A Self-Evolving Anomaly Detection Framework for Developing Highly Dependable Utility Clouds

Husanbir S. Pannu and Jianguo Liu  
Department of Mathematics  
University of North Texas  
Email: HusanbirPannu@my.unt.edu, Jianguo.Liu@unt.edu

Song Fu  
Department of Computer Science and Engineering  
University of North Texas  
Email: Song.Fu@unt.edu

Abstract—Utility clouds continue to grow in scale and in the complexity of their components and interactions, which introduces a key challenge to failure and resource management for highly dependable cloud computing. Autonomic anomaly detection is a crucial technique for understanding emergent, cloud-wide phenomena and self-managing cloud resources for system-level dependability assurance. To identify anomalies, we need to monitor the system execution and collect health-related runtime performance data. These data are usually unlabeled and a prior failure history is not always available in production systems, especially for newly deployed or managed utility clouds. In this paper, we present a self-evolving anomaly detection framework with mechanisms for dependability assurance in utility clouds. No prior failure history is required. The detector self-evolves by recursively exploring newly generated verified detection results for future anomaly identification. Statistical learning technologies are exploited in detector determination and working dataset selection. Experimental results in an institute-wide cloud computing system show that the detection accuracy improves as it evolves. With self-evolvement, the detector can achieve 92.1% detection sensitivity and 83.8% detection specificity, which makes it well suitable for building highly dependable utility clouds.

Keywords: Cloud computing; Anomaly identification; Self evolvement; Dependable systems; Autonomic management.

I. INTRODUCTION

Modern production utility clouds contain hundreds to thousands of computing and storage servers. Such a scale, combined with the ever-growing system complexity, is introducing a key challenge to failure and resource management for highly dependable cloud computing. Despite great efforts on the design of ultra-reliable components [3], [9], the increase of cloud size and complexity has outpaced the improvement of component reliability. Results from recent studies [44], [27] show that the reliability of existing datacenters and utility clouds is constrained by a mean time between failure (MTBF) on the order of 10 - 100 hours. This situation is only likely to deteriorate in the near future, thereby threatening the promising productivity of utility clouds. Failure occurrence as well as its impact on system performance and operating costs is becoming an increasingly important concern to cloud designers and operators [1], [44]. The success of cloud computing will depend on the ability to provide dependable at scale.

Considering the high complexity and dynamicity of utility clouds, autonomic failure management is an effective approach to enhancing cloud dependability [1]. Anomaly detection is a key technique [5]. It identifies anomalous behaviors and possibly forecasts failure occurrences in a cloud by exploring the cloud’s runtime execution states and failure records. It provides valuable information for resource allocation, virtual machine reconfiguration and cloud maintenance [35].

Efficient and accurate anomaly detection in utility clouds 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. Recent work has developed various technologies to tackle the problems of anomaly detection and failure management. They, however, require failure history and fall short, in one area or the other, in terms of efficiency, accuracy, adaptivity, and online dependability.

Most studies on anomaly detection identify anomalous behavior by examining historical data, which is based on a vertical view of the system. They are often characterized by the requirement of training a classifier on samples of both normal and failure data. The fundamental problem with this approach is that it limits us to the detection and localization of failures with known signatures. To identify anomalies, we need to monitor the system execution and collect health-related runtime performance data. These data are usually unlabeled and a prior failure history is not always available in production systems, especially for newly deployed or managed utility clouds.

Distinguished from the existing approaches, in this paper we exploit an integrated horizontal and vertical view of utility clouds for anomaly detection. Our self-evolving anomaly detection (SEAD) system does not require training on previously labeled events, thereby being capable of finding anomalies not yet seen in the past, and it can self-evolve by learning from newly generated verified detection results. Statistical learning technologies are exploited in detector determination and working dataset selection. We implement a prototype of SEAD and evaluate its performance on an institute-wide cloud computing system. Experimental results show that SEAD can detect failures more accurately as it evolves by learning more verified detection results. With self-evolvement, the SEAD can achieve 92.1% detection sensitivity and 83.8% detection specificity, which shows SEAD can be exploited to enhance system dependability in utility clouds.
The rest of this paper is organized as follows. Section II describes a framework for monitoring cloud health and assuring system dependability, in which SEAD is a major component. Section III presents the key mechanisms of SEAD. Experimental results are included and discussed in Section IV. Section V presents the related research on anomaly detection and failure management. Conclusion and remarks on future work are presented in Section VI.

