DARC: High-dimensional Diffusing Anomaly Detection and Root Cause Location in Cloud Computing Systems

Anonymous authors
Submission ID: 106

Abstract

The modern cloud computing system has evolved into a highly dynamic and complex ecosystem with thousands of modules. Modifications and updates to these modules occur every day to accommodate customer needs. Unfortunately, these frequent changes may introduce anomalies to the system, whose diffusion can undermine the system performance and even cause system outage. Though very important, it is challenging to detect anomalies at the early stages and locate their root causes, due to the complexity of the cloud ecosystem and the huge number of attribute combinations. This paper proposes DARC for high-dimensional diffusing anomaly detection and root cause location in cloud computing systems. DARC uses first two-stage percentile analysis and Mann-Kendall score thresholding to detect rare anomalies, and then a bottom-up search strategy with three computational complexity reduction techniques to efficiently locate the root causes. Extensive experiments showed that DARC is able to accurately and efficiently locate root causes of diffusing anomalies. It has been successfully used in the daily practice of a cloud computing service provider.

CCS Concepts: • **Computer systems organization → Cloud computing**.

Keywords: high-dimensional diffusing anomaly detection, root cause location, cloud computing

1 Introduction

In recent years, the complexity of cloud systems along with the explosion of business scenarios, such as AI [8, 29], e-commerce [23, 25] and big data [20, 24], has reached an unprecedented level. To accommodate the surge of applications and more diverse scenarios, frequent changes and updates to the cloud system is necessary. For example, in large cloud computing service system, thousands of physical machines are online, hundreds of thousands of modules are updated, and multiple new functionalities are released every day. These changes and updates may degrade the service stability [30] and even cause system outage [10, 19], when a modification has hidden anomalies and is unfortunately unnoticed.

Typically, these anomalies exist on a few machines at early deployment stages under the control of gray release [27], and would gradually spread and undermine the system performance. Some anomalies can generate explicit error signals, such as out of memory (OOM) and kernel/disk errors, which can be used to trigger a rollback. However, most anomalies do not have obvious error signals at the early stages, and some even do not produce any error signals at all during the entire life cycle of anomaly diffusion. For example, the network transmission speed may fluctuate significantly for a period of time every day due to driver version defects, but no error events are generated, which makes existing error signals and events based approaches [16] ineffective. Therefore, it is necessary to perform anomaly detection and root cause location based on Key Performance Indicators (KPIs) [28].

However, it is an extremely challenging task to detect anomalies and locate their root causes at the early stages:

1. Analyzing the KPIs of the entire cluster generally cannot locate anomalies, because the scope of an anomaly is usually local. We therefore need to analyze the KPIs of some attribute combinations (e.g., XXX region & YYY product & software Version ZZZ) to locate the anomalies. As the number of attributes increases, the number of their combinations explodes. For example, the number of combinations is as high as v^d, if a system has d attributes, each with v different values. In an actual cloud computing system, the attribute dimensionality d is over 30, and each attribute may contain dozens of different values. It is an NP-hard problem of searching for the optimal combination in such a high-dimensional space.
2. Identifying the true abnormal attribute combinations is very difficult at the early stages, because each combination includes both normal and abnormal samples, and the rare abnormal ones are usually overwhelmed by the massive normal ones. Additionally, a good anomaly detection algorithm largely hinges on how well it tolerates attributes collected from various modules, which may be significantly different in their amplitudes, variance, and perturbations. Detecting instance anomaly by simple difference or proportion does not fit every scenario and may easily lead to false alarms.
3. Collecting all these high-dimensional attributes periodically is computationally infeasible. For example, a

virtual machine module in cloud systems contains network, storage, controller, and resource distribution modules, where the resource distribution module is further composed of virtual memory/CPU resource distribution, virtual switch settings, network bandwidth support, and so on. Monitoring all these small components is uneconomic, because it costs tremendous computation and storage resources (more than 3PB/day).

Although great progresses have been made, existing approaches failed to address all challenges mentioned above, because the anomaly diffusion problem has been overlooked. For instance, Hotspot [28] and Squeeze [17] can only locate root causes after the modification has been fully deployed, due to their Ripple Effect assumption, i.e., all samples under the root causes are abnormal. This is not true in early stages of anomaly diffusion, where only few samples become abnormal. Additionally, though various optimization techniques like Monte Carlo Tree Search and deviation based filtering [17] have been adopted, most state-of-the-art approaches struggle at processing millions of data with dimensionality larger than five, which is common in modern cloud computing systems.

This paper proposes a high-dimensional diffusing anomaly root cause (DARC) location system, to accurately and efficiently detect anomalies and locate their root causes in cloud systems at the early stages of diffusion. DARC implements automatic dimensionality reduction to significantly reduce the search space, and also sophisticated search algorithms to further improve the search efficiency.

DARC has at least two advantages over existing signal-based and ripple effect based approaches. First, DARC is able to detect not only obvious anomalies, but also less obvious ones, e.g., continuous decline in available memory, at an early stage. Existing signal-based approaches can only detect obvious anomalies. Second, DARC's root cause location algorithm has better performance and scalability. It can support up to 30 attributes, whereas existing approaches can only handle fewer than five attributes.

