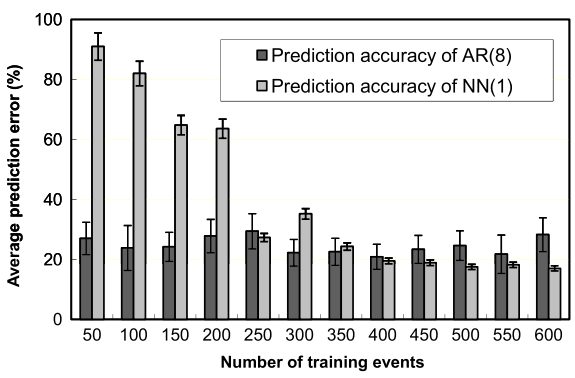
4.1. Offline prediction performance

Fig. 4 shows the performance of the system-wide predictions. For illustration, we include four popular time-series algorithms: MEAN, which takes the average of previous measures as the prediction; LAST, which uses the last measure; AVE(n), which uses the average of the last n measures; AR(m), which is autoregressive; and one artificial neural network algorithm NN(n), which uses the last n measures to update the neural network and to predict failures. The predictor neural network has four input neurons to receive utilization and failure measures; one output neuron for prediction result; three hidden layers and four neurons in each hidden layer. The predictor uses these illustrating algorithms to forecast the number of failure instances that may occur in a prediction window. The first half of the failure trace is used for training, while the other half is used for prediction. From the figure, we can see AR performs the best among the time-series algorithms. But its prediction accuracy is still worse than that of the neural-network-based predictors, because it does not characterize and adapt to the spatial correlations well in prediction. As described earlier, the high-order predictor considers the interrelation of failure measures in consecutive windows and feeds them as inputs to the network. The prediction error is reduced by 4.1% and the prediction accuracy reaches 85.3% when the prediction window size is 8 days.

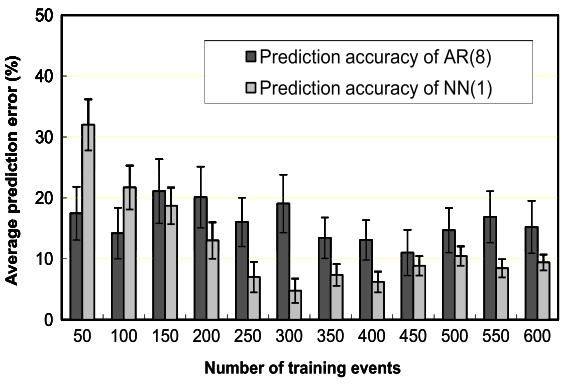
On each cluster, we deploy a failure predictor on the master node. This predictor collects cluster-wide failure signatures and clusters them to explore failure correlations. The prediction error *err* only presents the absolute value of the discrepancy between actual and predicted values. To provide more insight for performance analysis of our predictors, we distinguish two cases, where (a) the number of predicted failures is less than the number of actually observed failures in a prediction window; (b) the former is greater than the latter. We refer to Case (a) as *under-prediction* and to Case (b) as *over-prediction*. The relative prediction error of an under-prediction is calculated by $err = (ActualValue - PredictedValue)/ActualValue * 100\%$, and for over-prediction, it is the opposite. We conducted experiments to quantify the prediction error of under-predictions and over-predictions by using the NN failure predictor and the AR(8) on the three clusters. We measured the relative error of these two types of prediction. Fig. 5(a) and (b) plot the average prediction error of over-predictions and under-predictions made by the two predictors on the ISC cluster. 95% confidence intervals are included in the figures. From these figures, we can see both predictors provide accurate results and the neural-network-based approach is a little better. Failure predictions on the CIT and CHM clusters display similar performance to that on the ISC cluster.