II. A FRAMEWORK FOR CLOUD HEALTH MONITORING AND DEPENDABILITY ASSURANCE

To build highly dependable utility clouds, we propose a reconfigurable distributed virtual machine (RDVM) infrastructure, which leverages the virtualization technologies to facilitate failure-aware cloud resource management. SEAD is a key component in this infrastructure. A RDVM consists of a set of virtual machines running on top of physical servers in a utility cloud. Each VM encapsulates execution states of cloud services and running client applications. It is the basic unit of management for RDVM construction and reconfiguration. Each cloud server hosts multiple virtual machines. These virtual machines multiplex resources of the underlying physical server. The virtual machine monitor (VMM, also called hypervisor) is a thin layer that manages hardware resources and exports a uniform interface to the upper VMs [33].

When a client application is submitted with its computation and storage requirement to the cloud, the Cloud Coordinator evaluates qualifications of available cloud servers. It selects one or a set of them for the application, initiates the creation of VMs on them, and then dispatches the application instances for execution. Virtual machines on a cloud server are managed locally by a RDVM daemon, which is also responsible for communication with Resource Manager, SEAD Anomaly Detector and Cloud Coordinator. The RDVM daemon monitors the health status of the corresponding cloud server, collects runtime performance data of the local VMs, and sends them to the SEAD Anomaly Detector, which characterizes cloud behaviors, identifies anomalous states, and reports the detected anomalies to cloud operators for verification. The verified detections will be input back to the SEAD Anomaly Detector for self-evolution. Based on the performance data and failure reports, the Resource Manager analyzes the workload distribution, online availability, and allocated and available cloud resources, and then makes RDVM reconfiguration decisions. The SEAD Anomaly Detector and Resource Manager form a closed feedback control loop to deal with dynamics and uncertainty of the utility cloud.

To identify anomalies, SEAD needs the runtime health-related performance data. The performance data collected periodically by the RDVM daemons include the application execution status and the runtime utilization information of various virtualized resources on client virtual machines. RDVM daemons also work with hypervisors to record the performance of hypervisors and monitor the utilization of underlying hardware resources/devices. These data and information from multiple system levels (i.e., hardware, hypervisor, virtual machine, RDVM, and the cloud) are valuable for accurate assessment of the cloud’s health and for detecting anomalies and pinpointing failures. They constitute the health-related cloud performance dataset, which is explored by SEAD for anomaly identification.

III. SELF-EVOLVING ANOMALY DETECTION METHOD

At a high level, the proposed self-evolving anomaly detection (SEAD) system works as follows. When the prior failure history is not available, the SEAD detector identifies anomalies by searching for those cloud health states that are significantly different from the majority. As the anomalies get verified by cloud operators, they are confirmed as either failures or normal states. Our SEAD anomaly detector self-evolves by recursively learning from the newly generated verified detection results to refine future detections in a closed-loop control manner.

SEAD includes two components. One is anomaly detector determination. The detector is self-evolving and constantly learning. For a new cloud performance data record, the SEAD detector calculates an abnormality score. If the score is above a threshold, a warning is triggered with the type of abnormality, which will help cloud operators to pinpoint the anomaly. The other component is anomaly detector evolving and working dataset selection. The detector evolves when certain newly verified and labeled data records are included in the working dataset.