In summary, we make the following contributions:

1. We propose a two-stage percentile analysis plus Mann-Kendall (MK) score thresholding approach to reliably detect anomalies in each attribute combination, at the early diffusing stage.
2. We propose a tree based bottom-up search strategy to locate the root causes accurately.
3. We propose three techniques to reduce the computational cost of DARC, making it very suitable for large scale cloud computing systems.
4. We evaluated the performance of DARC on both semi-synthetic fault-injection data and real-world cloud systems. Results showed that DARC can accurately and efficiently locate the root causes with up to 30 attributes, unprecedent in previous works. DARC has been successfully implemented in the daily practice of a cloud computing service provider.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 overviews the architecture of DARC. Sections 4 and 5 introduce our proposed anomaly detection approach and root cause location approach, respectively. Section 6 describes two additional computational cost reduction tricks. Section 7 presents experimental results on three semi-synthetic datasets. Section 8 presents experimental results on three real-world datasets and two detailed case studies. Finally, Section 9 draws conclusions.

2 Related Work

Our proposed DARC consists of a diffusing anomaly detection system followed by a root cause location system. To our knowledge, identifying diffusing anomalies at the early stages has not been studied before, though there have been plenty of sophisticated algorithms on locating root causes with multi-dimensional attributes.

Adtributor [4] detects anomalies via an auto-regression moving average [5] model based approach, and locates their root causes by using the concepts of explanatory power, succinctness and surprise in advertising revenue debugging. It searches through all dimensions and aims to find the one with the most explanatory power, which can most explain the abnormal value changes. Adtributor searches each dimension individually, because it is rare for multiple dimensions to contribute together to a root cause in advertising revenue system. However, this assumption does not hold in cloud computing systems, where a root cause generally involves multiple attributes, and usually not all instances with the same attribute are abnormal.

R-Adtributor [26] identifies root causes (attribute combinations) by recursively applying Adtributor to each subcubes in the candidate set. However, its termination condition is hard to determine. Additionally, both Adtributor and R-Adtributor identify root causes based on the explanatory power, defined as the percentage change in the overall metric that is explained by the change in each candidate. They cannot be used at the early stage of anomaly diffusion, in which the overall change is imperceptible due to the small number of abnormal instances.

End2End [2] identifies the root cause of performance degradation of cellular service providers. It builds models that capture the normal performance to detect performance degradation and label every instance. Then, End2End searches through all abnormal ones and applies the Apriori Principle in Association Rule Mining [1] to locate the root cause of performance degradation. Unfortunately, it is sensitive to the parameters of the models and those in association rule mining, hence its performance is unstable. Additionally, since attributes in cloud computing systems are collected from assorted modules and vary significantly in their magnitude and variance, training an independent model for each attribute is impracticable, making End2End unsuitable for cloud system anomaly root cause location.

In addition to the algorithms mentioned above, many empirical approaches have also been proposed to locate root causes by searching attribute combination trees. iDice [18] reduces the top-down search complexity by pruning off attribute combinations associated with fewer issue reports and impacting fewer customers, exhibiting small or no changes in issue volumes, or with less isolation power [3] and more redundancy. Then, all remaining combinations are ranked according to their relative significance by Fisher distance [11], and combinations with scores lower than a cutoff threshold are ignored. iDice cannot identify abnormal attribute combinations with insignificant magnitude changes, and rare abnormal instances at the early stage of diffusion. Hence, it cannot address diffusing anomaly root cause location.

HotSpot [28] uses a statistic algorithm [15] to detect anomalies, by calculating the upper and lower thresholds with a periodically updated parameter, and labeling abnormal instances when their values are beyond these thresholds. Then, it searches top-down through all attribute combinations for the root cause, using Monte Carlo Tree Search [6] in combination with Apriori Principle based hierarchical pruning to reduce the search space. A Ripple Effect based potential score is proposed to measure the potential for each attribute combination and the value function in Monte Carlo Tree Search. Due to the limitations of the ripple effect, it holds only for fundamental measures but cannot deal with forecasts with zeros, and the search process may sometimes be time-consuming.

Squeeze [17] improves HotPot by proposing Generalized Ripple Effect and searching first top-down and then bottom-up. Squeeze filters out potential normal attribute combinations, whose deviation between the true and the forecasted values is lower than a knee-point, and clusters the abnormal combinations by the deviation score density. Then, a root cause location algorithm is applied to each cluster to find the most likely attribute combination. Only one root cause is identified in each cluster, because according to Generalized Ripple Effect, anomalies in each cluster are triggered by the same root cause. Squeeze applies a top-down search strategy to locate the root cause, where attribute combinations with the largest generalized potential score are appended to the root cause candidate set until the score is higher than a threshold. The simplest candidate with the largest generalized potential score is finally selected as the root cause.

The basic Ripple Effect theory, which assumes all samples under abnormal attribute combinations are anomalies and derives their forecasted values from the corresponding real values by the same proportion, makes Hotspot and Squeeze only applicable when the anomaly is fully diffused. Hotspot and Squeeze also have difficulty locating root causes with tens of attributes, which is common in modern cloud computing systems.