To perform node-wide failure prediction, we cluster the failure signatures of a node according to the causes of the problems. Fig. 6 plots the distributions of inter-failure time of five categories of failure event: hardware, software, network, operator, and undetermined failures, on Node 1 of the ISC cluster. A diagnosis memo in the administrative logs of WSU Grid recorded the underlying root problem of each failure event. The right box of each group marks the quartiles of the distribution and the horizontal line inside each box is the median. From the figure, we can see that the average interval between consecutive failures caused by hardware, software, network, operator, and undetermined faults are 1721.8, 264.7, 1602.5, 2079.0, and 900.4 h, respectively. The time between software-caused failures is the least one among the five categories. An explanation for this skewed distribution of failures of different types in the time domain is that certain types of fault cause the majority of failures. For example, 60.7% failure events on the node in question were caused by software faults.

We cluster the failure signatures of each category for Node 1. Temporal and spatial correlations are quantified based on the signature groups. Then the predictor forecasts the number of prospective failure instances in the next prediction window for each category. We calculate a weighted sum using the prediction result of each category and the prediction accuracy of that category in the previous predictions. This weighted sum is returned as a failure prediction. Fig. 7 shows the prediction performance on Node 1 by using this weighted category approach. We notice that node-wide failure prediction is not as accurate as those in the cluster-wide and system-wide cases, due to the limited amount of failure events occurring on a compute node. We achieved up to 72.7% accuracy by using the NN(8) algorithm with the prediction window size equal to 6 days. Our failure prediction results are useful in coarse-grain job scheduling and node maintenance.



(a) Over-predictions (window size = 4 days).



(b) Under-predictions (window size = 4 days).

Fig. 5. Cluster-wide failure prediction on the ISC cluster.

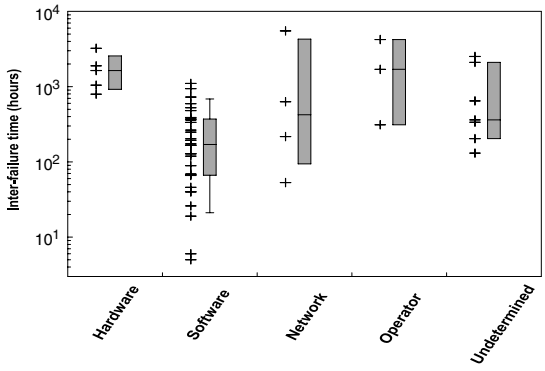


Fig. 6. Node-wide failure signature clustering for Node 1 based on the underlying root problems.

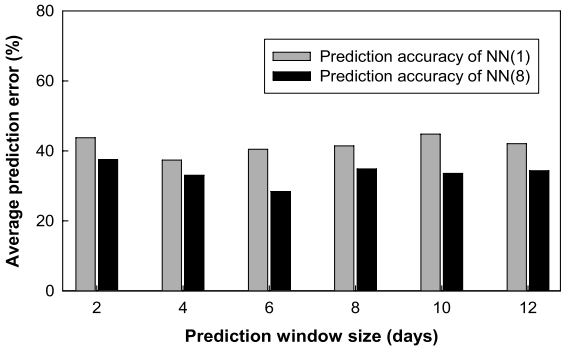


Fig. 7. Node-wide failure prediction on Node 1 of the ISC cluster.

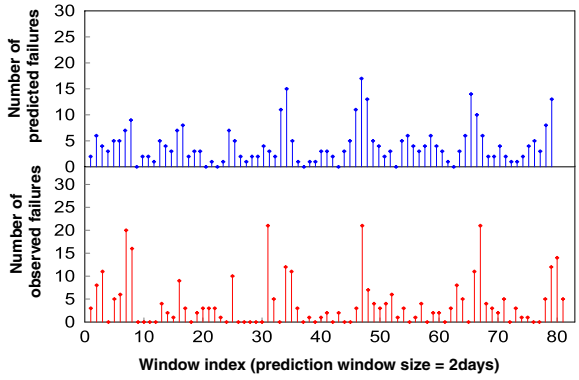


Fig. 8. Online system-wide failure prediction in comparison with observed failure events.

