

On the Effectiveness of Isolation-Based Anomaly Detection in Cloud Data Centers

Rodrigo N. Calheiros^{1*†}, Kotagiri Ramamohanarao², Rajkumar Buyya²,
Christopher Leckie², Steve Versteeg³

¹*School of Computing, Engineering and Mathematics, Western Sydney University, Australia*

²*School of Computing and Information Systems, The University of Melbourne, Australia*

³*CA Technologies, Melbourne, Australia*

SUMMARY

The high volume of monitoring information generated by large-scale cloud infrastructures poses a challenge to the capacity of cloud providers in detecting anomalies in the infrastructure. Traditional anomaly detection methods are resource-intensive and computationally complex for training and/or detection, what is undesirable in very dynamic and large-scale environment such as clouds. Isolation-based methods have the advantage of low complexity for training and detection and are optimized for detecting failures. In this work, we explore the feasibility of Isolation Forest, an isolation-based anomaly detection method, to detect anomalies in large-scale cloud data centers. We propose a method to code time-series information as extra attributes that enable temporal anomaly detection and establish its feasibility to adapt to seasonality and trends in the time-series and to be applied on-line and in real-time. Copyright © 2017 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Cloud Computing; Data Centers; Anomaly Detection; Time-Series

1. INTRODUCTION

The rapid adoption of cloud computing [1] by governments and businesses is astonishing. This is because cloud computing offers the possibility for its customers to elastically meet their application's demands, paying only for resources that are used. Such business model contrasts with conventional operation of in-house infrastructures that need to be provisioned for the peak capacity, generating huge capital costs (e.g., acquisition of computing servers and network equipment) and operational costs (e.g., electricity costs incurred by the facility and its cooling system and salary of system administrators).

However, there are a few concerns that inhibit the adoption of clouds by governments and companies. One of those concerns is performance and availability of applications once they are migrated to the cloud. Even short periods of outage or substantial performance degradation in a critical time (e.g., high demand for on-line shopping during Christmas period) can lead to substantial losses to companies hosting applications on clouds.

One possible way that cloud providers can reassure their customers is through strict Service-Level Agreements (SLAs) that determine the Quality of Service (QoS) and compensations to be paid to customers if QoS is not met. Besides metrics such as availability, the SLA can also include

*Correspondence to: Western Sydney University, Locked Bag 1797 Penrith NSW 2751 Australia

†E-mail: r.calheiros@westernsydney.edu.au

guarantees that Virtual Machine (VM) entitlements (the amount of resources from physical hosts allocated to VMs) are respected.

To meet the availability requirements and minimize performance degradation in virtual machines, cloud providers apply monitoring in the infrastructure. However, monitoring and management of cloud infrastructures is a complex task. Large-scale cloud data centers operated by cloud providers usually contain thousands or tens of thousands of servers that can host hundreds of thousands of virtual machines. Failures on an infrastructure of such scale are very common, thus mechanisms must be in place to identify them as quickly as possible to prevent, or at least minimize, downtime of virtual machines (and thus applications) hosted in the infrastructure. Moreover, virtual machines can host a multitude of applications, and thus separating monitoring measurements that indicate failures from noisy application behavior is challenging.

Anomaly detection techniques [2] can be applied to monitor behavior such as variation of patterns in user demand, applications, and infrastructure in cloud data centers [3,4]. Nevertheless, traditional anomaly detection methods have two main limitations to be applied in the discussed context [5]: (i) they are computationally intensive for training and/or detection; and (ii) they are optimized to detect normal behavior, and the identification of the anomaly is a “by product” of the classification process.

To circumvent the aforementioned limitations of anomaly detection methods, Liu et al. [5, 6] proposed the first isolation-based anomaly detection method, called *Isolation Forest* or iForest. This method explores the fact that anomalies tend to be in smaller number and isolated (in terms of attribute values) from normal observations [5]. The method explores such characteristics of anomalies to isolate them close to the root of binary trees that represent the available data. This method has smaller asymptotic complexity than other popular anomaly detection methods such as k-Nearest Neighbors (k-NN) and Random Forests.

A key challenge related to utilization of iForest for anomaly detection in cloud infrastructures relates to the nature of the infrastructure monitoring data such as CPU, memory, disk, and network usage are time-series data. This is because resource usage tends to vary along the day and usually present seasonal pattern. For example, it can be expected that a disk will have high volume of access in a given day and time (say, Saturdays at midnight, when a backup process is scheduled to execute), and not in other periods. Thus, the date and time where an event occurs is as important as the monitored value itself for anomaly detection in clouds. Existing literature regarding iForest, including applications of the method, did not explore its feasibility for anomaly detection in time-series data.

Given the benefits of iForest, such as small asymptotic complexity, and focus on identifying anomalies rather than normal instances, this method is an excellent candidate to be applied for anomaly detection in cloud data centers—a domain where timely detection of anomalies *is critical to enable QoS specified in SLAs between cloud providers and customers*. To this end, in this paper we investigate the feasibility of iForest for anomaly detection in large-scale cloud data centers.

The contributions of this paper are: (i) we propose a method to code time-series information as extra attributes that enable temporal anomaly detection in cloud data centers; and (ii) we investigate and establish its feasibility to adapt to seasonality and trends in the time-series; and (iii) we study the feasibility of the approach to be applied on-line and in real-time.

The rest of this paper is organized as follows. Section 2 discusses related work in the areas of data analytics supporting cloud operations and anomaly detection in clouds. Section 3 discusses the motivation for this work and the iForest anomaly detection method, the technique utilized in this paper. Section 4 introduces the dataset used for evaluation. Sections 5 and 6 discuss the effects of seasonality and trend, respectively, in the anomaly detection process. Section 7 evaluates the viability of the method for on-line anomaly prediction and Section 8 presents conclusion and future research directions.

2. RELATED WORK

In this section, we discuss related works in three key areas: (i) anomaly detection algorithms; (ii) data analytics supporting cloud operations; and (iii) anomaly detection in clouds.

2.1. Anomaly detection algorithms

Anomaly detection is a well-developed area with different approaches to different contexts. Several surveys were written that explore different techniques from a conceptual perspective [2, 7] or for specific applications (e.g., Estevez-Tapiador et al. [8] for networks, Xie et al. [9] for wireless networks, and García-Teodoro et al. [10] for intrusion detection). These works classify anomaly detection algorithms in a number of classes based on the method used for detection. Another relevant approach for anomaly detection is ORCA [11], which is an improvement over k-NN for anomaly detection that applies randomization and pruning in the search space to reduce the complexity of the detection to nearly linear time. However, as pointed out by Liu et al. [5], these traditional methods target identification of normal instances and can be computationally intensive for training or detection, what are undesirable characteristics in our application scenario.