3 Overview of DARC

Fig. 1 shows the block diagram of our proposed DARC, which consists of four components: data preparation, anomaly detection, root cause location, and remediation.

In Data Preparation, KPIs from multiple components of the cloud system (e.g. CPU, memory, and processes) are periodically collected and delivered to Logstore. Some of these data are also exported to DataStore (big data distributed computing and storage service of the cloud computing service provider) for subsequent offline batch processing. The gathered data are pre-processed by cleaning dirty and duplicate logs, structuring and joining data, and data transformation, such as grouping logs of the same instance.

Then, the processed data are sent to the Anomaly Detection component, where a decision tree is first adopted to analyze the attribute importance. Since the attributes are stable, their importance only needs to be calculated once at the beginning. Next, the 10 most important attributes are selected, and 2nd-stage percentile analysis is used to compute the percentile values of their combinations. Finally, MK scores are computed and compared with a threshed to detect the anomalies.

Based on the Anomaly Detection results, the Root Cause Location system searches for root causes using two strategies. First, since some attribute combinations are more important or more prone to fail, percentile values of these attribute combinations are monitored. When anomalies are observed by experts or the monitoring system, their root cause is searched from a specific node rather than from all abnormal combinations, saving lots of computational cost. Second, DARC periodically searches the entire dataset when no anomalies are observed. Abnormal attribute combinations with high frequency values are selected and searched in a bottom-up fashion with three computational complexity reduction techniques.

4 Anomaly Detection

The cloud computing system is highly dynamic and complex, with thousands of modules and frequent updates. In order to maintain the system stability, these modules, especially those important and prone to fail, are monitored and rolled back immediately when anomalies are detected.

However, attributes collected from different modules can vary significantly in their magnitudes, variance and perturbations, and new attributes keep emerging due to the rapid evolving nature of the cloud system to meet customer demands.

Table 1 shows the statistics of three attributes on the same day. The standard deviation of Attribute A is as high as 3,340

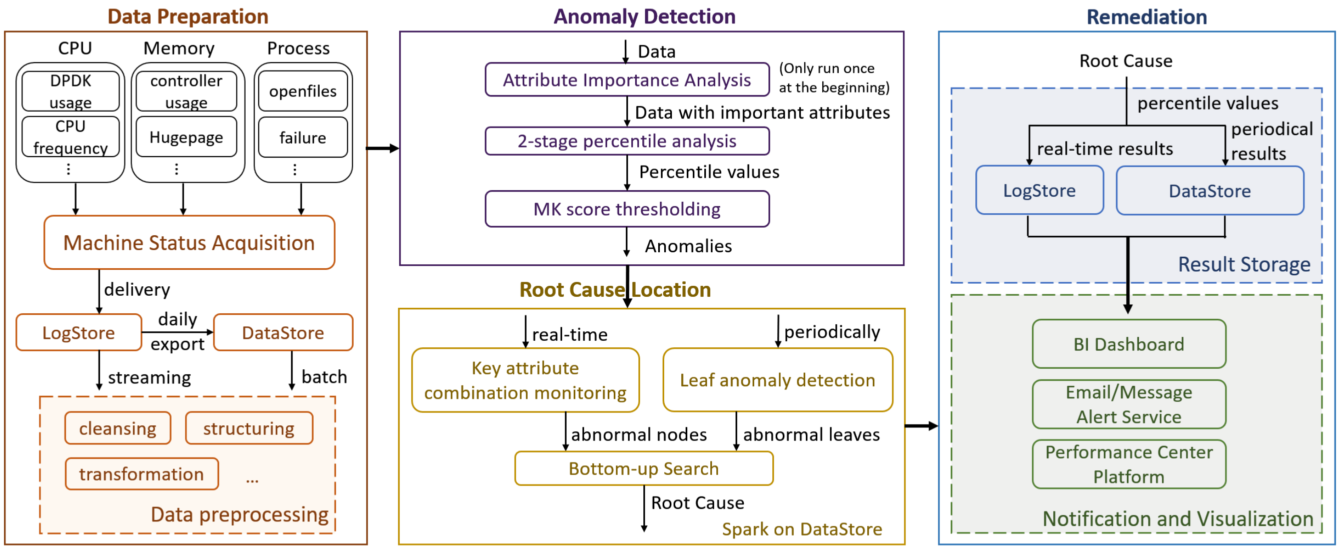


Figure 1. DARC, which includes data preparation, anomaly detection, root cause location, and remediation.

million, and Attribute B has 368 spikes. These large variations are typical in cloud systems. For example, the CPU usage gathered at a specific time might be influenced by customers' applications, control programs, underlying kernel performance, and so on. Attribute values are very unstable in some occasions, and anomalies can hardly be accurately detected from them.

Table 1. Statistics of three different attributes on a specific day.