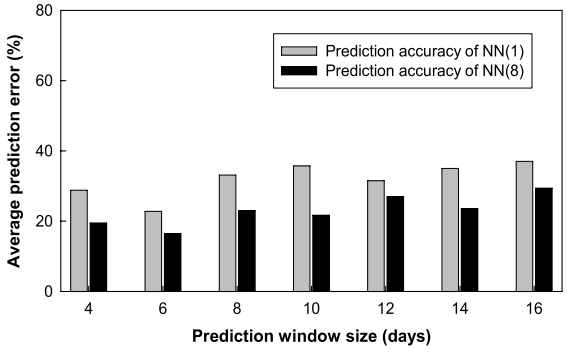


Fig. 9. Online cluster-wide prediction on the ISC cluster.

4.2. Online prediction performance

To evaluate the prediction performance in a real system at runtime, we installed our predictors on compute nodes of each clusters and their master nodes in the WSU Grid. By making online predictions, our failure predictors provide useful information for resource management and load distribution. In this experiment, first, we trained the predictors using failure event records between May 2005 and April 2006. Then, we evaluated the online prediction performance from May 12, 2006 to April 2, 2007. We record the failure predictions and compare them with the observed failure events mined from the event logs later.

Fig. 8 depicts predicted and observed failure events from the entire computational grid during the online prediction based on the NN algorithm. Temporal and spatial correlations among failure occurrences were utilized for prediction. The node allocation information was used to refine the correlations in predictions. From the figure, we can see the predictor can capture the trends of the failure dynamics. An exception is noticed at time $t = 32$, when 21 failure events were observed while only 4 was predicted. Later, we checked the administration logs and found that on July 15, 2006, in the time interval of $t = 32$, a problem with a switch caused network breakdown in the CIT cluster and its compute nodes were unavailable. To make a prediction, it took 2.17 s 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.

Fig. 9 shows the performance of online predictions in the ISC cluster. We used predictors based on the NN(1) and NN(8) algorithms for prediction. According to these figures, the accuracy of online predictions is a little worse than that of offline predictions. This is due to the variance among the limited number of prediction results. But still, the predictors provided useful information for autonomic management of the Grid. Online predictions on the CIT and CHM clusters display similar performance as that on the ISC cluster.

5. 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 real-time streams of computer events, and performing continuous monitoring of the runtime services. The core of *autonomic computing* [15,20] is the ability to analyze data in real time and to predict potential problems. The goal is to avoid catastrophic failures through prompt execution of remedial actions.

To realize proactive management of failures, it is imperative to understand the characteristics of failure behaviors. Research in [29,22,27,43] 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. [27] 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 [1,37,39] have examined system logs to identify causal events that lead to failures. Correlation between the workload intensity and the failure rate in real systems has been pointed out in many studies [5,23,25,19,3].

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. [42] 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. 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 [17] collected failure data from three different clustered servers, and used a Weibull distribution to model the time-between-failure. Both these studies [42,17] 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. Fu and Xu [14,13] analyzed and modeled the failure correlation in an institution-wide computational grid system and a large-scale high-performance cluster system. They also proposed failure-aware resource management mechanisms by using reconfigurable distributed virtual machines in networked computer environments [10,44,8,9].

There exist techniques for managing failures of specific components in production 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 provide, to help capture failure statistics of storage devices [18]. Data Lifeguard [6] and SIGuardian [31] are utilities to check and monitor the state of a hard drive, and derive the failure trend, to take proactive remedies. More recently, a Reliability Odometer [33] has been proposed for processors to track their wear-and-tear and forecast lifetimes.