The method utilized for anomaly detection in this work, iForest [5, 12], is detailed in Section 3.2, and works in the principle of *isolation* of anomalous instances. It has been explored in different contexts including anomalies in taxi trajectories [13], wireless networks [14], and clustered anomalies [12]. Tan et al. [15] propose a method, called HS-Trees, that applies principles of isolation-based anomaly detection in streaming data. Although data generation in our domain scenario can be viewed as stream, it is also a time-series and thus the moment in time when a data point is generated is important in the decision-making process. Different studies investigating workload prediction based on time-series in the context of web servers [16, 17] demonstrated that web workloads present seasonal behavior. Seasonality, if not taken in consideration by anomaly detectors, lead to classification errors. So, our proposed approach for time-series coding can complement the work by Tan et al. [15] by enabling expected seasonal behavior not to be interpreted as a change in the distribution of data.

None of the previous works explored how isolation-based anomaly detection can be successfully applied for anomaly detection in time-series data. We investigate the suitability of the method for anomaly detection in time-series data, in particular in the context of anomalies in cloud data center infrastructures.

2.2. Data analytics supporting cloud operations

Data analytics has been used as a means to support cloud operations. Islam et al. [18] investigated the application of Artificial Neural Networks (ANN) and Linear Regression to optimize the resource provisioning process from the perspective of a cloud user launching new instances to support growing application demand. Rather than provisioning, our approach targets anomalies in the pattern of resource utilization by virtual machines that are already running, and thus it is complementary to the method by Islam et al. [18].

Dean et al. [19] presented a method for prediction of performance anomalies in cloud infrastructures based on Self Organizing Maps. The approach is decentralized and models are built for each application, utilizing hypervisor-level information (resources requested by VMs to the hypervisor) and VM-level information. Our approach targets detection rather than prediction, and therefore could be applied concurrently with the approach by Dean et al. [19], so it could rectify anomalies caused by errors in the prediction.

Davis et al. [20] presented an adaptive approach for prediction of CPU usage in cloud data centers based on linear regression and discrete Fourier Transform. The models are built for every machine in the data center, and the approach focuses only on the CPU utilization of servers. Rather than prediction, our approach focuses on detection of anomalies, and includes not only CPU, but also memory, network, and disk. As in the previous case, our approach could be applied jointly to overcome prediction errors.

Yu et al. [21] applied clustering and ANN on information about workload of existing applications to predict workload characteristics of new tasks. When an unseen task arrives, it is initially mapped to one of the existing clusters, and then task requirements are predicted using the ANN that was trained for the chosen cluster. Zhang et al. [22] applied deep-belief network to predict the number and characteristics of resource requests that will arrive in a cloud data center. These works focus on the problem of predicting future resource requirements of tasks and characteristics of tasks

respectively, whereas our work focuses in detecting anomaly in resource utilization of virtual machines, without requiring information about the application workload.

Neves et al. [23] applied analytics to optimize runtime performance of the data center network. In particular, it applies predictive techniques to estimate the required communication demand of MapReduce applications and uses this information to reconfigure flows via Software-Defined Networking techniques. Whereas the approach by Neves et al. [23] targets data center networking, our approach targets data centers' computing resources.

These works demonstrate the potential that data analytics has to improve management of cloud infrastructures. Next, we discuss research related to anomaly detection in clouds.

2.3. Anomaly detection in clouds

On the particular topic of anomaly detection in clouds, research has been carried out with different objectives. Ibidunmoye et al. [24] presents a comprehensive review of the topic of performance anomaly detection and bottleneck identification in clouds. We focus here either on works that are closely related to our approach in terms of target environment and objectives or on works that are not included in the survey by Ibidunmoye et al. [24].

Bhaduri et al. [4] proposed a method for anomaly detection in cloud infrastructures based on k-NN. It detects anomalies in both infrastructure and resource management level (anomalies in the job submission process, job runtime, machine boot time), and can detect nodes in a cluster behaving differently from other nodes. Because the approach requires all-to-all communication among hosts, it is suitable for small-scale infrastructures. Furthermore, k-NN is a method that incurs high asymptotic complexity (namely $O(n^2)$) and thus loses performance drastically with the increase in number of data points (hosts, in the context of the work). Our approach, on the other hand, is suitable for data centers with thousands of hosts, as it relies on anomaly detection methods with low complexity and optimized for finding the anomalies.

Tan et al. [3] proposed PREPARE, an approach for anomaly detection at application level using Markov chains. It monitors virtual machine metrics at the same time it checks if service-level objectives of the applications (e.g., response time, throughput) are being met. Thus, PREPARE can identify virtual machines whose behavior, in terms of resource utilization and application performance, are different from other VMs in the cluster. Dean et al. [25] proposed an approach for detecting anomalies in the performance of applications running on IaaS clouds. It inspects the traces of system calls issued by applications to detect anomalies in the execution time of system calls or on the pattern of calls generated by the application. It can detect machines that are slower than others while processing system calls and can detect abnormal system calls, which may indicate malicious behavior. These approaches are more suitable for small-scale infrastructures, and are designed to detect problems in the applications rather than in the physical infrastructure, which is the target of our approach.

Vallis et al. [26] applied anomaly detection in large-scale time-series of service level metrics. It uses time-series decomposition and robust statistics to identify anomalous behavior in the service operation. The approach is mostly applied a posteriori, i.e., to identify anomalous behavior in utilization of virtual machine resources after it occurred. Our approach, on contrary, aims at real-time anomaly detection in the infrastructure.

Shen et al. [27] proposed a method to detect performance anomalies in software systems such as Operating Systems and application servers. The approach is probabilistic and is based on a reference execution of the software which is seen as achieving expected performance. It utilizes application and operating system performance and configuration information, and can detect software systems that are anomalous. It requires introspection in the virtual machine, whereas our approach, suitable for IaaS, targets resource-level information which can be obtained via hypervisor without requiring access to the VM.

Jehangiri et al. [28] applied anomaly detection in clouds using the Holt-Winters forecasting method as a MapReduce application. The anomaly detection is performed when a violation in application metrics is detected, and thus it can detect anomalous VMs. Frattini et al. [29] utilize the concept of *invariant* mining to detect anomalies in the execution of SaaS application. Invariant

is defined as a system property that is guaranteed to be observed during its execution, and thus their violation is a strong indicator of anomalies [29]. The method is used to detect violation in application performance metrics such as execution time and throughput, and can indicate when the application is experiencing problems. Our method is applied in the context of IaaS cloud infrastructures, where providers do not have access to information about the application layer to drive their decisions.

Solaimani et al. [30] proposed an anomaly detection system that handles VM monitoring data as streams and performs clustering to classify the typical behavior of VMs. Machines whose behavior do not fit into the clusters are labeled as anomalous. This approach is not suitable in the context of IaaS (our target service model), where the multitude of clients, operating systems, web and application servers, and applications cause behavior among VMs to vary drastically, thwarting clustering algorithms to be used effectively.

Doelitzscher et al. [31] proposed an approach for detection of abnormal utilization of IaaS cloud resources, which may indicate malicious usage of the cloud. The approach is based on ANN, and is applied on VM management operations (creation, destruction, and migration) with focus on security. Our approach targets resource utilization metrics with focus on performance.

