2013 IEEE 32nd International Symposium on Reliable Distributed Systems

Adaptive Anomaly Identification by Exploring Metric Subspace in Cloud Computing Infrastructures

Qiang Guan and Song Fu
Department of Computer Science and Engineering
University of North Texas
Denton, Texas 76203-5017, USA
E-mail: QiangGuan@my.unt.edu, Song.Fu@unt.edu

Abstract*—Cloud computing has become increasingly popular by obviating the need for users to own and maintain complex computing infrastructures. However, due to their inherent complexity and large scale, production cloud computing systems are prone to various runtime problems caused by hardware and software faults and environmental factors. Autonomic anomaly detection is a crucial technique for understanding emergent, cloud-wide phenomena and self-managing cloud resources for system-level dependability assurance. To detect anomalous cloud behaviors, we need to monitor the cloud execution and collect runtime cloud performance data. These data consist of values of performance metrics for different types of failures, which display different correlations with the performance metrics. In this paper, we present an adaptive anomaly identification mechanism that explores the most relevant principal components of different failure types in cloud computing infrastructures. It integrates the cloud performance metric analysis with filtering techniques to achieve automated, efficient, and accurate anomaly identification. The proposed mechanism adapts itself by recursively learning from the newly verified detection results to refine future detections. We have implemented a prototype of the anomaly identification system and conducted experiments in an on-campus cloud computing environment and by using the Google datacenter traces. Our experimental results show that our mechanism can achieve more efficient and accurate anomaly detection than other existing schemes.

***Keywords*—Cloud computing; Dependable systems; Failure detection; Autonomic management; Learning algorithms.**

I. INTRODUCTION

Modern cloud computing systems contain hundreds to thousands of computing and storage servers. Such a scale, combined with ever-growing system complexity, is introducing a key challenge to failure and resource management for dependable cloud computing [4]. Despite great efforts on the design of ultra-reliable components [5], the increase of cloud scale and complexity has outpaced the improvement of component reliability. Results from recent studies [41], [28] show that the reliability of existing data centers and cloud computing systems is constrained by a mean time between failure on the order of 10-100 hours. Failure occurrence as well as its impact on cloud performance and operating costs is becoming an increasingly important concern to cloud designers and operators [4], [41]. The success of cloud computing will depend on its ability to provide dependability at scale.

With the ever-growing complexity and dynamicity of cloud systems, autonomic failure management is an effective approach to enhancing cloud dependability [4]. Anomaly detection is a key technique [7]. It identifies activities that do not conform to expected, normal cloud behavior. The importance of anomaly detection is due to the fact that anomalies in the cloud performance data may translate to significant and critical component or system failures, which disrupt cloud services and waste useful computation. Anomaly detectors provide valuable information for resource allocation, virtual machine reconfiguration and cloud maintenance [35].

Efficient and accurate anomaly detection in cloud systems is challenging due to the dynamics of runtime cloud states, heterogeneity of configuration, nonlinearity of failure occurrences, and overwhelming volume of performance data in production environments. Principal component analysis (PCA) is a well-know dimensionality reduction technique [9]. It performs linear transformation to map a set of data points to new axes (i.e., principal components). Each principal component has the property that it points in the direction of maximum variance remaining in the data. It has been widely used in feature extraction and network anomaly detection [24], [6], [25], [38], [12]. However, indiscriminately choosing the first several principal components does not always yield desirable accuracy of anomaly detection, as different types of failures display different correlations with cloud performance metrics (An illustrating example is presented in Section II.). A more subtle analysis of the relation between principal components and failure types is necessary in order to develop efficient and accurate anomaly identification mechanisms in cloud systems.

In this paper, we present an adaptive mechanism that explores PCA to identify anomalous behaviors in cloud computing systems. Different from existing PCA-based detection approaches, our proposed mechanism characterizes cloud health dynamics and finds the most relevant principal components (MRPCs) for each type of possible failures. The selection of MRPCs is motivated by our observation in experiments that higher order principal components possess strong correlation with failure occurrences, even though they maintain less variance of the cloud performance data. By exploiting MRPCs and learning techniques, we propose an adaptive anomaly detection mechanism in the cloud. It adapts its anomaly detector by learning from the newly verified detection results to refine future detections. We have implemented a prototype of the anomaly identification system and conducted extensive experimental study in a 362-node cloud computing environment on campus. We compare the anomaly detection accuracy of several algorithms using the receiver operating characteristic (ROC) curves. The experimental results show that the proposed

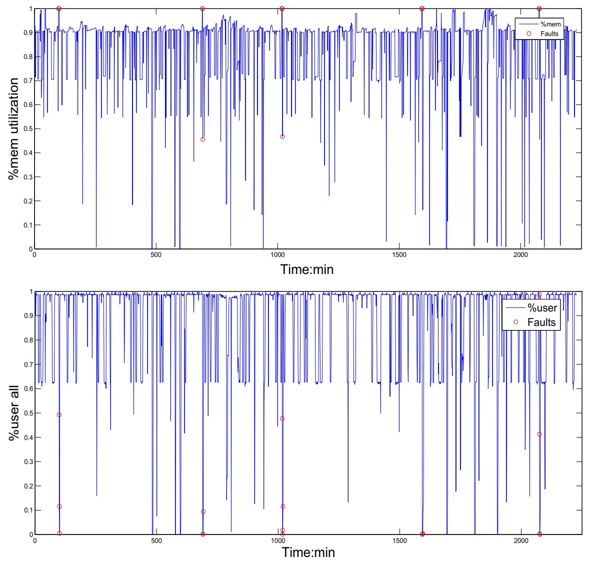


Fig. 1. Examples of memory related faults injected to a cloud testbed. The memory utilization and CPU utilization time serials are plotted.

mechanism can achieve 91.4% true positive rate while keeping the false positive rate as low as 3.7%. We also conduct experiments by using traces from a Google datacenter. Our MRPC-based anomaly detection mechanism performs well on the traces from the production system with the true positive rate reaching 81.5%. Our mechanism is lightweight as it takes only several seconds to initialize the detector and a couple of seconds for adaptation and anomaly identification. Thus, it is well suitable for building dependable cloud computing systems.

The rest of the paper is organized as follows. In Section II, we present an example to illustrate limitation of the traditional principal components based anomaly detection approaches. In Section III, we describe the proposed most relevant principal component approach. The experiment settings and data collection process are described in Section IV. Experimental results are presented in Sections IV-B, IV-C and IV-D. We discuss the related research in Section V. Finally, the paper concludes in Section VI.

II. A Motivating Example

Virtualization plays a key role in cloud computing infrastructures, because it makes possible to significantly reduce the number of servers in data centers by having each server host multiple independent virtual machines (VMs), which are managed by a virtual machine monitor (VMM) often referred to as a hypervisor [39]. By enabling the consolidation of multiple applications on a small number of physical servers, virtualization promises significant cost savings resulting from higher resource utilization and lower system management costs.