Salfner et al. [28] proposed a Similar Events Prediction method and demonstrated its effectiveness using field data of a telecommunication system. Their approach focused on time-series-based failure prediction of single systems. In contrast, our approach emphasized on the impact of both temporal and spatial correlation of failure events in parallel computing systems. Challagulla et al. [4] applied machine learning techniques to predicting software defects. In our work, we apply statistical learning algorithms as an illustration to predict component failures in networked computer systems. The strong correlations among failure events result in better prediction precision by statistical learning. Besides, failure prediction aside, there are many works on failure detection [41,30,7,32]. These techniques are complementary to our failure prediction approaches in constructing a comprehensive failure management infrastructure.

There were recent works utilizing temporal and/or spatial correlations of failures for failure prediction and proactive management. Sahoo et al. [26] inspected the event set within a fixed time window before a target event for repeated patterns to predict the failure event of all types. Later, Liang et al. [21] 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 [24] 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. [26] 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. [21]. The authors analyzed the number of failures in every midplane of an IBM BlueGene/L supercomputer. They found skewness in the distribution of network failures only, among the midplanes.

6. Conclusions

In this paper, we have analyzed and quantified the correlation among failure events in networked computing 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 a production coalition system present the feasibility of applying failure prediction to autonomic management for high-availability network computing.

In our experiments on the WSU Grid and analysis of the failure data from HPC clusters in some national laboratories, we observed that both software failures and hardware failures display strong temporal and spatial correlations. The proposed approach in this paper can be applied to them in quantifying the correlations and forecast failure dynamics for specific types of failure. The quantification and prediction of failure correlation by our approaches are useful for coarse-grain scheduling, resource management, and system maintenance. As an ongoing work, we are analyzing our proposed approaches rigorously and deriving their properties theoretically. In this work, the predictor forecasts the number of failure instances in a look-ahead window, based on the temporal and spatial correlation among failure events. However, it does not predict the time point when a failure will occur. As a future work, we will analyze the occurrence time of failures and design a mechanism to predict when a failure is going to happen. To complement failure management, we proposed service migration [11] to realize system reconfiguration for mitigating the impact of failures and load-unbalance on the availability and performance of distributed virtual machines. We looked into the migration decision problem [12] in reconfigurable distributed virtual machines. As ongoing work, we are integrating failure prediction with the service migration mechanism and enhancing system availability by migrating runtime services when a failure is predicted to occur with high probability. In this work, we tested our proactive failure management mechanisms in the WSU Grid system. To our knowledge, our work is the first that analyzes the performance of online failure prediction in a production environment. As the scale of a system increases to thousands or millions of nodes, we need to deal with scalability and many other issues. We are working with some national laboratories and plan to evaluate the performance of our failure predictors in their large-scale computer systems.

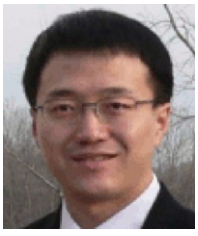
Acknowledgments

This research was supported in part by US NSF Grant CNS-0915396, LANL Grant IAS-1103, CNS-0702488 and CRI-0708232. The authors thanks Dr. Kai Hwang and anonymous reviewers for their constructive comments and suggestions. A preliminary version of this paper was presented in the Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems (SRDS'07) [14].

References

[1] H. Berenji, J. Ametha, D. Vengerov, Inductive learning for fault diagnosis, in: Proceeding of IEEE International Conference on Fuzzy Systems, 2003.
[2] J. Berger, V. Oliveira, B. Sansó, Objective Bayesian analysis of spatially correlated data, Journal of the American Statistical Association 96 (456) (2001) 1361–1374.
[3] X. CastiUo, D.P. Siewlorek, Workload, performance and reliability of digital computing systems, in: Proceeding of Symposium on Fault-Tolerant Computing, FTCS, 1981.
[4] V.U.B. Challagulla, F.B. Bastani, I.-L. Yen, R.A. Paul, Empirical assessment of machine learning based software defect prediction techniques, in: Proceeding of Workshop on Object-Oriented Real-Time Dependable Systems, 2005.
[5] B. Chun, A. Vahdat, Workload and failure characterization on a large-scale federated testbed, Technical Report IRB-TR-03-040, Intel Research Berkeley, 2003.
[6] Data lifeguard. Available at: http://www.wdc.com/en/library/2579-850105.pdf.
[7] J. Dunagan, N.J.A. Harvey, M.B. Jones, D. Kostic, M. Theimer, A. Wolman, FUSE: Lightweight guaranteed distributed failure notification, in: Proceeding of USENIX Symposium on Operating Systems Design and Implementation, OSDI, 2004.
[8] S. Fu, Failure-aware construction and reconfiguration of distributed virtual machines for high availability computing, in: Proceeding of IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid, 2009.