Xu et al. [32] addressed the problem of detecting anomalies during the execution of sporadic operations, such as deployment, integration, and reconfiguration, in cloud applications. The approach detects anomalies in the form of errors in the execution of the sporadic operation. It also provides root cause analysis of anomalies observed during such events. Our approach, on the other hand, targets non-sporadic events, specifically anomalies in resource utilization observed during application runtime.

Huang et al. [33] proposed an approach that combines Local Outlier Factors (a semi-supervised anomaly detection algorithm) and Symbolic Aggregate Approximation (a methodology to represent time-series) to identify anomalies in the process of live migration of virtual machines. It analyzes resource-level metrics and can identify machines that experienced issues during live-migration. Rather than live migration of VMs, our approach focuses on unsupervised anomaly detection in the performance of running virtual machines.

Farshchi et al. [34] proposed a regression-based approach for anomaly detection during DevOps application operations. It utilizes information from log files and infrastructure monitoring tools and can detect problems during operations such as backup, application upgrade, migration, reconfiguration, auto-scaling and deployment. Our approach, on the other hand, focus on the “steady” stage of the life of a virtual machine, and thus both approaches could be used side-by-side: as the moment that DevOps operations are triggered are known, the anomaly detection method could be switched when these operations occur, reverting back to our approach at the end of the operation.

Zhang et al. [35] proposed an approach for anomaly detection in cloud applications based in clustering. It detects and identifies *application threads* that have abnormal behavior by analyzing system-level data. As stated above, our method targets IaaS clouds where providers do not have access to the applications running on a VM, and can detect anomalies without requiring such low-level access to VMs.

3. MOTIVATION AND BACKGROUND

This section introduces the main challenges motivating this work and presents a brief introduction to the Isolation Forest algorithm applied in this paper.

3.1. Challenges in anomaly detection in clouds

IaaS cloud data centers are composed of tens of thousands of physical servers that support the operation of tens or hundreds of thousands of virtual machines. The VMs host a multitude of combinations of operating system, application containers, web servers, and user-specific applications. Regardless of what is hosted in the VMs, providers must deliver the expected

performance to the application. Usually, performance expectations are coded in the form of Service-Level Agreements (SLAs) that determine the Quality of Service (QoS) to be provided, penalties for the provider case they are not achieved, and rewards when they are. Because IaaS providers do not have control over the usage of VMs, QoS is usually defined in terms of availability and resource allocation to VMs (for example, memory, CPU, I/O, and network).

To ensure SLA compliance, the cloud infrastructure is constantly monitored by providers. Monitoring software, such as Nagios[†], and CA Application Performance Management[‡], can provide instant information about physical resource utilization of VMs and hosts. However, given the scale of cloud data centers, large volume of monitoring information is generated in a given time, and there is the need to this data to be timely processed in order to reduce performance issues caused by ongoing infrastructure problems when they occur.

The challenge faced by operators of such infrastructures is how to quickly identify unusual behavior in hosts and/or VMs, via analysis of monitoring data that can indicate that some incident is taking place, either at application or at infrastructure level. This is required to enable quick reaction to anomalies that may lead to SLA violations. While static, manually set thresholds in resource utilization can provide a simple solution for the problem, it cannot account for heterogeneity in applications. This can lead to an excess of false-positives, because some applications may have usual patterns of utilization that are higher from manually set values, or false-negatives, in the opposite case. Manually fine-tuning the thresholds for each VM or application is not a scalable solution for the problem, besides being a costly operation, as it requires substantial human intervention.

The above facts demonstrate that machine learning-based solutions are the preferred method for this problem. In particular, we argue that an effective solution for automatic anomaly detection in the context of large-scale cloud data centers should meet the following requirements:

- *Real time operation.* The solution must be able to detect anomalies as they happen, to allow data center operators to take actions in timely manner to minimize the risk of SLA violations.
- *Unsupervised learning.* Because of the huge heterogeneity of software executing in a VM, and because cloud providers do not have access to customer VMs, it is impractical for providers to label the anomalies in the data. Vallis et al. [26] also point out the challenge of velocity in the monitoring generation as another obstacle for labeling anomalies in monitoring data.
- *Adaptability.* Workloads change over time. The application hosted on a VM can be changed by the VM owner, or it can be updated, changing its behavior. Utilization of the hosted application can increase due to grow of popularity, leading to a positive trend. The solution should adapt to these changes.
- *Robustness to seasonal behavior.* Cloud applications usually present a pattern of utilization that contains expected peaks and troughs. The solution should account for this fact when signaling anomalies.
- *Low computational cost for training.* Thousands of different applications can be present in a cloud data center in a given time. Moreover, clouds are dynamic environments, where new applications can be quickly added and removed and workloads can change quickly. Thus, it is important that the computational complexity for training models is low, so the amount and time of resources dedicated to this task can be reduced, freeing resources to the core business of service providers (i.e., hosting customer's applications).

Among the works in the area of anomaly detection in clouds discussed in Section 3.1, only Bhaduri et al. [4] and Vallis et al. [26] target anomaly detection at infrastructure level. However, none of these works target real-time anomaly detection (the first of the above requirements). Furthermore, the work by Bhaduri et al. [4] is based on k-NN, and results presented in Section 6 for a method based in k-NN show that it is not very robust to seasonal behavior. Thus, the method does not meet the fourth requirement above. The work by Vallis et al. [26], however, meets all the other requirements.

[†]<http://www.nagios.org>

[‡]<http://www.ca.com/us/opscenter/ca-application-performance-management.aspx>

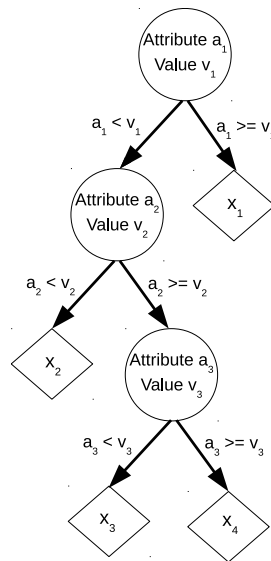


Figure 1. One Isolation Tree of an iForest. Non-terminal nodes represent attributes a and split values v for data points to the two subtrees. Terminal nodes are data points. Anomalies are more likely to be isolated closer to the root of the tree than normal points. In this example, only the data point x_1 has a value for attribute a_1 greater than v_1 .

Among different solutions for anomaly detection available in the literature, isolation-based methods [5] have characteristics that make them promising to meet all the above requirements. Next, we discuss iForest, the method for anomaly detection that is utilized in this paper in the context of large-scale cloud data center monitoring.

3.2. Isolation Forest

Isolation Forest[§], or iForest [5, 6] is an unsupervised anomaly detection method based on the isolation principle. It explores the fact that anomalies tend to be data points that are distant from normal points of the data set. Given this property, these points are more likely to be isolated (separated) from normal points by random partitions in the attributes space. It means that less random partitions in the attributes space are necessary to isolate an anomaly than a normal point.