However, virtualization also complicates the interactions among applications sharing the physical cloud infrastructure, which is referred to as multi-tenancy. The inability to predict such interactions and adapt the system accordingly makes it difficult to provide dependability assurance in terms of

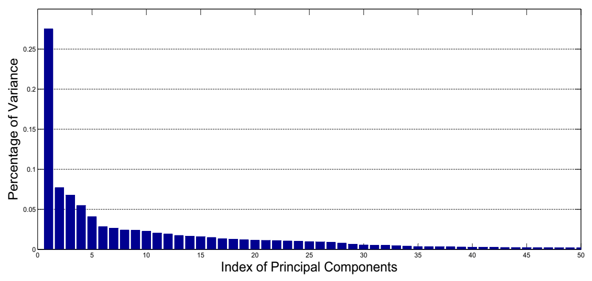


Fig. 2. Distribution of data variance retained by the first 50 principal components.

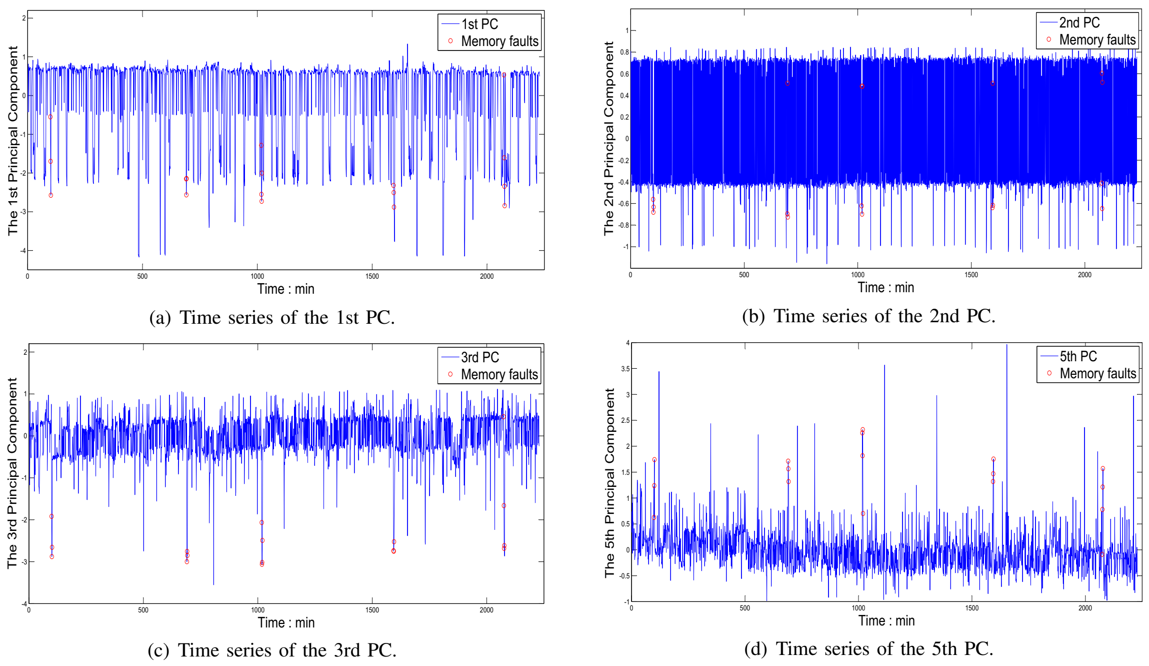
availability and responsiveness to failures. Virtualization and multi-tenancy introduce new sources of failure degrading the dependability of cloud computing systems and making anomaly identification more challenging.

As an example, Figure 1 depicts several memory-related failures, which are denoted by red circles, observed on a cloud testbed. The corresponding CPU utilization and memory utilization profiled from the testbed are also shown. From the figure, we can see these failures are difficult to identify if single performance metric is explored by approaches such as [42]. To improve anomaly detection accuracy, more metrics should be considered. Cloud computing systems are large-scale and complex. Hundreds and even thousands of performance metrics are profiled periodically [4], [12]. The large metric dimension and the overwhelming volume of cloud performance data dramatically increase the processing overhead and reduce the anomaly detection accuracy. Feature extraction techniques have been widely applied to reduce the metric dimensionality. Among these techniques, principal component analysis (PCA) is a well-known one, which extracts a smaller set of metrics while retaining as much data variance as possible [9].

Figure 2 shows the distribution of variance retained by the first 50 principal components (PCs) for the memory related failures. In total, 651 performance metrics are profiled on the cloud testbed. The 1st and 2nd PCs keep 27% and 8% of data variance, respectively, while the 3rd and 5th PCs retain only less than 5% of data variance each. Their time series are plotted in Figure 3. Figures 3(a) and 3(b) show that the low order PCs are less significant for identifying failures even though they retain a large portion of the data variance. Some failure events are overwhelmed by normal data records with high resource utilization and some are indistinguishable from noises. In contrast, the higher order PCs, such as the 3rd and 5th PCs in Figures 3(c) and 3(d), manifest stronger correlation with the failure events. We call these PCs the most relevant principal components (MRPCs) for a failure type. They are usually higher order PCs, which are not explored by existing anomaly detection techniques, while they provide important insights of failure occurrences. In the following sections, we discuss how to find the MRPCs and exploit MRPCs to identify anomalies in cloud computing systems.

III. MRPC-Based Adaptive Cloud Anomaly Identification

To find the *most relevant principal components* (MRPCs) and identify cloud anomalies, we explore the runtime performance data profiled from multiple layers of cloud servers.



(a) Time series of the 1st PC.

(b) Time series of the 2nd PC.

(c) Time series of the 3rd PC.

(d) Time series of the 5th PC.

Fig. 3. Time series of principal components and their correlation with the memory-related faults.

We install profiling instruments in hypervisors and operating systems in virtual machines to measure performance metrics from the hardware performance counters, resource utilization and critical events from the hypervisors and virtual machines (VMs), while user applications are running in VMs. To capture the dynamics of cloud systems, we employ a rolling window to analyze the profiled cloud performance data and dynamic normalization to pre-process the data. Then, MRPCs will be selected from cloud performance principal components by examining their correlation with failure events. In this section, we present the three key parts of our proposed MRPC-based adaptive cloud anomaly identification mechanism, i.e., *dynamic normalization*, MRPC *selection*, and *adaptive anomaly identification*.

A. Dynamic Normalization

The cloud performance metrics are profiled from various components at runtime. The collected data include performance and runtime states of hardware devices, hypervisors, virtual machines and applications. The unit and scale vary among these metrics. To explore them in cloud anomaly detection, normalization is necessary.