A. Cloud Performance Metric Selection

Continuous monitoring and large cloud scale lead to an overwhelming volume of the collected health-related performance data, which can easily reach hundreds and even thousands of gigabytes in production systems [28], [37]. In addition to the data size, the large number of performance metrics that are measured makes the data model extremely complex. The existence of interacting metrics and external environmental factors introduce measurement noises to the collected data. High metric dimension will cause low detection accuracy and high computational complexity.

To make anomaly detection tractable and yield high accuracy, we have developed an integrated performance metric selection and extraction framework, which has been presented in [10]. The mechanisms in the framework transform the collected health-related cloud performance data to a new metric space with only the most essential metrics included. We have developed two methods: metric selection based on mutual information and metric extraction by metric combination and separation. The goal is to present the cloud performance data in a low-dimensional subspace with most of the principal properties preserved. SEAD exploits those mechanisms for efficient and accurate anomaly identification. Because of the space limit, we do not present the details of the metric selection mechanisms in this paper.

B. SEAD Anomaly Detector Determination

Without loss of generality, we assume the given utility cloud is newly deployed or managed, i.e., no history information of
failure occurrences is available. Health-related cloud performance data are continuously collected at runtime. Initially, all the data records are treated as normal ones. As time goes by, a small percentage of anomalous records is detected. Each of those anomalies is then verified and labeled as either a normal event or a failure with the corresponding failure type.

Numerous anomaly detection algorithms have been proposed and investigated. Chandola et al. [5] categorized them into six techniques: classification based, clustering based, statistical, nearest neighbor based, information theoretic, and spectral. Nearest neighbor based, information theoretic, and spectral methods do not have a training phase. However, they can be slow for testing. Classification based, clustering based, and statistical methods are computationally expensive for training. However, they are very fast for testing. In addition, training can be performed offline. Anomaly detection requires fast and accurate online calculation. Therefore, classification based and clustering based methods are the choices for our design. For now, statistical methods are not included, because their performance heavily relies on certain data distribution assumptions.

There are several approaches for classification and clustering. Our preliminary experimental results show that support vector machines (SVM) and one-class support vector machines (one-class SVM, support vector clustering (SVC), and support vector data description (SVDD)), work quite well. References for SVM and one-class SVM include [42], [36], [2], [40]. We consider these two methods and propose how to adapt them for developing self-evolving anomaly detector in utility clouds. An ensemble incorporated with other methods can be designed using the same framework.

At an early stage of the deployment of the SEAD anomaly detector, the working dataset includes only normal data. The detector is a function generated by one-class classification. To be more specific, let $D = \{ x_i \mid i = 1, 2, \ldots, L \}$ be the working cloud performance dataset including $L$ records $x_i \in \mathbb{R}^d$, $(i = 1, 2, \ldots, L)$. Let $\phi$ be a mapping from $\mathbb{R}^d$ to a cloud performance metric space, where dot products are evaluated by some kernel functions: $k(x, y) = \langle \phi(x), \phi(y) \rangle$. A common kernel function is the Gaussian kernel $k(x, y) = \exp(-\|x-y\|^2/\sigma^2)$. The SEAD anomaly detector separates the working dataset from the origin by solving a minimization problem:

$$\min_{w, b, \xi} \left\{ \frac{1}{2} \|w\|^2 - b + \frac{1}{2} \sum_i \xi_i \right\}$$

subject to $\langle w, \phi(x_i) \rangle \geq b - \xi_i$, $(i = 1, 2, \ldots, L)$, \hspace{1cm} (III.1)

where $w$ is a vector perpendicular to the hyperplane in the metric space, $b$ is the distance from the hyperplane to the origin, and $\xi$ are soft-margin slack variables to handle anomalies. The parameter $\nu \in (0, 1)$ controls the trade-off between the number of records in the dataset mapped as positive by the detection function $f(x) = \text{sgn}(\langle w, \phi(x) \rangle - b)$ and having a small value of $||w||$ to control the detector’s complexity. In practice, the dual form of Equation (III.1) is often solved. Let $\alpha_i$, $(i = 1, 2, \ldots, L)$ be the dual variables. Then the detection function is $f(x) = \text{sgn}(\sum_i \alpha_i k(x, x_i) - b)$. A new cloud performance data record $x$ is identified to be normal if $f(x) = 1$ or an anomaly if $f(x) = -1$. One of the advantages of the dual form is that the detection function can be evaluated by using a simple kernel function instead of the expensive inner product in the metric space.