Attribute	Mean	std	No. spikes	Trend
A	3,339,887,411	196,375,454	63	0.19
B	20.81	10.47	368	0.52
C	880,870,872	84,426,718	2	0.78

Fig. 2 shows the network bandwidth of an instance during a single day. When the instance is running network-related business (roughly between 5:00 and 9:30), the network bandwidth is as high as 1.2 Gb, whereas it is close to 0 when the relevant processes are closed (e.g., between 15:00 and 24:00). These long non-business periods should be excluded from computing the statistics, for they do not represent the true network performance of the instance.

Therefore, a robust anomaly detection algorithm that fits various attributes is desired.

4.1 1-Stage Percentile Analysis

1-stage percentile analysis may be used to reduce disturbances by spikes and non-business periods.

Consider the scenario shown in Fig. 3, which contains an attribute combination with diffusing anomaly, and only very few machines are abnormal. There are n machines. The

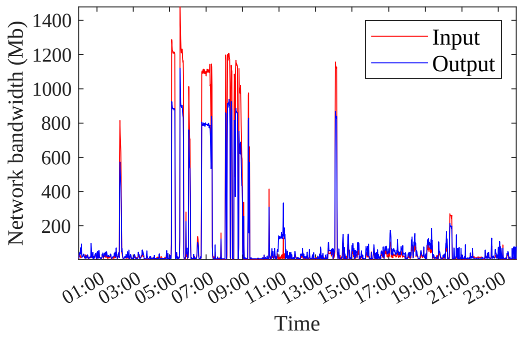


Figure 2. The network bandwidth of an instance within one day.

first a are abnormal, and the last $n-a$ are normal. The ith machine contains m_i attribute values (samples) sorted in descending order. The first s_i samples are spikes, and the last z_i samples are not running any business. Abnormal values are $\{v_{ij} | i \in [1,a],\ j \in [z_i+1, m_i]\}$, and spikes are $\{v_{ij} | i \in [1,n],\ j \in [m_i - s_i + 1, m_i]\}$.

1st-stage evaluation identifies a percentile parameter p, such that only $\{v_{ij} | v_{ij} \le v_p, i \in [1,n], j \in [1,m_i]\}$ are used in computing the statistics, where v_p is the p percentile value of all v_{ij}. p needs to satisfy the following constraint:

$$\frac{\sum_{i=1}^{n} s_i}{\sum_{i=1}^{n} m_i} \le 1-p \le \frac{\sum_{i=1}^{a} (m_i - z_i)}{\sum_{i=1}^{n} m_i}, \tag{1}$$

so that the spikes are excluded from consideration.

A challenge with 1-stage percentile analysis is that the parameter p is very difficult to determine. If $\frac{\sum_{i=1}^{n} s_i}{\sum_{i=1}^{n} m_i} > 1-p$, then p is too large, and some spikes are still included in computing the statistics. If $1-p > \frac{\sum_{i=1}^{a}(m_i - z_i)}{\sum_{i=1}^{n} m_i}$, then p is

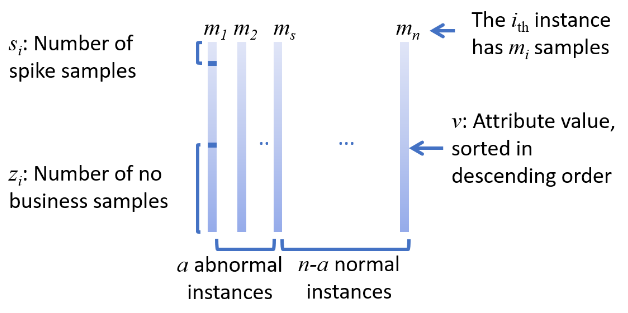

s_i: Number of spike samples
z_i: Number of no business samples
m_1 m_2 m_s m_n
The i_{th} instance has m_i samples
v: Attribute value, sorted in descending order
a abnormal instances
n-a normal instances

Figure 3. Illustration of the scenario in our analysis.

too small, and some non-business samples are included in computing the statistics. Both situations are undesirable.

4.2 2-Stage Percentile Analysis

Our proposed 2-stage percentile analysis, applied to a specific attribute combination, is illustrated in Fig. 4. Temporal data of each instance under this attribute combination are fed into the 1st-stage percentile analysis to assess the instance performance, which is robust to noise, spikes, and business variations. The performance of this attribute combination is further evaluated by the 2nd-stage percentile analysis on all instances. A percentile value is then computed for later anomaly detection.

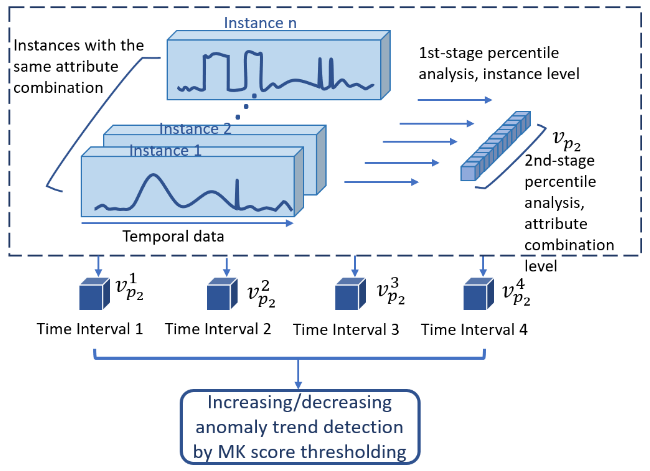

Instances with the same attribute combination
Instance n
Instance 2
Instance 1
Temporal data
1st-stage percentile analysis, instance level
v_{p_2}
2nd-stage percentile analysis, attribute combination level
$v^1_{p_2}$
$v^2_{p_2}$
$v^3_{p_2}$
$v^4_{p_2}$
Time Interval 1
Time Interval 2
Time Interval 3
Time Interval 4
Increasing/decreasing anomaly trend detection by MK score thresholding

Figure 4. The anomaly detection component of DARC.