An Isolation Forest is composed of a set of Isolation Trees. An Isolation Tree is a binary tree where leaves are data points and non-terminal nodes contain an attribute a and the attribute value v . Points in the two subtrees are split according to v , so all remaining points are sent to a subtree depending whether the points' value for a is less than, or greater or equal than v . The tree is recursively built by random selection of attributes and values until all the points are in a leaf of the tree (see Figure 1). Each tree in the forest is built with a subsample of the whole data set. Because of the isolation principle, points that represent anomalies are more likely to be isolated closer to the root of the trees.

In the evaluation stage, data points traverse each tree of the forest until they reach a terminal node (i.e., they are isolated) or until a maximum traversal length l is reached. The choice of l affects the granularity of the anomaly detection. The path length $h(x)$ obtained on each tree while the data point traverses the tree is recorded. It is expected that anomalous points will, in average, traverse a shorter path on the forest than normal points. Thus, the output of the process—the *Anomaly Score* s —is a function of the normalized value of the average $h(x)$ among all trees, with $0 < s \leq 1$. The

[§]Not to be confused with *Random Forest*, a classical and widely used machine learning classifier proposed by Breiman [36].

higher the s , the more likely the point is an anomaly. Tuning parameters of the algorithm are the subsample size ψ , the number of trees t , and maximum path length l .

The effectiveness of iForest for anomaly detection in large-scale cloud data centers is investigated in this paper because the method has properties that meet the requirements listed in the previous section. It is an unsupervised anomaly detection method and is promising for real time utilization because of its lower asymptotic complexity, both in training and evaluation stages—respectively $O(t\psi^2)$ and $O(nt\psi)$, where n is the testing data size [5]—compared to competitive approaches such as k-NN and Random Forest [6].

Previous cases of use of iForest did not investigate its suitability for time-series data such as monitoring data from cloud infrastructures. In particular, seasonality and trends are challenging aspects that, if not accounted for, can impair the anomaly detection process [26]. Thus, in the rest of this paper, we describe how we adapted our dataset to enable iForest to operate on it without losses in the temporal aspect of the data and evaluate the method for different scenarios observed in cloud infrastructures.

4. DATASET AND DERIVED ATTRIBUTES

In this section, we discuss the dataset we utilized and the derived attributes we added in the analysis to increase the performance and to encode the temporal aspect of the attributes in the anomaly detection process.

4.1. Dataset

The dataset we utilized contains monitoring data from a subset of IaaS data centers from an Information Technology company that operates worldwide. The analyzed cloud infrastructure comprises 30 data centers spread across 18 countries in 3 continents. Each data center contains a mix of servers using virtualization technology and servers where applications run in the bare hardware (non-virtualized). Virtualized servers utilize VMware ESXi[¶], which provides the monitored metrics. Data was collected by the data centers' monitoring software, which was aggregated in 30 minutes intervals. Sources of information are the data centers' resources (virtual machines, virtualized hosts, and non-virtualized hosts) and the collected metrics are:

- Disk Transfer Rate;
- Network Transfer Rate;
- CPU utilization; and
- Memory utilization.

The dataset contains dynamic information about a randomly sampled subset of the data centers resources (VMs). Each row of the table contains a single measurement obtained from a single resource on a given time for the four metrics, along with a timestamp. Thus, each row of the table has the following format: `<name, timestamp, disk rate, network rate, CPU utilization, memory utilization>`.

As the collection happens in 30 minutes intervals, a single day of observations contains 48 of such rows per host. The subset of hosts monitored in a single day varies. Thus, measurement for a single host can be missing for some days. However, there are no missing rows for a host that is monitored in a particular day, although rows can miss information for one or more metrics (but never for all metrics).

Information about utilization of each resource by each host has been organized in the form of time-series. The period of data analysis comprises 2 months. In order to guarantee confidentiality of the company and its customers, we arbitrarily refer to the first day of data collection as March, 1st, 2014 and the last day as May, 1st, 2014. Moreover, because the data analysis is performed

[¶]<http://www.vmware.com/products/esxi-and-esx>

individually in each VM, the y-axis of the plots in Figure 2 (described later) contains the normalized value of the metric with the highest observed value for the parameter on such particular VM in the time-series.

One particular limitation of the discussed method and the dataset is that it cannot detect annual season patterns, such as peak in access close to Christmas or other holidays. To be able to correctly identify annual patterns as anomalies, many years of data would need to be available for training purposes. Therefore, correct classification of annual patterns is outside the scope of this work.

4.2. Derived attributes

The dataset contains four variable attributes—CPU, memory, network, and disk utilization. Initial attempts of applying iForest with such small number of attributes did not lead to good anomaly detection power on the approach. Furthermore, the utilization of the measured attributes as collected lack the temporal aspect that, as discussed before, is important for monitoring of cloud data centers.

Our data analysis revealed that, besides the raw value observed in a given time, the difference between two successive measurements, for each attribute, increased the performed of the approach and doubled the number of available attributes. The differences of values observed in bigger intervals (up to one week) were also explored, with little or no improvements in the anomaly detection performance.

The use of 1-lag difference in attributes was not enough to capture the time-series behavior to the level required. This has been solved with the use of another set of derived attributes (one per original attribute) that encodes the expected seasonal behavior of the metric (based on previous observed values). For this purpose, we assume a weekly periodicity in the half-hour series, and obtain the *median of the sub-cycle series* [26] (each sub-cycle contains all the observed values at a given week day and time), which is defined as follows.

Consider a time-series data organized as a matrix $A \in \mathbb{R}^{m \times n}$, which contains m season cycles, each of size n . In our dataset, season cycles are weekly and observations are obtained hourly. Thus, if A contains 4 weeks of collected data, $m = 4$ (four weeks—thus four cycles—of data), $n = 168$ (24 samples are obtained per day during seven days in a cycle), and each column j represents one hour of one week day (e.g., Tuesday, 4pm).

Each column j of A is called *sub-cycle series* [26]. Consider a vector V whose element v_j contain the median of column j of A . In this case, v_j is said to store the median of the corresponding sub-cycle series:

$$v_j = \text{median}(a_{1,j}, a_{2,j}, \dots, a_{m,j}), 1 \leq j \leq n \quad (1)$$

The choice of medians rather than averages is due to the fact that the median is more robust to anomalies, and results in more robust anomaly detection, as demonstrated by Vallis et al. [26].

The extra set of attributes contains the difference between the observed value in a given time and the median of the sub-cycle series for the corresponding day of the week and time (i.e., the difference between the observed value x_j at a day and time j and v_j).

Table I presents a summary of all the explored attributes. In the rest of the paper, we explore how the iForest method and the derived attributes perform for anomaly detection in time-series data from cloud data centers.

5. DETECTION OF SEASONALITY

Studies demonstrated that web workloads present seasonal behavior [16, 17]. This means that peaks and troughs in requests can be expected at certain times. It is important that such periodic variations in the workload are correctly identified because their occurrence must not be regarded as anomalous. In fact, the opposite should be expected: expected peaks or troughs in demands that failed to occur should be regarded as anomalies. As discussed in Section 2, identification and detection of such seasonal behavior is a recurring issue in data analytics applied in the area of cloud computing.