As the working cloud performance dataset grows, it gradually contains verified detection records generated by the cloud operators. In other words, two classes or multiple classes (for multiple failure types) of data records are available. The detection mechanism of the SEAD anomaly detector is updated and evolved, which can be formulated as

$$\min_{w, b, \xi} \left\{ \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \right\}$$

subject to $y_i(\langle w, \phi(x_i) \rangle + b) \geq 1 - \xi_i$, and $\xi_i \geq 0$ $(i = 1, \ldots, L)$, \hspace{1cm} (III.2)

where $C > 0$ is a parameter to deal with mis-detection and $y_i \in \{+1, -1\}$ are given class labels. A data record $x_i$ is normal if the corresponding class label $y_i = 1$ or a failure if $y_i = -1$. Once again, a dual form is solved and the detection function is $f(x) = \text{sgn}(\sum_i \alpha_i k(x, x_i) + b)$. A new cloud performance data record $x$ could be identified to be normal if $f(x) = 1$ or anomalous if $f(x) = -1$. Multi-class detection can be implemented by using several binary detectors.

A challenge to anomaly detection is that the working dataset is often highly unbalanced: normal data records outnumber failure records by a big margin. The detector’s accuracy is often degraded when working on unbalanced datasets. However, as the percentage of failure records increases, the performance of the SEAD anomaly detector will improve. Our experiments show that the detector starts to perform reasonably well for this particular unbalanced problem once the percentage reaches 5%. Our detector is determined by combining Equations (III.1) and (III.2) with a sliding scale weighting strategy.

The weighting is based on two factors. One is the credibility score and the other is the percentage of failure records in the working dataset. The method with a higher credibility score weighs more and more weight is given to the detection mechanism as the percentage of failure records increases. For a given method, let $a(t)$ denote the number of attempted detections and $c(t)$ denote the number of correct detections, where $t$ is any given time. The credibility score is defined as

$$s(t) = \begin{cases} \frac{c(t)}{a(t)} & \text{if } a(t) > 0 \text{ and } \frac{c(t)}{a(t)} > \nu \\ 0 & \text{if } a(t) = 0 \text{ or } \frac{c(t)}{a(t)} \leq \nu, \end{cases} \hspace{1cm} (III.3)$$

where $\nu \in (0, 1)$ is a parameter of zero trust. A good choice is $\nu = 0.5$. Let $s_1(t)$ and $s_2(t)$ be the credibility scores of the detection mechanisms (III.1) and (III.2), respectively. Let $p(t)$ denote the percentage of failure records in the working dataset.

Suppose $f_1(x)$ is the detection function generated by detection mechanism (III.1) and $f_2(x)$ is generated by mechanism (III.2), where $x$ is a new cloud performance data record at time $t$. Then the combined detection function is given by

$$f(x) = \begin{cases} f_1(x) s_1(t) & \text{if } p(t) = 0 \\ f_1(x) s_1(t) (1 - \frac{p(t)}{\eta}) + f_2(x) s_2(t) \frac{p(t)}{\theta} & \text{if } 0 < p(t) < \theta \\ \frac{1}{2} (f_1(x) s_1(t) + f_2(x) s_2(t)) & \text{if } p(t) \geq \theta, \end{cases} \hspace{1cm} (III.4)$$
where \( \theta \in (0, 1) \) is a parameter of trust on the mechanism (III.2) related to the percentage of failure records. A reasonable choice is \( \theta = 0.1 \). An anomaly warning is triggered if \( f(x) \) is smaller than a threshold \( \tau \), for example \( \tau = 0 \). When multiple labels (for different failure types) are available for failure records, a multi-class SEAD anomaly detector can be developed to detect the type of an anomaly, if a new data record is identified to be anomalous.