More specifically, our proposed 2-stage percentile analysis is:

$$\frac{\sum_{i=1}^{n} s_i}{\sum_{i=1}^{n} m_i} \leq 1 - p_1 \leq \frac{\sum_{i=1}^{n}(m_i - z_i)}{\sum_{i=1}^{n} m_i}, \tag{2}$$

$$1 - p_2 \leq \frac{a}{n}, \tag{3}$$

where p_1 is the 1st-stage percentile parameter, and p_2 the 2nd-stage parameter. We apply 1st-stage percentile analysis to each instance to filter out the spikes, concatenate the remaining attribute performance values from all instances, and then apply 2nd-stage percentile analysis to them to identify the percentile value v_{p_2} for diffusing anomaly detection.

Since $s_i \ll m_i - z_i$ and $\frac{a}{n}$ is usually greater than 0.1 in practice, p_1 and p_2 are easy to determine.

4.3 Diffusing Anomaly Detection

After v_{p_2} is computed for each attribute combination, the next step is to determine whether the combination contains anomaly. At the early stages of deployment, performance degradation caused by anomalies can only be observed on a few instances. The number of abnormal instances keeps rising until the deployment is complete. Although it is relatively easy to detect anomalies after the full deployment, it is desirable to detect diffusing anomalies at early stages to avoid catastrophic failures.

v_{p_2} is relatively stable in cloud computing systems, for each combination contains thousands of instances whose businesses are mostly consistent, and a few changes have little impact on v_{p_2}. When this attribute combination becomes abnormal and the anomaly starts spreading, the number of abnormal instances gradually increases, and v_{p_2} rises steadily. This abnormal statistic significantly differs form the normal ones, making it easy to be detected.

The diffusing anomaly detection problem can be transformed into a trend analysis task. As shown in Fig. 4, DARC calculates each attribute combination's performance in four time intervals and applies an enhanced trend analysis approach based on the MK test [14, 21] to detect whether anomalies are diffusing in this attribute combination. MK hypothesis test aims at finding the trend of data by the increasing/decreasing relationship between each pair of samples.

We propose to test data trends by calculating normalized slopes with monotonicity restrictions. We first compute

$$S = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \text{sign}(v^j_{p_2} - v^i_{p_2}), \tag{4}$$

where k is the number of time intervals in Fig. 4, and then compute the monotonicity and slope by

$$\text{mono} = \max\left(S + 1 - \frac{k(k-1)}{2}, 0\right), \tag{5}$$

$$\text{slope} = \text{median}\left(\frac{v^i_{p_2} - v^j_{p_2}}{(i-j) \times \max(\min(v_{p_2}), 1)}\right), \forall i > j. \tag{6}$$

mono constrains the monotonically increasing trend, since the number of abnormal instances keeps increasing, and hence the percentile values cannot decrease. The maximum of mono is 1, since the maximum of S is $k(k-1)/2$. slope calculates the increasing slope of data with a normalizer $\max(\min(v_{p_2}), 1)$ to reduce the influence of the attribute performance magnitude.

MK is then obtained by:

$$MK = \text{mono} \times \text{slope}. \tag{7}$$

Usually different attribute blue performance values result in different MK, which helps reduce the computational complexity, as will be further explained in Section 5.2.

Note that MK can also be used to test monotonically decreasing trend, with a simple sign change.

Finally, MK is compared with a user-defined threshold θ to identify the anomalies.

5 Root Cause Location

This section introduces our proposed root cause location approach.

5.1 Bottom-up Search Strategy

Given the high-dimensional attributes and the large number of values for each attribute, searching through all attribute combinations for root causes is computationally prohibitive. Some approaches address this issue by building an attribute combination tree, which provides parent-child relationship that can guide the search directions and expedite the search process.

Most of these techniques locate root causes by top-down search, which is not applicable to the anomaly diffusing scenario due to the unique causal relationship between parent nodes and child nodes. Since our approach uses percentile analysis to detect anomalies, the parent percentile values of the root cause may look normal when the number of instances in the root cause node is much smaller than that of other child nodes, or when the instances under the root cause node have much smaller magnitude. Consider a parent attribute combination with a root cause node and a normal node, whose values are in $[0, 10]$ and $[50, 100]$, respectively. Even though the performance of the root cause node keeps degrading, it has little impact on the percentile values of the parent node, i.e., a normal parent node does not necessarily mean all its child nodes are normal. Thus, all child nodes and all attribute combinations have to be checked.