[9] S. Fu, Dependability enhancement for coalition clusters with autonomic failure management, in: Proceeding of IEEE International Symposium on Computers and Communications, ISCC, 2010.
[10] S. Fu, Failure-aware resource management for high-availability computing clusters with distributed virtual machines, Journal of Parallel and Distributed Computing 70 (4) (2010) 384–393.
[11] S. Fu, C.-Z. Xu, Service migration in distributed virtual machines for adaptive grid computing, in: Proceeding of the International Conference on Parallel Processing, ICPP, 2005.
[12] S. Fu, C.-Z. Xu, Stochastic modeling and analysis of hybrid mobility in reconfigurable distributed virtual machines, Journal of Parallel and Distributed Computing 66 (11) (2006) 1442–1454.
[13] S. Fu, C.-Z. Xu, Exploring event correlation for failure prediction in coalitions of clusters, in: Proceeding of ACM/IEEE Conference on Supercomputing, SC, 2007.
[14] S. Fu, 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.
[15] A.G. Ganek, T.A. Corbi, The dawning of the autonomic computing era, IBM Systems Journal 42 (1) (2003) 5–18.
[16] Gretl: GNU regression, econometrics and time-series library. Available at: http://gretl.sourceforge.net/.
[17] T. Heath, R.P. Martin, T.D. Nguyen, Improving cluster availability using workstation validation, in: Proceeding of ACM International Conference on Measurement and modeling of computer systems, SIGMETRICS, 2002.
[18] G. Hughes, J. Murray, K. Kreutz-Delgado, C. Elkan, Improved disk-drive failure warnings, IEEE Transactions on Reliability 51 (3) (2002) 350–357.
[19] R.K. Iyer, D. Rossetti, M. Hsueh, Measurement and modeling of computer reliability as affected by system activity, ACM Transactions on Computer Systems 4 (3) (1986) 214–237.
[20] J.O. Kephart, D.M. Chess, The vision of autonomic computing, IEEE Computer 36 (1) (2003) 41–50.
[21] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, R.K. Sahoo, BlueGene/L failure analysis and prediction models, in: Proceeding of IEEE International Conference on Dependable Systems and Networks, DSN, 2006.
[22] Y. Liang, Y. Zhang, A. Sivasubramaniam, R. Sahoo, J. Moreira, M. Gupta, Filtering failure logs for a BlueGene/L prototype, in: Proceeding of IEEE International Conference on Dependable Systems and Networks, DSN, 2005.
[23] J. Meyer, L. Wei, Analysis of workload influence on dependability, in: Proceeding of IEEE International Symposium on Fault-Tolerant Computing, FTCS, 1988.
[24] J. Mickens, B. Noble, Exploiting availability prediction in distributed systems, in: Proceeding of USENIX Symposium on Networked Systems Design and Implementation, NSDI, 2006.
[25] S. Mourad, D. Andrews, On the reliability of the IBM MVS/XA operating system, IEEE Transactions on Software Engineering 13 (10) (1987) 1135–1139.
[26] R.K. Sahoo, A.J. Oliner, I. Rish, et al. Critical event prediction for proactive management in large-scale computer clusters, in: Proceeding of ACM Conference on Knowledge Discovery and Data Mining, SIGKDD, 2003.
[27] R.K. Sahoo, A. Sivasubramaniam, M.S. Squillante, 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.
[28] F. Salfner, M. Schieschke, M. Malek, Predicting failures of computer systems: a case study for a telecommunication system, in: Proceeding of Workshop on Dependable Parallel, Distributed and Network-Centric Systems in Conjunction with International Parallel and Distributed Processing Symposium, 2006.
[29] B. Schroeder, G. Gibson, A large-scale study of failures in HPC systems, in: Proceeding of IEEE International Conference on Dependable Systems and Networks, DSN, 2006.
[30] E. Schuchman, 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] Siguardian. Available at: http://www.siguardian.com/.
[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, 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, R.K. Iyer, Impact of correlated failures on dependability in a VAX cluster system, in: Proceeding of IFIP Working Conference on Dependable Computing for Critical Applications, 1991.
[35] D. Tang, R.K. Iyer, S.S. Subramani, Failure analysis and modelling of a VAX cluster system, in: Proceeding of IEEE International Symposium on Fault-Tolerant Computing, FTCS, 1990.
[36] K. Vaidyanathan, R.E. Harper, S.W. Hunter, 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, S. Ma, Predicting rare events in temporal domains, in: Proceeding of IEEE International Conference on Data Mining, ICDM, 2002.
[38] Wayne State University, Grid computing. Available at: https://www.grid.wayne.edu/.
[39] G.M. Weiss, H. Hirsh, Learning to predict rare events in event sequences, in: Proceeding of ACM International Conference on Knowledge Discovery and Data Mining, SIGKDD, 1998.
[40] Weka: the University of Waikato, machine learning software in java. Available at: http://www.cs.waikato.ac.nz/ml/weka/.
[41] M. Wiesmann, P. Urban, X. Defago, An SNMP based failure detection service, in: Proceeding of IEEE International Symposium on Reliable Distributed Systems, SRDS, 2006.
[42] J. Xu, Z. Kalbarczyk, R.K. Iyer, Networked windows NT system field failure data analysis, in: Proceeding of Pacific Rim Symposium on Dependable Computing, PRDC, 1999.
[43] P. Yalagandula, S. Nath, H. Yu, P.B. Gibbons, S. Sesha, Beyond availability: towards a deeper understanding of machine failure characteristics in large distributed systems, in: Proceeding of USENIX WORLDS, 2004.
[44] Z. Zhang, S. Fu, Failure prediction for autonomic management of networked computer systems with availability assurance, in: Proceeding of IEEE International Workshop on Dependable Parallel, Distributed and Network-Centric Systems, in conjunction with IEEE International Parallel and Distributed Processing Symposium, IPDPS, 2010.