**C. SEAD Anomaly Detector Evolvement and Working Dataset Selection**

Detector evolvement and working dataset selection are important parts of the SEAD’s learning process. The idea is to learn and improve from mistakes and maintain a reasonable size of the dataset for efficient evolvement. Initially, all data records are included in the working dataset to build up a good base to train the detector. Once the dataset reaches a certain size and the detection accuracy is stabilized, the inclusion becomes selective. A new cloud performance data record \( x \) is included in the working dataset only if one or more of the following is true:

- The data record corresponds to an anomaly and \( p(t) < 0.5 \). It is ideal to include more failure records in the working dataset but not too many.
- One of the detections by \( f_1(x) \), \( f_2(x) \), or \( f(x) \) is incorrect. The detector will be evolved to learn from the mistake.
- The data record may change the boundary points for classification. This happens when the absolute value of \( \sum_i a_i k(x_i, x) + b \) is less than 1, where we assume \( f_2(x) = \text{sgn}(\sum_i a_i k(x_i, x) + b) \). The detector will be evolved to achieve better detection accuracy.

The decision functions \( f_1(x) \) and \( f_2(x) \) are evolved whenever a new data record enters the working dataset. The evolvement can be done quickly because the size of the dataset is well maintained. In addition, the solutions of the old-version anomaly detection problem are used as the initial identification for the solutions of the new problems. Solving the anomaly detection problem is an iterative process. Having good initial identification will make the iterations converge fast to the new solutions.

**IV. Performance Evaluation**

We have implemented a proof-of-concept prototype of SEAD and tested it on a cloud computing system on campus. The cloud consists of 362 servers, which are connected by gigabit Ethernet. The cloud servers are equipped with two to four Intel Xeon or AMD Opteron cores and 2.5 to 8 GB of RAM. We have installed Xen 3.1.2 hypervisors on the cloud servers. The operating system on a virtual machine is Linux 2.6.18 as distributed with Xen 3.1.2. Each cloud server hosts up to eight VMs. A VM is assigned up to two VCPUs, among which the number of active ones depends on applications. The amount of memory allocated to a VM is set to 512 MB.

We run the RUBiS [4] distributed online service benchmark and MapReduce [8] jobs as cloud applications on VMs. The applications are submitted to the cloud computing system through a web based interface. We have also developed a fault injection program, 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.

We exploit the third-party monitoring tools, sysstat [39] to collect runtime performance data in Dom0 and a modified perf [31] to obtain the values of performance counters from the Xen hypervisor on each cloud server. In total, 653 metrics are profiled every minute. They cover the statistics of every component of a cloud server, including CPU usage, process creation, task switching activity, memory and swap space utilization, paging and page faults, interrupts, network activity, I/O and data transfer, power management, and more. We tested the system from June 25, 2011 to January 16, 2012. In total, about 601.4 GB health-related performance data were collected and recorded from the cloud in that period of time. Among all the metrics, 186 of them display zero variance, which provides no contribution to anomaly detection. After removing them, we have 464 non-constant metrics left. By applying the metric selection mechanisms presented in [10], SEAD selects 14 metrics, whose normalized (redundancy-relevance) values are less than a threshold (i.e., \( \alpha = 0.15 \)) and then combines them to three major metrics for anomaly detection.
We evaluate the performance of SEAD’s anomaly detection function. We measure the detection sensitivity and specificity. The detection sensitivity is calculated by the number of detected failures / the number of true failures. The detection specificity equals to the number of detected normal states / the number of true normal states. The SEAD detector identifies anomalies in the collected health-related cloud performance data. It self-evolves by learning the verified detection results from the cloud operators. Figure 1 depicts the detection performance after three selected rounds of self-evolvement. 78, 66, and 43 verified detection results are learned by the detection during the Self-evolvement I, II, and III, respectively. From the figure, we can see both the detection sensitivity and the detection specificity improve as more verified detection results are exploited to evolve the detector. After Self-evolvement III is applied, the SEAD anomaly detector identifies possible failures in the new health-related performance data. Figure 2 shows the detection results. Only 20% of the detected normal data points are included in the figure for better readability. In total, 35 anomalies are identified. The detector achieves 92.1% detection sensitivity and 83.8% detection specificity. It indicates that SEAD is well suitable for building highly dependable utility clouds.