In summary, the unique causal relationship of percentile nodes renders only bottom-up search feasible.

5.2 MK Score Guided Search Strategy

We propose an MK score guided search strategy that focuses only on the most abnormal parents and ignores values with relatively lower scores. Since MK scores are unique and there is little chance that different nodes give the same score, redundant parent nodes with the same score are discarded. This frequently occurs in cloud computing systems since some attribute values are highly correlated. In addition, nodes that have been visited are marked and excluded from future searching, which further reduces the computational cost.

Fig. 5 illustrates the MK score guided bottom-up search strategy. Blue nodes represent normal attribute combinations, red ones are abnormal, and gray ones are unvisited nodes. The location algorithm starts from the abnormal combination "ABCD", whose normal parent nodes "ABD" and "BCD", as well as "ACD", an abnormal parent node with a lower MK score, are discarded. Only "ABC", the node with the highest MK score, needs to be searched.

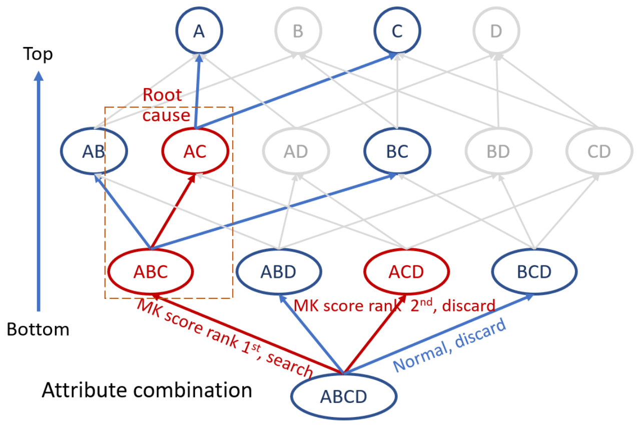


Figure 5. The bottom-up search strategy. Blue nodes and lines represent normal attribute combinations and search paths. Red ones are abnormal. Gray nodes are discarded.

Fig. 6 illustrates the detailed steps of searching from "ABC" and locating the root cause "AC". The search starts from "ABC" and goes through all its parents. Parent nodes "AB" and "BC" are normal (Steps 1, 2 and 5, 6), but "AC" needs to be searched (Steps 3, 4 and 7). The same approach is applied to "AC", and DARC detects its both parents, "A" (Steps 8 and 9) and "C" (Steps 10 and 11) are normal. Then, the node "AC" is marked as the root cause (Step 12), and the searching from the combination "ABCD" is finished.

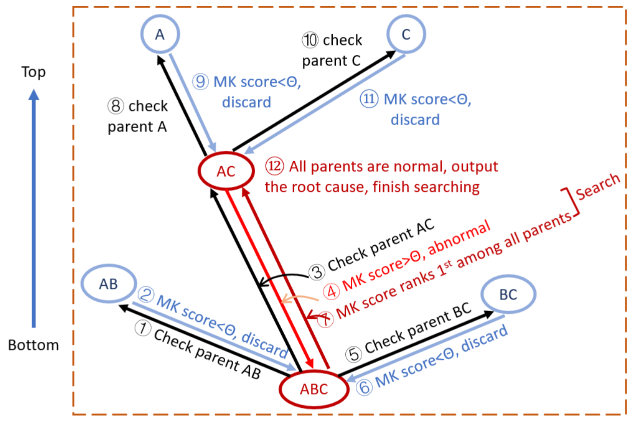


Figure 6. Step-by-step illustration of the bottom-up search strategy.

5.3 Root Causes Selection

A parent node of the root cause attribute combination may also look abnormal when other child nodes under this parent have small attribute values or fewer instances. In this case, the parent node may be mistakenly marked as the root cause. Therefore, a double-check is applied, which goes through all child nodes of each potential root cause node and detects anomalies. If more than one children act abnormal, then the root cause is correctly located. Otherwise, the combination is parent/grandparent of the root cause, and it is also double-checked until the true root cause is located.

Finally, all root causes are sorted according to their simplicity and impact. Those that influence more users deserve more attention.

6 Computational Complexity Reduction

The computational cost of bottom-up search is prohibitive. For example, the number of leaves of a tree built from d attributes, each with v different values, is as high as v^d. In cloud computing systems, d is over 30, and each attribute may contain dozens of different values.

In addition to MK score guided search introduced in the previous section, we propose two more tricks to further reduce the computational cost.

As shown in Fig. 7, attribute importance analysis is used to select the 10 most important attributes. Their combinations are checked for anomalies, where normal ones (blue cubes) are truncated and abnormal ones (light red cubes) are kept. Then, value frequency analysis is applied to the abnormal combinations, and only those with high frequency values (red cubes) are searched.