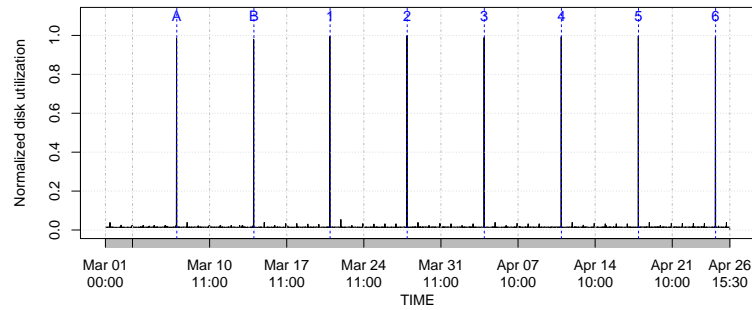
Table I. Attributes of the data used in this paper. The first four attributes are directly obtained from the dataset, and the remaining attributes are derived from them and assist in the capture of time-series behavior.

Attribute name	Meaning
Disk	Disk transfer rate (MB/s)
Net	Network transfer rate (Mb/s)
CPU	Utilization of CPU (%)
Mem	Utilization of memory (%)
Disk_diff	Difference between the current and previous observation of the Disk attribute
Net_diff	Difference between the current and previous observation of the Net attribute
CPU_diff	Difference between the current and previous observation of the CPU attribute
Mem_diff	Difference between the current and previous observation of the Mem attribute
Disk_SS_dev	Difference between the Disk observation and the median of the corresponding sub-cycle series
Net_SS_dev	Difference between the Net observation and the median of the corresponding sub-cycle series
CPU_SS_dev	Difference between the CPU observation and the median of the corresponding sub-cycle series
Mem_SS_dev	Difference between the Mem observation and the median of the corresponding sub-cycle series

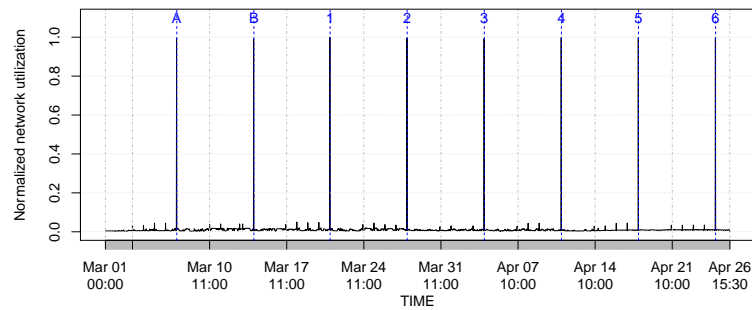
In this section, we evaluate the capacity of the iForest algorithm to detect and correctly classify seasonal patterns in the workload. Figure 2 shows the workload we use for evaluation, which relates to the monitoring of one particular server, extracted from the studied dataset. The time-series depicted in Figures Figure 2(a)–Figure 2(d) correspond respectively to disk access, network access, CPU usage, and memory usage (all the values are normalized with basis on the largest value for the attribute observed on the respective series). There is a visible seasonal pattern where a peak in resource utilization, more prominently seeing in the disk and network access, occurs weekly (because these values are normalized, such peaks have values of 1.0, or are very close to 1.0). Such points are labeled as 1–6 in the figure. Two other points, marked as A and B, occur earlier in the workload. However, these points are shifted in time from the rest of the points and thus they are not treated as belonging to the same seasonal behavior observed in the rest of the labeled points. The key point of this figure is to show that a seasonal behavior is present in this workload (starting from the third week), and thus points 1–6 should not be classified as anomalous.

To evaluate the capacity of adapting to seasonal behavior in workloads, the algorithm has been executed against the original workload and the values of s of the points of interest were collected. Then, the experiment was repeated, this time with a modified version of the workload where the peak marked as 1 was replaced by the average of the surrounding values for each attribute, thus removing the seasonal effect for that point. We continued the application of such procedure, each time removing another peak: first Point 2, than Point 3, and so on, until Point 5 was removed. The whole process has been repeated for different maximum traversal length l ($l = \{1, 3, 6, 9\}$). Every time a peak was removed, the remaining ones were less of an expected behavior than before, until the point where peaks occurrences were rare, and thus should be characterized as anomalous.

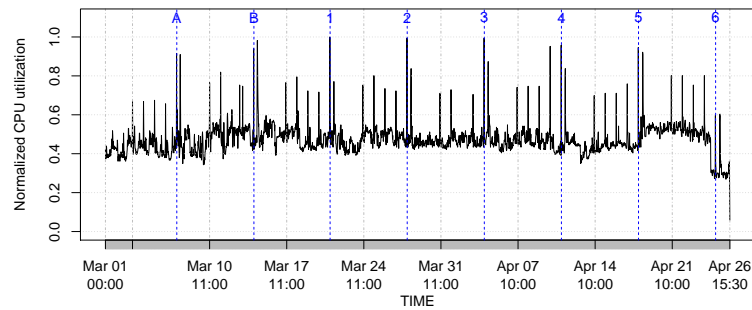
Figure 3 shows how the resulting anomaly scores computed for each labeled point changed as peaks were removed from the workload. As the number of peaks is reduced, the score of a given point tends to increase. The degree of increment is higher for $l = 1$ and decreases as l increases. Thus, the choice of a proper value for the anomaly score s (which is workload-dependent, and is also affected by l) enables the classification of seasonal peaks as non-anomalies. Such a cut-off value is 0.60, 0.68, 0.80, and 0.83, respectively, in Figures 3(a)–(d). Results show that iForest enables detection of seasonal behavior by correctly attributing lower anomalies scores for points that are significantly different from the expected if similar behavior is observed in similar times in the workload. Points with same values but occurring in a different pattern of time (points A and B)



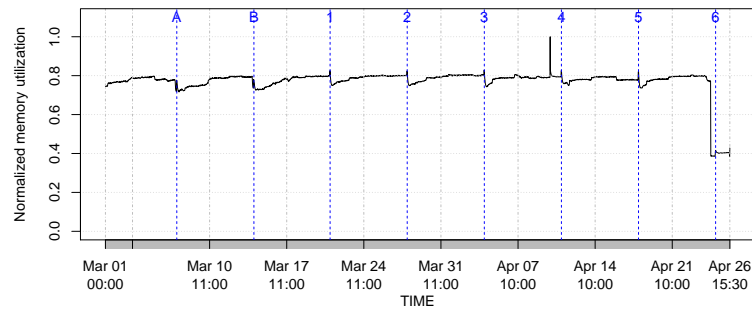
(a)



(b)



(c)



(d)

Figure 2. Workload of one server obtained from the dataset, which is utilized to evaluate detection of seasonal behavior by iForest. Labeled points correspond to the peak of the workload and have values close to 1.0. Points labeled A and B are slightly shifted in time from the points labeled 1–6, and thus they are not part of the weekly cycle observed in the rest of the labeled points. (a) Disk (b) Network (c) CPU (d) Memory.