V. Related Research

The online detection of anomalous system behavior [5] caused by operator errors [29], hardware/software failures [32], resource over-/under-provisioning [23], [22], and similar causes is a vital element of operations in large-scale data centers and utility clouds. Given the ever-increasing cloud scale coupled with the increasing complexity of software, applications, and workload patterns, anomaly detection methods must operate automatically at runtime and without the need for prior knowledge about normal or anomalous behaviors.

Failure and anomaly detection based on analysis of performance logs has been the topic of numerous studies. Hodge and Austin [21] provided an extensive survey of anomaly detection techniques developed in machine learning and statistical domains. A structured and broad overview of extensive research on anomaly detection techniques was presented in [5]. Broadly speaking, existing approaches can be classified into two categories: model-based and data-driven.

A model-based approach derives a probabilistic or analytical model of a system. A warning is triggered when a deviation from the model is detected [20]. Examples include an adaptive statistical data fitting method called MSET presented in [41], naive Bayesian based models for disk failure prediction [19], and Semi-Markov reward models described in [14]. In large-scale systems, errors may propagate from one component to others, thereby making it difficult to identify the causes of failures. A common solution is to develop fault propagation models, such as causality graphs or dependency graphs [38]. Generating dependency graphs, however, requires a priori knowledge of the system structure and the dependencies among different components, which is hard to obtain in large-scale systems. The major limitation of model-based methods is their difficulty of generating and maintaining an accurate model, especially given the unprecedented size and complexity of production cloud computing systems.

Recently, data mining and statistical learning theories have received growing attention for anomaly detection and failure management. These methods extract failure patterns from systems’ normal behaviors, and detect abnormal observations based on the learned knowledge [30]. For example, the RAD laboratory at UC-Berkeley has applied statistical learning techniques for failure diagnosis in Internet services [6], [46]. The SLIC (Statistical Learning, Inference and Control) project at HP has explored similar techniques for automating failure management of IT systems [7]. In [34], [43], the authors have presented several methods to forecast failure events in IBM clusters. In [26], [25], Liang et al. have examined several statistical methods for failure prediction in IBM Blue Gene/L systems. In [24], Lan et al. have investigated meta-learning based method for improving failure prediction. In [15], [16], [13], [11], [17], [45], [18], Guan and Fu have developed proactive failure management mechanisms and frameworks in networked computing systems.

VI. Conclusion

Large-scale and complex utility clouds are susceptible to software and hardware failures and administrators’ mistakes, which significantly affect the cloud performance and management. In this paper, we present SEAD, a self-evolving anomaly detection framework with mechanisms. It does not require a prior failure history, thereby being capable of finding anomalies not yet seen in the past. It can self-evolve by learning from newly generated verified detection results. The proposed framework with mechanisms in this research can also aid failure prediction. Existing predictive methods such as [25], [34], [11], [24], [12] are used on the performance and failure logs to predict when the underlying system will experience a critical event. Results from this research can be utilized to determine the potential localizations of the problem by analyzing the runtime cloud performance data.

In this work, we adapt one-class SVM and SVM for self-evolving anomaly detection. We plan to explore other advanced statistical learning techniques for anomaly identification and self evolvement in utility clouds. We note that even with the most advanced learning techniques, the accuracy of anomaly detection cannot reach 100%. Reactive approaches, such as checkpointing and redundant execution, should be included to handle mis-predictions. 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