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

Cheng-Zhong Xu received his B.S. and M.S. degrees from Nanjing University in 1986 and 1989, respectively, and his Ph.D. degree in Computer Science from the University of Hong Kong in 1993. He is currently a Professor in the Department of Electrical and Computer Engineering of Wayne State University and the Director of Sun's Center of Excellence in Open Source Computing and Applications. His recent research interests are mainly in distributed and parallel systems, particularly in scalable and secure Internet services, autonomic cloud management, energy-aware task scheduling in wireless embedded systems, and high-performance cluster and grid computing. He has published more than 150 articles in peer-reviewed journals and conference proceedings in these areas. He is the author of Scalable and Secure Internet Services and Architecture (Chapman & Hall/CRC Press, 2005) and the leading co-author of Load Balancing in Parallel Computers: Theory and Practice (Kluwer Academic/Springer, 1997). He serves on five journal editorial boards, including IEEE Trans. on Parallel and Distributed Systems and J. of Parallel and Distributed Computing. He has been a program chair or general chair of a number of conferences, including Infoscale'08, EUC'08, and GCC'07. He is a recipient of the Faculty Research Award of Wayne State University in 2000, the President's Award for Excellence in Teaching in 2002, and the Career Development Chair Award in 2003. He is a senior member of the IEEE.