In our work, we employ a *dynamic normalization* approach with a rolling window to pre-process the cloud performance data and scale the metrics to $[0, 1]$. Let $X(t)$ represent a vector of M cloud performance metrics $(m_1(t), m_2(t), ..., m_M(t))$. For $X(t)$ at time t in a rolling window of size N, the dynamic normalization process is described as follows.

$$\overline{X_N(t)} = \begin{cases} \dfrac{X(t) - min_N(t)}{max_N(t) - min_N(t)} & \text{if } max_N(t) \neq min_N(t) \\ 0 & \text{if } max_N(t) = min_N(t), \end{cases} \tag{1}$$

where $max_N(t)$ and $min_N(t)$ denote the synthetic maximum and minimum values of a cloud performance metric in a time window, which is ended at time t. They are updated iteratively based on $max_N(t-1)$ and $min_N(t-1)$ and the new measurement of the metric by following the equations below.

$$max_N(t+1) = max_N(t)\lambda e^{X(t+1) - max_N(t)}, \tag{2}$$

$$min_N(t+1) = min_N(t)\lambda e^{X(t+1) - min_N(t)}, \tag{3}$$

where the coefficient λ represents the adjustment ratio. It controls the ascending or descending rate. For a new measurement of the metric as the time window moves forward, if the measurement is smaller than the current maximum, the maximum value decreases. If the measurement is larger than the current minimum, the minimum value increases. In such a way, the measurements close to the end of a rolling window have less impact to the normalized value than the measurements at the head of the time window.

B. MRPC Selection

The most relevant principal components (MRPCs) are the set of PCs which have strong correlation with occurrences of failures. For each failure type, we select a corresponding set of MRPCs. As described in Section III-A, we employ the dynamic normalization approach to re-scale performance metrics in a rolling time window. To adapt to the dynamics in a time window and at the same time consider the values of cloud performance metrics in previous windows, we propose an adaptive learning approach that exploits neural networks to compute principal components (PCs) from normalized values of cloud performance metrics in time windows. The approach proceeds as follows.

1) Initialize a neural network $\Re^m \rightarrow \Re^l$ with small

random synaptic weights w_{ji}, j and i are the index of input neural and output neural at time $t = 1$. Assign a small learning rate ε.

2) For $t = 1, j = 1, 2, ...l$ and $i = 1, 2, ..., m$ calculate $y_j(t) = \sum_{i=1}^{m} w_{ji}(t)x_i(t)$ and $\triangle w_{ji}(t) = \varepsilon(y_j(t)x_i(t) - y_j(t)\sum_{k=1}^{j} w_{ki}(t)y_k(t))$ where $x_i(t)$ is the ith component of the m-by-1 input vector $\mathbf{x}(t)$ and l is the dimension of principal components space.

3) Increment t by 1, repeat step 2 until the synaptic weight w_{ji} converges to the ith component of the eigenvector associated with the jth eigenvalue of the correlation matrix of the input vector $\mathbf{x}(t)$.

Then, the MRPC selection process decomposes the PCs in set S into k subsets $S_{sub}(f_i)$(one for each failure type) and a noise subset S_{noise}.The decomposition can be expressed as:

$$S = S_{sub}(f_1) \cap S_{sub}(f_2) \cap ... \cap S_{sub}(f_k) \cap S_{noise} \quad (4)$$

Each failure specific subset characterizes a failure type f_i. There may be intersections between two subsets, as some PCs are correlated with multiple failure types. The noise subset consists of PCs that are not correlated with any failure type. They can be represented by Gaussian noise. Algorithm 1 shows the pseudocode of the MRPC selection process.

Algorithm 1: **MRPC Selection Algorithm**

```
MRPCSelect() {
1:     $l$ = dimension of principal component space;
2:     $k$ = number of failure types;
3:     $f_i$ = failure type $i$;
4:     $N$ = time window size;
5:     $coeff_{ji}$ = correlation coefficient of $Y_j$ with $f_i$;
6:     $S_{sub(f_i)} = f_i$ specific PC subset;
7:     $Y_j(t,N) = (y_i(t-N), y_i(t-N-1), ..., y_i(t))$;
8:     $Lable_i(t,N) = (L_i(t-N), Li(t-N-1)..., Li(t))$;
9:     $S_{sub(f_i)} = \emptyset$;
10:    for $j = 1, 2, ..., l$ do
11:        for $i = 1, 2, ..., k$ do
12:            $coeff_{ji} = correlation(Y_j(t,N), Lable_i(t,N))$;
13:            if $coeff_{ji} < \sigma$ do
14:                $S_{sub(f_i)} = S_{sub(f_i)} \cap j$;
15:            end if
16:        end for
17:    end for
18: }
```

C. Adaptive Cloud Anomaly Identification

To identify anomalies using MRPCs, we leverage adaptive Kalman filters [22], because they are dynamic and do not require any prior failure history. A Kalman filter is optimized to achieve the best estimation of the next state. As the rolling window moves forward, the Kalman filter detects anomalies based on the corresponding cloud performance data record and the prior states by following Equation (5). Then it uses the measurement and the estimation to update the uncertainty for next time window by employing Equation (6) adaptively. Each iteration consists of anomaly detection and correction.

The anomaly detector identifies anomalous cloud behaviors by applying the following model.

$$\begin{cases} X_t^- = \Phi X_{t-1} \\ P_t^- = \Phi P_{t-1} + Q \end{cases} \quad (5)$$

The correction phase can be presented as follows.

$$\begin{cases} K_t = \dfrac{P_t^-}{P_t^- + R} \\ X_t = X_t^- + K_t(M_t - \Psi X_t^-) \\ P_t = (1 - K_t\Psi)P_t^- \end{cases} \quad (6)$$

where X_t^- and X_t denote the prior state and posteriori state at time t; P_t^- and P_t are the prior and posterior error covariance. Parameter Φ and Ψ are detector variables. K_t is the ratio, which is to control the difference between the prior state and the posteriori state. M_t represents the measurement of time t. Q and R are variances of the estimation noise and measurement noise respectively.

If the difference $|X_t - M_t|$ between the measurement and the estimation of MRPCs at a time point is greater than a threshold, an anomaly is detected and it is reported to cloud operators. The cloud operators check the identified anomalies to verify them as either true failures or false alarms. Those records of true failures are used to update the corresponding anomaly detector model following Equation (6). The cloud operators also input those observed but undetected failure records to the anomaly detector to generate a new MRPC subset. Algorithm 2 presents the adaptive anomaly detection process.