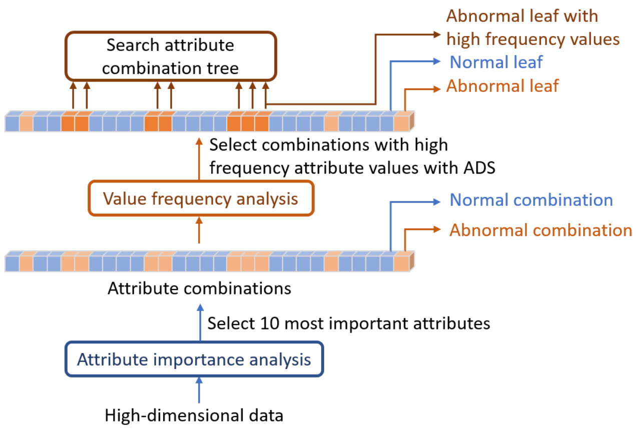

Abnormal leaf with high frequency values
Search attribute combination tree
Normal leaf
Abnormal leaf
Select combinations with high frequency attribute values with ADS
Value frequency analysis
Normal combination
Abnormal combination
Attribute combinations
Select 10 most important attributes
Attribute importance analysis
High-dimensional data

Figure 7. Techniques for reducing the computational complexity in Root Cause Location.

6.1 Attribute Selection

Due to the complex ecosystem of large scale cloud computing systems, there are a large number of attributes collected from machine hardware models, resource distribution settings, cluster attributes, etc. Some of them are more relevant to the system performance than others. The unimportant ones contribute little to the root cause location accuracy, but introduce noise and increase the search cost. Although some of them may be removed using domain knowledge, it is labor intensive, and there is no guarantee that all of them can be removed. We therefore propose an attribute importance analysis approach to automatically discard the less relevant ones.

Many machine learning algorithms can be used to estimate the attribute importance. Some approaches, e.g., Linear Regression [22] and Logistic Regression [12], are very efficient, but their linear nature may be insufficient in handling high-dimensional data. Boosting algorithms, e.g., GBDT [9], XGBoost [7], and lightGBM [13], are more accurate in assessing attribute importance, but their computational cost is very high on big data.

To balance effectiveness and efficiency, we use Decision Tree to evaluate the attribute importance. It takes the attributes as inputs, and their anomaly status (represented by the increasing trend in this paper) as labels. Attributes with low importance values are discarded. Since the attribute importance is stable, it only needs to be calculated once at the beginning.

6.2 Abnormal and Root-cause-related Combination Selection by Value Frequency Analysis

Most attribute combinations are normal, since performance degradation typically happens rarely and locally in cloud systems. These normal combinations are discarded, since child nodes of root causes are generally abnormal. This truncates most search branches.

Additionally, not all abnormal combinations are descendants of root causes. For instance, the business of Internet companies are sensitive to the number of users. Their resource cost may gradually increase from 5 PM to 10 PM due to the rising number of users, and the corresponding attribute combinations may be incorrectly labeled as abnormal. Searching through them not only introduces false positives, but also increases the computational cost. In practice, these miss-labeled combinations barely have strong correlations and their attribute values are inconsistent, whereas combinations affected by the root cause generally have many common attribute values. Besides, root cause related combinations significantly outnumber the miss-labeled ones. Therefore, only attribute values with high frequency need to be considered, and others are discarded to improve the location accuracy and reduce the computational complexity.

We propose a Derivation Split (DS) approach to automatically choose a proper threshold to isolate the abnormal values. For each attribute, DS first sorts all values according to their frequencies in descending order, and calculates the derivative of the data series. Then, we split at the value with the largest derivative, because values before or at it happen frequently, whereas those after it rarely happen.

Some attributes have dramatically imbalanced number of values in cloud systems. For example, the main product in the public cloud is YYY, whereas the rest account for only 22%. After DS, only the most popular value is kept, and all other possible root cause values are abandoned by mistake.

To automatically choose a proper split adaptive to both balanced and imbalanced data, Adaptive Derivation Split (ADS) is further proposed to enhance DS. It first calculates the standard deviation of the original frequencies, and also the standard deviation of the logarithmic frequencies. If the former is smaller, then it splits with the largest derivative of the original frequencies; otherwise, it splits with the largest derivative of the logarithmic frequencies.

Finally, attribute combinations whose value frequencies are lower than the selected threshold are discarded to reduce the computational cost.

7 Experimental Results on Semi-synthetic Datasets

This section presents experimental results on three semi-synthetic datasets to demonstrate the accuracy and efficiency of DARC.

7.1 Datasets

We used three datasets to validate the performance of DARC. They were semi-synthetic, because of the insufficient number of anomaly diffusing cases with root causes in cloud computing systems: manually recognizing diffusing anomalies in the high dimensional large-scale data of cloud systems and locating their root causes are highly time-consuming and challenging even for experienced support engineers. Therefore, in addition to real data collected from different components of a cloud computing service provider's computing system, synthetic data generated with Gaussian random values were also included. Anomalies with various impact degrees were injected into each dataset, and Gaussian random noise was introduced to imitate business variations.

A summary of the three datasets is shown in Table 2. They all have 10 attributes, whose number of values varied from 2 to 54. D1 is the largest dataset, with more than 34 million samples. Attributes in D1, D2 and D3 have different magnitudes, since data were collected from different components of cloud systems. "Anomaly type" indicates whether an increasing or decreasing trend leads to anomaly. An example of the former is network latency increase, and the latter is free memory decrease.