Table II. Evaluation of different cut-off values for the Anomaly Score s in the number of anomalies generated.

Workload	Cut-off value								
	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9
Seasonal, no peaks removed	36	22	16	11	8	3	2	0	0
Seasonal, 1 peak removed	34	21	14	10	7	3	2	0	0
Seasonal, 2 peaks removed	36	21	11	9	6	6	3	0	0
Seasonal, 3 peaks removed	36	18	13	7	6	5	5	1	0
Seasonal, 4 peaks removed	34	13	8	5	5	4	4	1	0
Seasonal, 5 peaks removed	47	14	8	6	4	3	3	2	0

are also correctly awarded high anomaly scores by iForest, what makes the algorithm suitable for use for anomaly detection in time-series with seasonality.

Determination of the cut-off value for anomaly detection is application-specific and influenced by the frequency of occurrence of peaks and troughs in the workload. To exemplify this, Table II shows the number of detected anomalies for different cut-off values for the workloads discussed in this section with $l = 6$. Besides the expected effect that lower cut-offs causes too many points being considered anomalous, the table shows that, the more regular the patterns observed in a workload (i.e., the more the seasonal behavior was replaced by average of surrounding points), the higher is the Anomaly Score, and thus the cut-off value for the most salient anomalies.

The implication of the above is that the choice of cut-off value needs to take in consideration the cost of false-positives (unnecessary deployment of technical staff in the data center), the cost of false negatives (SLA violations and loss of customer confidence), and the degree of variability in the workload: low cut-off values and/or higher values for l generate more reported anomalies and thus result in the former (more technician calls), whereas high cut-off values and/or low values for l result in the latter (more potential SLA violations).

In situations where the workload contains seasonal trends with chaotic daily behavior, it could be expected high hourly variation, what reflects in larger values for the *_diff attributes. If this is the regular behavior, periods of stable utilization might be considered the abnormal behavior. This would reflect in close to zero for *_diff attributes, contributing to higher Anomaly Scores for this particular case. Nevertheless, it is still likely that this type of workload would require lower cut-off values to be properly detected.

6. DETECTION OF TRENDS

Another important aspect of anomaly detection in the context of cloud computing regards the effect of trends in the detection. The cumulative effect of trends in the time-series makes the values towards the end of the series to be significantly different from those at the beginning. This in turn may lead detectors unaware of the effect to yield more false positives towards the end of the series [26].

To investigate the effect of trends in our approach for anomaly detection based on iForest, we modified the workload depicted in Figure 2 and added a trend element to all the attributes. To this end, we injected a cumulative increase in load of 5% per week for the duration of the workload. If the trend affects the outcome of the anomaly score, then it is expected that scores towards the end of the series will be different from those at the beginning of the series, for the workload with added trend.

To investigate how sensitive is the method to trends, we conducted a paired t-test on values of s generated by the original workload and the one with the added trend. The experiment was repeated for different maximum traversal length l ($l = \{1, 3, 6, 9\}$).

The paired t-test for the experiment with each l showed a slight decrease in the mean s when the trend was injected in the workload. The mean reduction was respectively 0.002, 0.004, 0.005, and 0.006 for $l = \{1, 3, 6, 9\}$ (p-values $< 2.2e^{-16}$ for all cases). This corresponds to a worst case

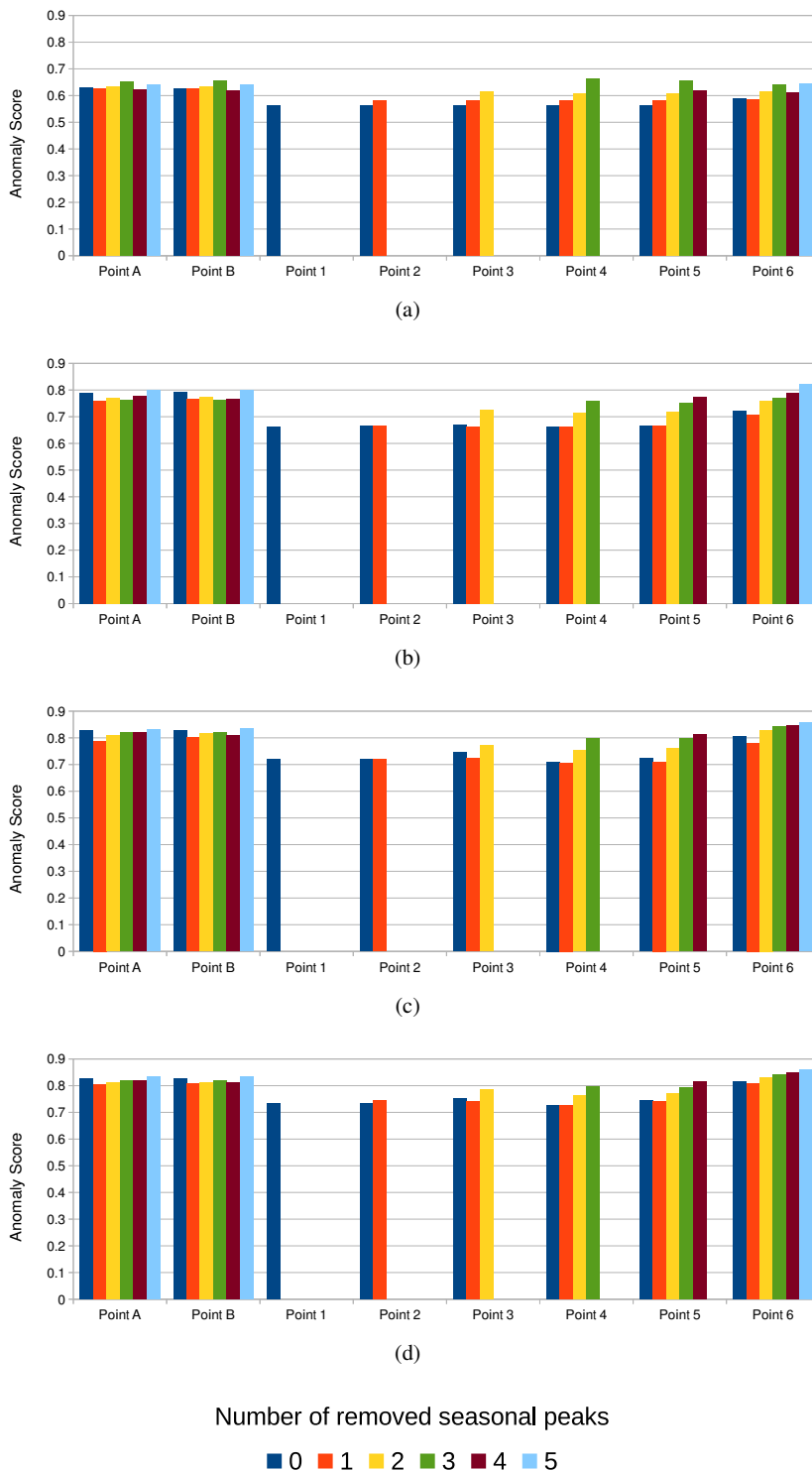


Figure 3. Evaluation of the effect of seasonal behavior detection in iForest. The anomaly score of the points labeled in Figure 2 were calculated with different number of peaks removed from the workload. The algorithm correctly assigns lower anomalies scores for points with different behavior if similar behavior is observed in similar times in the workload. Each plot corresponds to one value of maximum traversal length l . (a) $l = 1$ (b) $l = 3$ (c) $l = 6$ (d) $l = 9$.