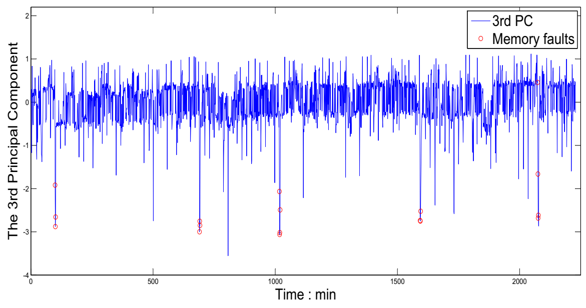
Algorithm 2: **Adaptive Anomaly Identification Algorithm**

```
AnomalyDetector() {
1:    while(TRUE) do
2:        On receiving a cloud performance record $x_t$
3:            if $|x_t - M_t| > \tau$ then
4:                Report the anomaly state;
5:            end if
6:        On receiving a verified failure or an observed
              but undetected failure record
7:        MRPCSelect();
8:    end while
9: }
```

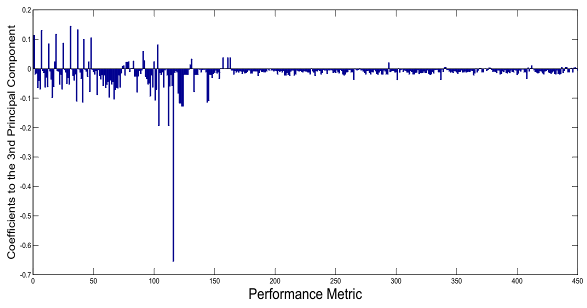
IV. Performance Evaluation

A. Experiment Settings

Our experimental cloud testbed consists of 362 servers, which are divided into several rack clusters of 40 nodes and are connected by 10 GB Ethernet inter-rack and 1 GB Ethernet inner-rack. Two rack clusters are configured to run Apache Hadoop MapReduce benchmark suite [1] and the rest run RUBiS benchmark program. Each cloud server hosts up to four virtual machines. Each server is equipped with a quad-cores 2.8GHz Intel Xeon processor with two hyperthreads for each core, 8 GB of RAM and 1 TB of storage. The underlying hypervisor is Xen 3.1.2. A virtual machine image is configured



(a) Time series of the 3rd principal component



(b) Coefficients of performance metrics for the 3rd principal component

Fig. 4. MRPCs of memory-related failures. (a) plots the time series of the 3rd principal component. (b) shows the performance metric *avgrq-sz* displays the highest contribution to the MRPC.

with two virtual processors and 512 MB memory space. The cloud workloads are kept in a high and well-balanced level within cloud for good resource utilization and energy saving.

We exploit third-party monitoring tools, sysstat [2] to collect runtime performance data in the hypervisor and virtual machines, and a linux performance counters profiler perf [3] to profile performance counters from the Xen hypervisor on each cloud server. In total, 518 metrics are profiled, i.e., 182 from the hypervisor, 182 from virtual machines by sysstat and 154 from performance counters every minute. They cover the statistics of every component of a cloud server, including the CPU usage, process creation, task switching activity, memory and swap space utilization, page faults, interrupts, network activity, I/O and data transfer, power management and so on. We collected the performance data from the cloud testbed for two months. In total, about 520 GB performance data were collected and recorded from the cloud computing testbed during the period. The rolling time window is set as 1440 records for a day.

We have also developed a fault injection program [16], which is able to randomly inject four major types with 17 sub-types of faults to cloud servers. They mimic faults from CPU, memory, disk, and network.

B. Experimental Results of MRPC Selection

In this section, we study the four types of failures caused by CPU-related faults, memory-related faults, disk-related faults, and network-related faults. For each failure type, we present the experimental results on MRPC selection.

1) MRPCs for Memory Related Failures: Figure 8(a) shows the correlation between PCs and the memory related faults. As we have discussed, the 1st and 2nd principal components do not possess high causal correlation with the occurrences of failures (only 0.16 and 0.04). This indicates the memory-related failures have little dependency upon them. On the contrary, the 3rd, 5th, 8th and 31th PCs display high correlation with failure records (greater than 0.2) as listed in Table I. Figure 4(a) shows that the 3rd PC displays a significantly identifiable performance to distinguish the failure states from normal states.

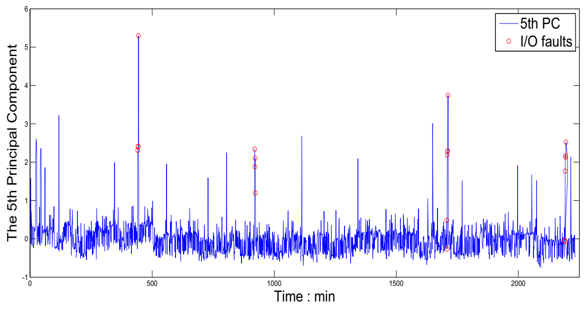
Based on the procedure described in Section III-B, the synaptic weight w_{ji}s represent the quantified impact from the original space to the anomaly specific subsets. Considering

TABLE I. MRPCs WITH THE RANKED CORRELATION WITH FOUR TYPES OF FAILURES.

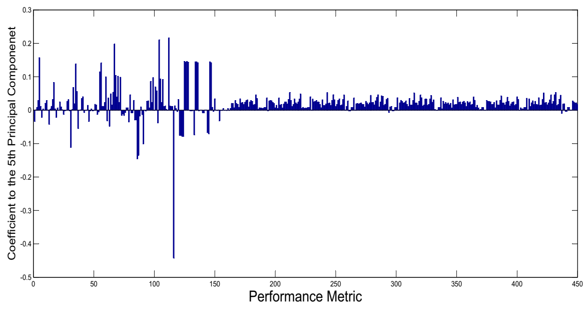
Failure Type	Most Relevant Principal Components		
	Rank	Order of PC	Correlation with failures
Memory-related failures	1	3rd	0.3898
	2	5th	0.2840
	3	8th	0.2522
	4	31st	0.2043
Disk-related failures	1	5th	0.4283
	2	7th	0.2961
	3	3rd	0.2402
CPU-related failures	1	35th	0.3738
	2	40th	0.3424
	3	103rd	0.2559
Network-related failures	1	29th	0.3532
	2	27th	0.2733
	3	23rd	0.2715

these synaptic weight w_{ji}s could be either positive or negative, we exploit $|w_{ji}|$ to identify the effect of each performance metric's contributing to the anomaly specific subsets. These weights computed are shown in Figure 4(b) for the 3rd principal component, which is selected as the top ranked MRPC with regard to memory related failures. In addition, one performance metric has a dominant contribution to this MRPC, as weight 0.65. Within it, multiple performance metrics have weights around 0.1-0.2. By checking the performance metrics list, we find that the highly weighted metric is "avgrq-sz dev253-1", which is "the average size (in sectors) of the requests that were issued to the hard drive device 253-1 [2]". Given that the memory related failures are injected by keeping allocating memory in a short period, after the physical memory is used up, the swap memory is put into use. As a result, this process will cause more requests issued to the hard disk.