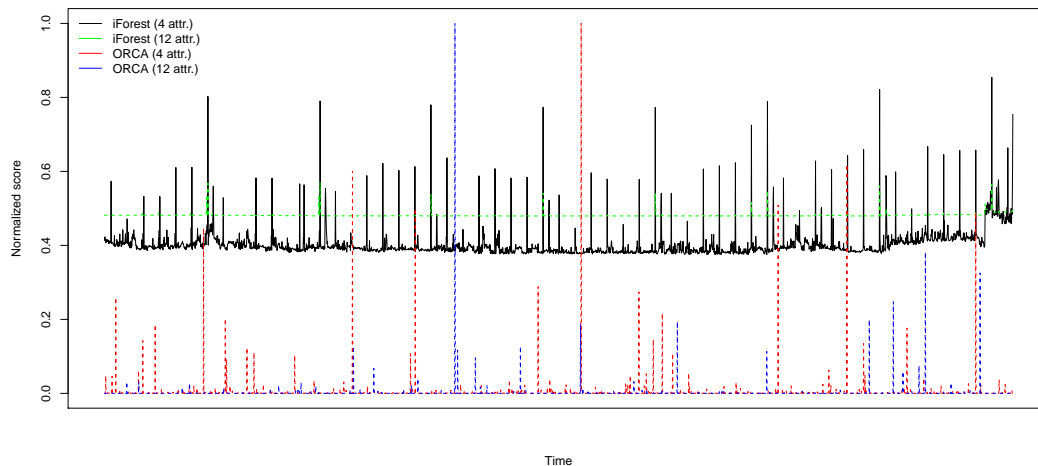


Figure 4. Variation of Anomaly Score along the time in a time-series workload with injected trend. iForest is executed with and without derived attributes. Results are also compared with normalized scores produced by ORCA, an unsupervised, k-NN-based anomaly detection algorithm, with and without derived attributes.

variation of less than 2% in the mean value of s , which does not affect the classification performance from a practical perspective. Thus, we can conclude the method is robust against trends in the time-series.

To determine if the robustness against trends is caused by the derived attributes or by iForest itself, we repeated the above experiment using iForest without the derived attributes and using another unsupervised anomaly detection method, namely ORCA [11], a k-NN-based anomaly detection algorithm. ORCA has been chosen for the evaluation because it is an unsupervised learning method (a requirement in the context of this work) with near linear asymptotic complexity for anomaly detection. Output provided by ORCA (with and without derived attributes) has been normalized in relation to the highest anomaly value generated by the method. Tests with iForest used 100 trees and $l = 9$ (which captures the largest reduction in the mean in the previous experiment).

Figure 4 shows the values of s along the time for iForest with and without derived attributes and ORCA. The figure shows that when the derived attributes are not present, scores oscillate significantly for longer traversal lengths, rendering the result ineffective. Furthermore, when derived attributes are not used, more anomalies are generated towards the end of the series, an effect that had been already documented by Vallis et al. [26]. ORCA also provided more clear anomalous points when the derived attribute were used. Thus the extra attributes contribute to the classification capability of both iForest and ORCA in the discussed context.

7. ON-LINE ANOMALY DETECTION

The analysis we performed so far is suitable for off-line analysis of the cloud infrastructure: by observing a large dataset composed of historical data, anomalies observed in the infrastructure can be detected. As discussed earlier, one of the advantages of iForest over competitors is the good performance of the method to quickly detect anomalies, which would make the method suitable for real time anomaly detection.

To understand the capacity of iForest of being applied in real time, we conducted simulation experiments with the available dataset. The experiment consists in utilizing only information obtained before a given time t to detect anomalies at time t . In the experiment, we utilize the first week of traces to perform an initial training of the model for each server. Starting from Week 2, anomaly detection is performed every 30 minutes. This time interval is chosen to align with the

Table III. Coefficient of correlation between Anomaly Scores obtained via on-line and off-line analyses for different values of maximum traversal length l . Seasonal workloads correspond to those used in Section 5, while trend refers to the workload used in Section 6.

Workload	l			
	1	3	6	9
Seasonal, no peaks removed	0.85	0.91	0.91	0.91
Seasonal, 1 peak removed	0.88	0.94	0.94	0.93
Seasonal, 2 peaks removed	0.91	0.95	0.92	0.92
Seasonal, 3 peaks removed	0.94	0.97	0.95	0.94
Seasonal, 4 peaks removed	0.94	0.95	0.93	0.92
Seasonal, 5 peaks removed	0.92	0.94	0.93	0.92
Trend	0.87	0.91	0.92	0.92

dataset used in this paper—longer or shorter time intervals could be used as well without any loss in generality.

At the end of each day (starting from Week 2), the model for each VM is updated at midnight, and the updated model is used for detection for the next day. The model is updated with the use of a 1-week sliding window (except for median calculation, where all the previous values are used). Shorter update intervals would be of little value, as a large amount of computation would be carried out with only a small amount of extra data to be incorporated to the model. Longer update intervals could be used as well (for example, the update could run in a batch over the weekend, when the demand for computing resources tend to be lower).

The currently available technologies enable implementation of such approach: for example, streaming processing frameworks (such as Apache Storm^{||}) enable the collection of real time data and the anomaly detection to be performed instantly. The daily retraining can be carried out by batch processing frameworks such as Hadoop^{**}.

We investigate the relationship between Anomaly Scores obtained in real time against scores obtained in off-line analysis. To this end, we employ the same data utilized in Section 5 and depicted in Figure 2, including all the variations with removed seasonal behavior, so our analysis contains different levels of seasonality. We also apply the same approach with the workload discussed in Section 6 to add trend to the time-series. Therefore, in total seven correlation analyses, with different degrees of seasonality and trends are investigated. The whole experiment was repeated for different values of l ($l = \{1, 3, 6, 9\}$, as in previous sections).

The coefficient of correlation of the values of s obtained with the two approaches (on-line and off-line) for the different workloads are depicted in Table III. Results show a strong correlation between values of s obtained via on-line and off-line techniques. In fact, although there are statistically significant difference between s obtained with on-line and off-line anomaly detection (confirmed by paired t-tests), this difference, for all cases, was found to be smaller than 1% of the average s , what does not affect the outcome of the anomaly detection. In terms of runtime, calculation of anomaly scores of 2722 data points with $l = 9$ (using R) took 28 ms in a machine with Intel Core i7 2600 (Quad core, 3.40 GHz, 8MB of cache) with 8 GB of RAM, and the corresponding iForest data structure utilized only 22.8MB of RAM. The training time was in 579 ms (average of 100 repetitions of the training process). All these results enable us to conclude that iForest is a suitable algorithm for real-time anomaly detection in cloud environments.