2) MRPCs for Disk Related Failures: Disk-related faults are injected by continuously issuing a big volume of disk requests to saturate the I/O bandwidth. The causal correlation with the disk related failures is computed for each principal component, which is shown in Figure 8(b). The top ranked MRPCs are listed in Table I. The 5th PC possesses the highest correlation with the disk-related failures as its casual correlation is more than 0.42. Analysis of the time series of the 5th principal component, plotted in Figure 5(a), shows that most of the anomalies could be identified by setting a proper threshold. From Figure 5(b), the performance metric named "rd-sec/s dev-253", with the coefficient of 0.4423, contributes to the 5th principal component more than other performance metrics. It refers to "The number of sectors read from the

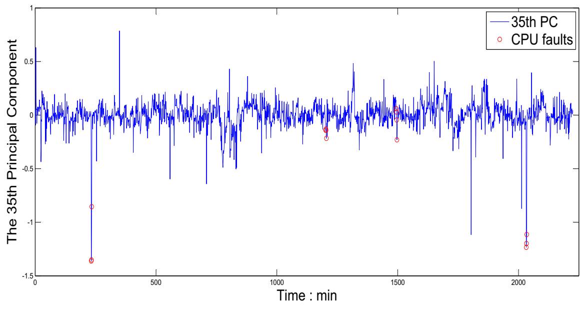


(a) Time series of the 5th principal component

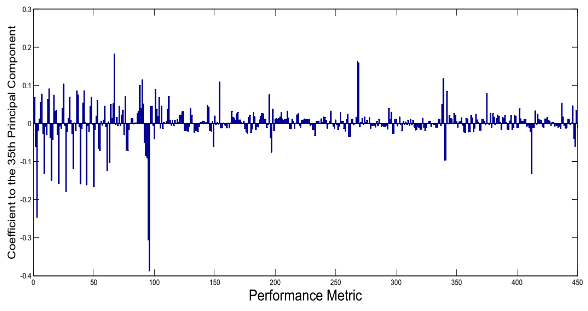


(b) Coefficients of performance metrics for the 5th principal component

Fig. 5. MRPCs of disk-related failures.(a) plots the time series of the 5th principal component. (b) shows the performance metric *rd-sec/s dev-253* displays the highest contribution to the MRPC.



(a) Time series of the 35th principal component



(b) Coefficients of performance metrics for the 35th principal component

Fig. 6. MRPCs of CPU-related failures.(a) plots the time series of the 35th principal component. (b) shows the performance metric *ldavg* displays the highest contribution to the MRPC.

device", which is an indicator to characterize the symptom of disk-related failures.

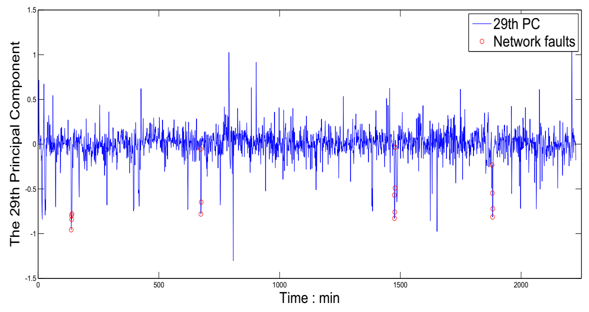
3) MRPCs for CPU Related Failures: CPU-related faults are injected by employing infinite loops that use up all CPU cycles. Table I lists MRPCs with the highest correlation with the CPU-related failures. Figure 6(a) presents the time series of the 35th principal component. From the figure, we can see some CPU-related failures are not easily identifiable, e.g., the failures occurred around 1250th minute and 1500th minute. Figure 6(b) plots weights of the cloud performance metrics for the 35th principal component. With largest weight of 0.3874, "ldavg-15" refers to "The load average calculated as the average number of runnable or running tasks (R state), and the number of tasks in uninterruptible sleep (D state) over past 15 minutes". The second and third largest weights correspond to the performance metrics "ldavg-5" and "%sys all" respectively. "%sys all" refers to "The average CPU utilization for system operations over all processors". All of the three performance metrics characterize the process behavior under failures.

4) MRPCs for Network Related Failures: Network-related faults are injected by saturating the network bandwidth by continuously transferring large files between servers. In cloud computing systems, denial-of-service attacks, virus infections, and failure of switches and routers may cause this type of anomalies. The MRPCs are listed in Table I. The 29th principal component is highly correlated with the network-related failures. The 27th and 23rd principal components are ranked as the second and third as shown in Figure 8(d). In Figure 7(a), we can see the 29th principal component is sensitive to the occurrences of network-related failures and the failure is distinguishable from normal states. Figure 7(b) shows that "%usr 4" possesses the highest weight of 0.292. The second and third highest weights are associated with performance metrics "%idle 4" and "svctm dev8-0" (i.e., "The average service time (in milliseconds) for I/O requests that were issued to the device."). Both %usr 4 and %idle 4 represent the states of processor core 4, which is assigned to the virtual machine where the network-related faults are injected. Therefore, MRPCs can assist cloud operator not only to identify anomalies, but also to localize faults even in virtual machines.

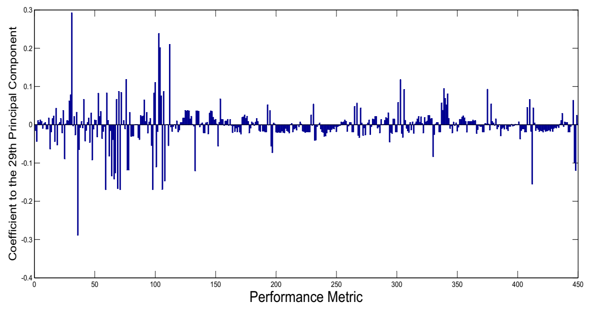
C. Experimental Results of Anomaly Identification

We study the performance of several anomaly detection techniques including our proposed MRPC-based detection approach. We use the receiver operating characteristic (ROC) curves to present the experimental results. An ROC curve displays the true positive rate (TPR) and the false positive rate (FPR) of the anomaly detection results. The area under the curve is used to evaluate the detection performance. A larger area infers higher sensitivity and specificity.

We compare the performance of the proposed MRPC-based anomaly detection approach with four widely used detection algorithms: decision tree, Bayesian network, support vector machine (SVM), and principal component analysis (PCA). The results are presented in Figure 9. Our MRPC-based anomaly detector achieves the best performance, with the true positive rate reaching 91.4% while keeping the false positive rate as



(a) Time series of the 29th principal component



(b) Coefficients of performance metric for the 29th principal component

Fig. 7. MRPCs of network-related failures.(a) plots the time series of the 29th principal component. (b) shows the performance metric *%user 4* displays the highest contribution to the MRPC.