^{||}<https://storm.apache.org/>

^{**}<https://hadoop.apache.org/>

8. CONCLUSIONS AND FUTURE WORK

We investigated the applicability of the iForest anomaly detection algorithm for detection of abnormal events in resource utilization of large-scale cloud data centers. Initially, we demonstrated how time-series information was extracted into extra attributes that enabled temporal anomaly detection. Next, we investigated the capacity of the method in detecting seasons and trends in the data set was explored, along with the method's feasibility for on-line and real-time anomaly detection.

As future work, we will investigate the applicability of the method for customer segmentation in terms of Quality of Service in the reaction to detected anomalies. We will also investigate the problem of anomaly detection in the services hosted by the cloud platform using application-level Quality of Service metrics.

REFERENCES

1. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* June 2009; **25**(6):599–616.
2. Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. *ACM Computing Surveys* July 2009; **41**(3):15:1–15:58.
3. Tan Y, Nguyen H, Shen Z, Gu X, Venkatramani C, Rajan D. PREPARE: Predictive performance anomaly prevention for virtualized cloud systems. *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2012.
4. Bhaduri K, Das K, Matthews BL. Detecting abnormal machine characteristics in cloud infrastructures. *Proceedings of the 11th International Conference on Data Mining Workshops (ICDMW)*, 2011.
5. Liu FT, Ting KM, Zhou ZH. Isolation forest. *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, 2008.
6. Liu FT, Ting KM, Zhou ZH. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data* Mar 2012; **6**(1):3:1–3:39.
7. Patcha A, Park JM. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks* Aug 2007; **51**(12):3448–3470.
8. Estevez-Tapiador JM, Garcia-Teodoro P, Diaz-Verdejo JE. Anomaly detection methods in wired networks: a survey and taxonomy. *Computer Communications* Oct 2004; **27**(16):1569–1584.
9. Xie M, Han S, Tian B, Parvin S. Anomaly detection in wireless sensor networks: A survey. *Journal of Network and Computer Applications* Jul 2011; **34**(4):1302–1325.
10. Garca-Teodoro P, Daz-Verdejo J, Maci-Fernndez G, Vzquez E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security* Feb-Mar 2009; **28**(1–2):18–28.
11. Bay SD, Schwabacher M. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003.
12. Liu FT, Ting KM, Zhou ZH. On detecting clustered anomalies using SCiForest. *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2010.
13. Zhang D, Li N, Zhou ZH, Chen C, Sun L, Li S. iBAT: Detecting anomalous taxi trajectories from GPS traces. *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp)*, 2011.
14. Ding ZG, Du DJ, Fei MR. An isolation principle based distributed anomaly detection method in wireless sensor networks. *International Journal of Automation and Computing* Aug 2015; **12**(4):402–412.
15. Tan SC, Ting KM, Liu TF. Fast anomaly detection for streaming data. *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.
16. Urdaneta G, Pierre G, van Steen M. Wikipedia workload analysis for decentralized hosting. *Computer Networks* Jul 2009; **53**(11):1830–1845.
17. Tran VG, Debuschere V, Bacha S. Hourly server workload forecasting up to 168 hours ahead using seasonal ARIMA model. *Proceedings of the 13th International Conference on Industrial Technology (ICIT)*, 2012.
18. Islam S, Keung J, Lee K, Liu A. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems* Jan 2012; **28**(1):155–162.
19. Dean DJ, Nguyen H, Gu X. UBL: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems. *Proceedings of the 9th International Conference on Autonomic Computing (ICAC)*, 2012.
20. Davis I, Hemmati H, Holt RC, Godfrey MW, Neuse D, Mankovskii S. Storm prediction in a cloud. *Proceedings of the Principles of the 2013 ICSE Workshop on Engineering Service-Oriented Systems (PESOS)*, 2013.
21. Yu Y, Jindal V, Yen IL, Bastani F. Integrating clustering and learning for improved workload prediction in the cloud. *Proceedings of the IEEE 9th International Conference on Cloud Computing (CLOUD)*, 2016.
22. Zhang W, Duan P, Yang LT, Xia F, Li Z, Lu Q, Gong W, Yang S. Resource requests prediction in the cloud computing environment with a deep belief network. *Software: Practice and Experience* Mar 2017; **47**(3):473–488.
23. Neves MV, Rose CAFD, Katrinis K, Franke H. Pythia: Faster big data in motion through predictive software-defined network optimization at runtime. *Proceedings of the 28th International Parallel and Distributed Processing Symposium (IPDPS)*, 2014.

24. Ibdunmoye O, Hernández-Rodríguez F, Elmroth E. Performance anomaly detection and bottleneck identification. *ACM Computer Surveys* Sep 2015; **48**(1):4:1–4:35.
25. Dean DJ, Nguyen H, Wang P, Gu X. PerfCompass: Toward runtime performance anomaly fault localization for Infrastructure-as-a-Service clouds. *Proceedings of the 6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2014.
26. Vallis O, Hochenbaum J, Kejariwal A. A novel technique for long-term anomaly detection in the cloud. *Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*, 2014.
27. Shen K, Stewart C, Li C, Li X. Reference-driven performance anomaly identification. *SIGMETRICS Performance Evaluation Review* Jun 2009; **37**(1):85–96.
28. Jehangiri AI, Yahyapour R, Wieder P, Yaqub E, Lu K. Diagnosing cloud performance anomalies using large time series dataset analysis. *Proceedings of the 7th International Conference on Cloud Computing (CLOUD)*, 2014.
29. Frattini F, Sarkar S, Khasnabish JN, Russo S. Using invariants for anomaly detection: The case study of a SaaS application. *Proceedings of the 2014 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2014.
30. Solaimani M, Iftekhar M, Khan L, Thuraisingham B. Spark-based anomaly detection over multi-source VMware performance data in real-time. *Proceedings of the 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, 2014.
31. Doelitzscher F, Knahl M, Reich C, Clarke N. Anomaly detection in IaaS clouds. *Proceedings of the 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2013.
32. Xu X, Zhu L, Weber I, Bass L, Sun D. POD-Diagnosis: Error diagnosis of sporadic operations on cloud applications. *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2014.
33. Huang T, Zhu Y, Wu Y, Bressan S, Dobbie G. Anomaly detection and identification scheme for VM live migration in cloud infrastructure. *Future Generation Computer Systems* Mar 2016; **56**:736–745.
34. Farshchi M, Schneider JG, Weber I, Grundy J. Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. *Journal of Systems and Software* To appear. doi:10.1016/j.jss.2017.03.012; .
35. Zhang X, Meng F, Chen P, Xu J. TaskInsight: A fine-grained performance anomaly detection and problem locating system. *Proceedings of the IEEE 9th International Conference on Cloud Computing (CLOUD)*, 2016.
36. Breiman L. Random forests. *Machine Learning* October 2001; **45**(1):5–32.