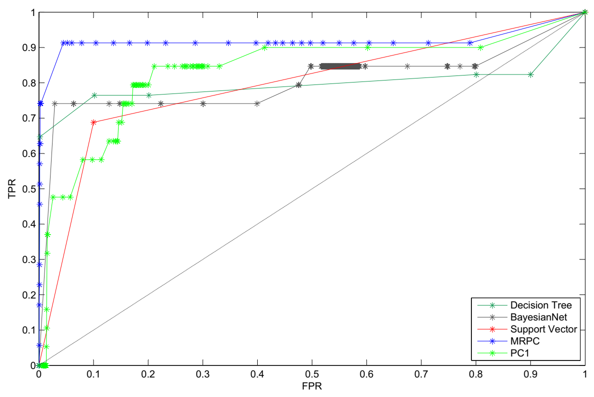


Fig. 9. Performance of the proposed MRPC-based anomaly detector compared with four other detection algorithms.

low as 3.7%. By applying only the first principal component, the false positive rate is higher than 40% in order to achieve a high true positive rate. Among other detection algorithms, the Bayesian network is relatively better, reaching 74.1% of TPR with a low FPR.

To make an anomaly detection, it takes 6.81 seconds on average for a control node in the cloud to process cloud performance data, select MRPCs, and make anomaly detections.

D. Experimental Results on Google Datacenter Traces

In addition to the experiments on our cloud computing testbed, we evaluate the performance of the proposed MR-PCbased anomaly detection mechanism by using the performance and events traces collected from a Google datacenter [34]. The Google datacenter trace is the first publicly available dataset collected from a large number of (about 13000) multipurpose servers over 29 days. In the dataset, multiple taskrelated events are recorded. Among them, we focus on the failure events. In total, there are 13 resource usage metrics profiled periodically, which are listed in Table II. By applying the MRPC selection algorithm presented in Section III-B, we obtain the casual correlation between principal components and failure events, which is plotted in Figure 10. The 13th principal component retains the highest correlation (i.e., 0.18) with the failures.

TABLE II. PERFORMANCE METRICS INCLUDED IN THE GOOGLE DATACENTER TRACES.

Index	Performance Metrics
1	Number of running tasks
2	CPU rate
3	Canonical memory usage
4	Assigned memory usage
5	Unmapped page cache
6	Total page cache
7	Maximum memory usage
8	Disk I/O time
9	Local disk space usage
10	Maximum CPU rate
11	Maximum disk I/O time
12	Cycles per instruction
13	memory accesses per instruction

The ROC curves in Figure 11 show the performance of the proposed anomaly detection approach and other four detection algorithms. By exploiting MRPC, we can achieve 81.5% of TPR and 27% of FPR. The results outperform all other detection methods by 22.9% - 68.7% of TPR with the same value of FPR. Compared with the results in Section IV-C, the performance of the proposed anomaly identification mechanism is a little worse on the Google traces. This is caused by the higher dynamicity and variety of workloads,more complex interactions among system components, less number of performance metrics and incomplete information of failure types. Our anomaly detector still provides valuable information of failure dynamics, which facilitates the system operators to proactively reconfigure resources and schedule workloads.

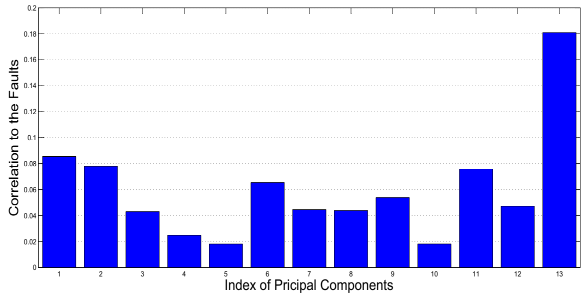


Fig. 10. Correlation between principal components and failure events using the Google datacenter traces.

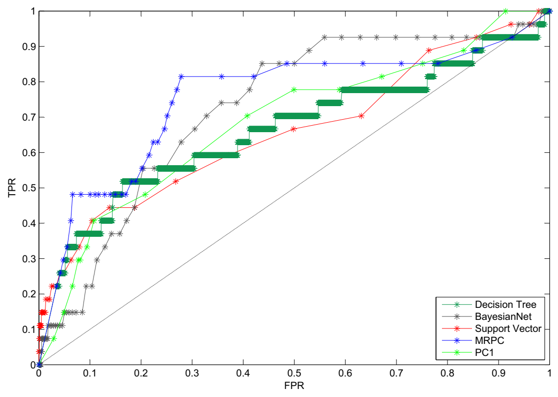


Fig. 11. Performance of the proposed MRPC-based anomaly detector compared with four other detection algorithms on the Google datacenter traces.

V. Related Work

Anomaly detection is an important topic and attracts considerable attention [40]. Typical methods treat anomalies as deviation of the system performance [27], [10]. Examples include diagnosis and prediction based on probabilistic or analytical model [21], real-time streams of computer events[24], [42] and continuous monitoring over the runtime services [43], [23]. These model-based system diagnosing methods analyze the system by deriving a probabilistic or analytical model. Models are trained with the help of prior knowledges. Examples include native Bayesian-based model for hardware disk failure [21], EM-algorithm model for the highest-likelihood mixture [30] and Hidden Markov model for online failure detection [36]. These approaches are all based on the study of the huge amount of the health-related data while being trained, which specially addresses the complication in mining daily log in the magnitude of GBs. Furthermore, the insufficient failure data makes it difficult to accurately diagnose the root causes.

In order to overcome these drawbacks, other researchers seek to involve the performance metrics selection and metrics extraction [29], [33] as a pre-process to maintain the variance of dataset as much as possible while shrinking the size to improve the speed for analysis. Metrics selection and extraction expect to remove the redundancy in the dataset. But still they are sharing the weakness originating from the characteristic of a model-based manners while maintaining the accuracy of the model is difficult for a complex system, especially while system is moved to cloud environment and featured with virtualization. Meanwhile, any maintenance, update and upgrades on the cloud system post the requirement of rebuilding the model. Moreover, any changes to virtual architecture(i.e., virtual machine migration, initialization and shutdown) make the model ineffective and inadequate. In addition, for many cloud infrastructures, the behavior of workload is never taken into account, though it may be a dominant factor to affect the results of metric selection and extraction. Besides, this kind of methods also suffer from the overfits, which are brought by specific training data sets.

Much of the work has referred to statistical technologies for specific type of failures by profiling the failures happened in the system and pursuing the knowledge that reveals on the failure pattern. Some work was done by investigating the failures happened in Hard disk, DRAM and network [37], [26], [41]. These results provide great help to the checkpointing and restoring mechanism for the proactive failure management, by analyzing the knowledge of failure distribution. However, the failure information is rare and needs the consistence of the system setting. But usually it is hard to be guaranteed due to the possible system update and reconfiguration. Moreover, without timely failure validation, this kind of unsupervised model bears a very high false alarm rate.

This work is supported by our previous work on failure management [16], [17], [31], [8], [19], [18], [20], [32], [11], [14], [15], [13]. The main contribution of our work includes it is the first time to use subsets of principal components as the most relevant metrics for different types of failures. With the analysis of each failure type, we show that anomalies are highly correlated with specific principal component subsets. Experimental results show the proposed method can achieve high accuracy of anomaly detection and low overhead.

VI. Conclusion

Modern large-scale and complex cloud computing systems are susceptible to software and hardware failures, which significantly affect the cloud dependability and performance. In this paper, we present an adaptive anomaly identification mechanism in cloud computing systems. We start by analyzing the correlation between principal components with failure occurrences, where we find the PCs retaining the highest variance cannot effectively characterize the failure events, while lower order PCs displaying high correlation with occurrences of failures. We then propose to exploit the most relevant principal components (MRPCs) to describe failure events and devise a learning based approach to identify cloud anomalies by leveraging MRPCs. The anomaly detector adapts itself by recursively learning from these newly verified detection results to refine future detections. Meanwhile, it exploits the observed but undetected failure records reported by the cloud operators to identify new types of failures. Experimental results from an on-campus cloud computing testbed show that the proposed MRPC-based anomaly identification mechanism can accurately detect failures while achieving a low overhead. The proposed MRPC-based anomaly identification mechanism in this research can also aid failure prediction. Complementing existing failure prediction methods, results from this research can be utilized to determine the potential localization of failures by analyzing the runtime cloud performance data. In this paper, we employ neural networks for adaptive anomaly identification. We plan to explore other advanced statistical learning methods for anomaly detection and automated adaptation for dependability assurance in cloud computing environments. We note that even with the most advanced learning methods, the accuracy of anomaly detection cannot reach 100%. Reactive approaches, such as checkpointing and redundant execution, should be included to handle misdetections. We plan to integrate the proactive and reactive failure management approaches to achieve even higher cloud dependability.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments and suggestions. This research was supported in part by U.S. NSF grant CNS-0915396 and LANL grant IAS-1103.

References

[1] *http://hadoop.apache.org/.*

[2] *http://sebastien.godard.pagesperso-orange.fr.*

[3] Linux profiling with performance counters. Available: http://perf.wiki.kernel.org/.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[5] D. Bernick, B. Bruckert, P. D. Vigna, D. Garcia, R. Jardine, J. Klecka, and J. Smullen. Nonstop advanced architecture. In *Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2005.

[6] D. Brauckhoff, K. Salamatian, and M. May. Applying pca for traffic anomaly detection: Problems and solutions. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58, 2009.

[8] N. DeBardeleben, S. Blanchard, Q. Guan, Z. Zhang, and S. Fu. Experimental framework for injecting logic errors in a virtual machine to profile applications for soft error resilience. In *Proceedings of Resilience, International European Conference on Parallel and Distributed Computing (Euro-Par)*, 2011.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

[10] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan. Availability in globally distributed storage systems. In *Proceedings of USENIX Conference on Operating Systems Design and Implementation (OSDI)*, 2010.

[11] S. Fu. Failure-aware resource management for high-availability computing clusters with distributed virtual machines. *Journal of Parallel and Distributed Computing*, 70(4):384–393, 2010.

[12] S. Fu. Performance metric selection for autonomic anomaly detection on cloud computing systems. In *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, 2011.

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

[14] S. Fu and C. Xu. Quantifying event correlations for proactive failure management in networked computing systems. *Journal of Parallel and Distributed Computing*, 70(11):1100–1109, 2010.

[15] S. Fu and C.-Z. Xu. Exploring event correlation for failure prediction in coalitions of clusters. In *Proceedings of ACM/IEEE Supercomputing Conference (SC)*, 2007.

[16] G. Guan, C.-C. Chiu, and S. Fu. CDA: A cloud dependability analysis framework for characterizing system dependability in cloud computing infrastructures. In *Proceedings of IEEE International Symposium on Dependable Computing (PRDC)*, 2012.

[17] Q. Guan, C.-C. Chiu, Z. Zhang, and S. Fu. Efficient and accurate anomaly identification using reduced metric space in utility clouds. In *Proceedings of IEEE International Conference on Networking, Architecture, and Storage (NAS)*, 2012.

[18] Q. Guan and S. Fu. auto-AID: A data mining framework for autonomic anomaly identification in networked computer systems. In *Proceedings of IEEE International Performance Computing and Communications Conference (IPCCC)*, 2010.

[19] Q. Guan, Z. Zhang, and S. Fu. Proactive failure management by integrated unsupervised and semi-supervised learning for dependable cloud systems. In *Proceedings of IEEE International Conference on Availability, Reliability and Security (ARES)*, 2011.

[20] Q. Guan, Z. Zhang, and S. Fu. Ensemble of bayesian predictors and decision trees for proactive failure management in cloud computing systems. *Journal of Communications*, 7(1):52–61, 2012.

[21] G. Hamerly and C. Elkan. Bayesian approaches to failure prediction for disk drives. In *Proceedings of International Conference on Machine Learning (ICML)*, 2001.

[22] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 1960.

[23] K. Kc and X. Gu. ELT: Efficient log-based troubleshooting system for cloud computing infrastructures. In *Proceedings of IEEE International Symposium on Reliable Distributed Systems (SRDS)*, 2011.

[24] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2004.

[25] Z. Lan, Z. Zheng, and Y. Li. Toward automated anomaly identification in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 21(2):174–187, 2010.

[26] J. B. Leners, H. Wu, W.-L. Hung, M. K. Aguilera, and M. Walfish. Detecting failures in distributed systems with the falcon spy network. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, 2011.

[27] Z. Li, M. Zhang, Z. Zhu, Y. Chen, A. Greenberg, and Y.-M. Wang. WebProphet: automating performance prediction for web services. In *Proceedings of USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2010.

[28] F. Longo, R. Ghosh, V. K. Naik, and K. S. Trivedi. A scalable availability model for infrastructure-as-a-service cloud. In *Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2011.

[29] J. McBain and M. Timusk. Feature extraction for novelty detection as applied to fault detection in machinery. *Pattern Recognition Letter*, 32(7):1054–1061.

[30] J. F. Murray, G. F. Hughes, and D. Schuurmans. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning research*, 2005.

[31] H. Pannu, J. Liu, and S. Fu. AAD: Adaptive anomaly detection system for cloud computing infrastructures. In *Proceedings of IEEE International Symposium on Reliable Distributed Systems (SRDS)*, 2012.

[32] H. Pannu, J. Liu, and S. Fu. Autonomic anomaly identification for developing highly dependable utility clouds. In *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, 2012.

[33] G. Qu, S. Hariri, and M. Yousif. A new dependency and correlation analysis for features. *IEEE Transactions on Knowledge and Data Engineering*, 17(9):1199–1207, 2005.

[34] C. Reiss, J. Wilkes, and J. L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., Nov. 2011.

[35] F. Salfner, M. Lenk, and M. Malek. A survey of online failure prediction methods. *ACM Computing Surveys*, 42(3):10:1–10:42, 2010.

[36] F. Salfner and M. Malek. Using hidden semi-markov models for effective online failure prediction. In *Proceedings of IEEE International Symposium on Reliable Distributed Systems (SRDS)*, 2007.

[37] B. Schroeder, E. Pinheiro, and W.-D. Weber. DRAM errors in the wild: a large-scale field study. In *Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2009.

[38] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. Principal component-based anomaly detection scheme. *Foundations and Novel Approaches in Data Mining, Springer-Verlag*, 21(2):311–329, 2006.

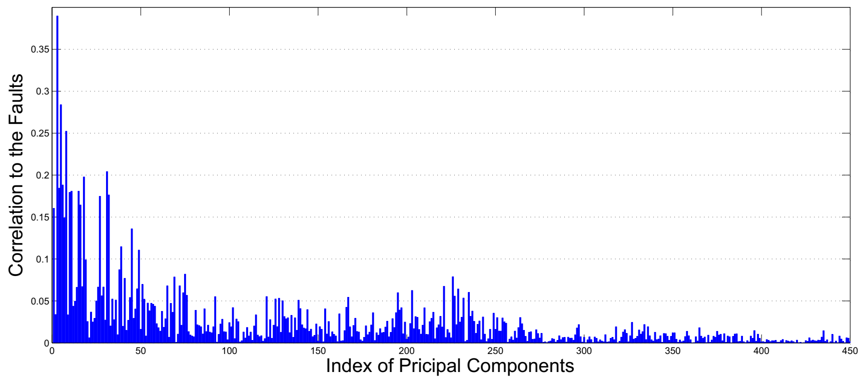
[39] J. E. Smith and R. Nair. The architecture of virtual machines. *IEEE Computer*, 38(5):32–38, 2005.

[40] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan. PREPARE: Predictive performance anomaly prevention for virtualized cloud systems. In *Proceedings of IEEE International Conference onDistributed Computing Systems (ICDCS)*, 2012.

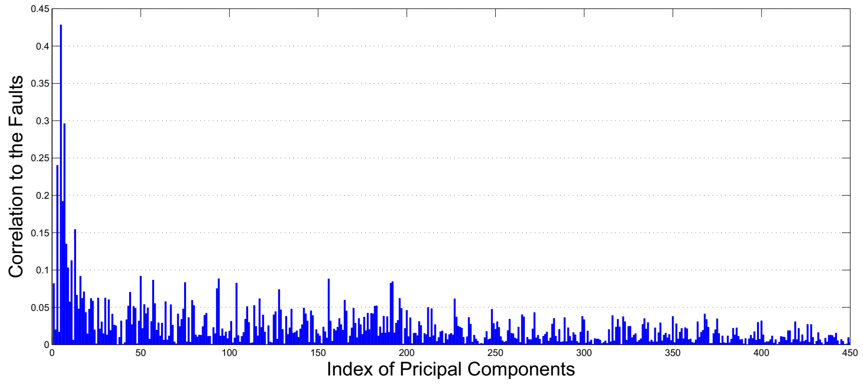
[41] K. V. Vishwanath and N. Nagappan. Characterizing cloud computing hardware reliability. In *Proceedings of ACM Symposium on Cloud Computing (SoCC)*, 2010.

[42] L. Yang, C. Liu, J. M. Schopf, and I. Foster. Anomaly detection and diagnosis in grid environments. In *Proceedings of ACM/IEEE Supercomputing Conference (SC)*, 2007.

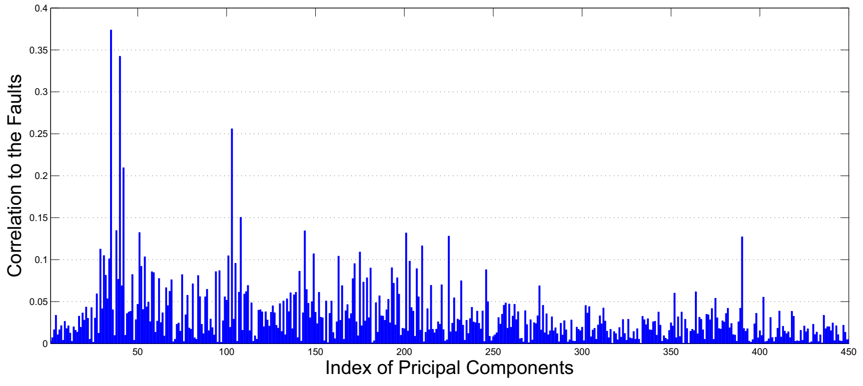
[43] Z. Zheng, L. Yu, W. Tang, Z. Lan, R. Gupta, N. Desai, S. Coghlan, and D. Buettner. Co-analysis of RAS log and job log on BlueGene/P. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2011.



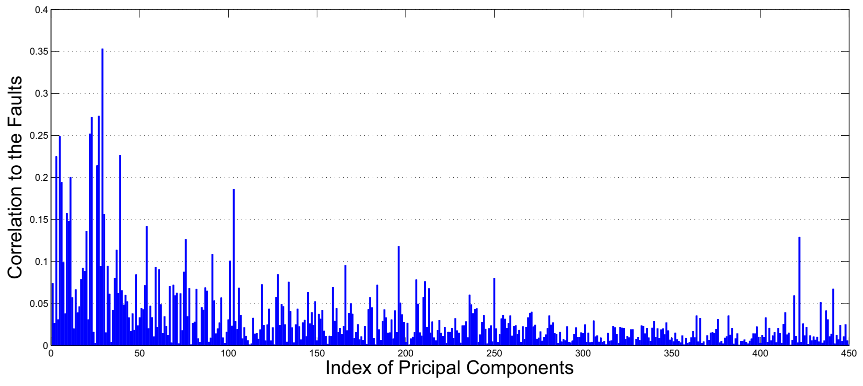
(a) Correlation with memory-related failures.



(b) Correlation with disk-related failures.



(c) Correlation with CPU-related failures.



(d) Correlation with network-related failures.

Fig. 8. Correlation between the principal components and different types of failures.