Anomalies with different impact degrees and diffusing speeds were injected into each dataset blue by mimicking online anomalies, where abnormal software is deployed to all machines by the Canary Deployment Strategy. For instance, an anomaly causing performance degradation is injected to 0%, 10%, 30%, 60% and 100% machines in 5 successive days. As shown in Table 3, D1 contains three anomalies, the root causes of which are combinations of respectively 4, 3 and 5 attributes, and it costs respectively 5, 4 and 3 time intervals to diffuse to all machines.

For each diffusing process, such as Anom1 diffusing in the first time interval, values of randomly selected instances under the root cause attribute combination were increased, imitating anomaly-caused CPU/memory cost increase. Gaussian noise was then added to all values.

7.2 Anomaly Detection

DARC detects anomalies via 2nd-stage percentile analysis, followed by the MK trend score thresholding. First, experiments were carried out on D1 to investigate the detection efficiency, when 1,000 attribute combinations were considered. On average, it took less than 1 second per attribute combination.

Then, we studied the robustness of DARC to its parameters, i.e., the 1st and 2nd stage percentile values. Results of different p_1 and p_2 on the same anomaly-spreading attribute combination within 27 time intervals are shown in Fig. 8. The 1st-stage parameter, p_1, was set between 50% and 99.9%. The 2nd one, p_2, was between 90% and 99.9%.

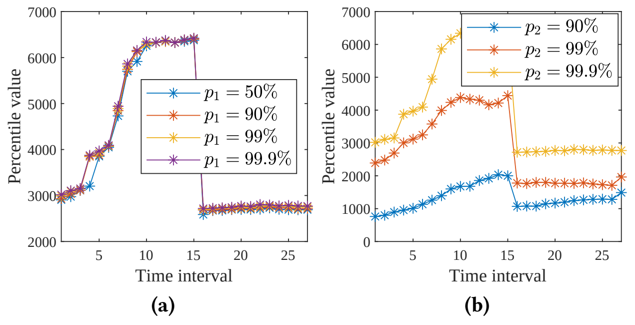


Figure 8. (a) Percentile values with $p_2 = 99.9\%$ and different p_1; and, (b) percentile values with $p_1 = 99.9\%$ and different p_2.

Fig. 8 shows that DARC is robust to p_1, since the client's business is relatively steady in a short time interval, but more sensitive to p_2. All parameter values within the aforementioned range were able to assess the performance of the attribute combinations. Finally, the abnormal rising trend caused by the increasing number of abnormal instances with increasing CPU/memory consumption can be detected by the enhanced MK algorithm.

7.3 Root Cause Location Accuracy

To show the effectiveness of DARC, it was applied to each of the three time intervals on the three datasets. For the nine

Table 2. Characteristics of the three semi-synthetic datasets. "Max distinct value number" means the maximum number of distinct values of all attributes.

Name	Dimensionality	Max distinct value number	Size	Value range	Measure	Anomaly type
D1	10	54	34,000,000	(0, 3,171)	Fundamental	Increasing
D2	10	5	5,200,000	(-29.05, 56.7)	Derived	Decreasing
D3	10	5	1,500,000	(0, 684,958)	Fundamental	Increasing

Table 3. The three injected anomalies in D1, and their root cause attributes.

Anomaly	Root cause attributes	Dimensionality	Diffusing intervals
Anom1	A, B, C, D	4	5
Anom2	A, B, C	3	4
Anom3	B, C, D, E, G	5	3

Table 4. Accuracy of DARC on the three semi-synthetic datasets.

Dataset	Time interval	Anomaly	Rank	No. false positives
D1	1	Anom1	1	0
	2	Anom1	2	0
		Anom2	1	
	3	Anom1	2	2
		Anom2	1	
		Anom3	3	
D2	1	Anom1	1	0
	2	Anom1	2	5
		Anom2	1	
	3	Anom1	2	5
		Anom2	1	
		Anom3	3	
D3	1	Anom1	1	1
	2	Anom1	2	2
		Anom2	1	
	3	Anom1	2	2
		Anom2	1	
		Anom3	3	

set of experiments, after the root cause searching process was finished, the root causes were ranked based on their impact degrees and number of abnormal child nodes.

Table 4 shows the results. For the three tested time intervals of each dataset, the injected root causes were successfully located at the top. Anom1, Anom2 and Anom3 on all three datasets were ranked with the same order, which were 2nd, 1st and 3rd, respectively. This is because the ranking algorithm emphasizes the impact degree of the root cause most. The number of attributes of Anom1, Anom2 and Anom3 was 4, 3 and 5, respectively. Anom2 had the smallest number of attributes, covering the most machines, and hence ranked the first.

We also tried to apply the recently proposed Squeeze algorithm to the three datasets. However, the Generalized Ripple Effect theory that Squeeze is based on does not agree with the anomaly diffusing situation, since not all instances under the root cause are abnormal. Therefore, we generated an additional data copy for each dataset, where anomalies were diffused to all instances under the root causes. The normal ones were labeled as "real", and anomaly diffused ones as "predict". When applying Squeeze to this data copy, it suffered from huge memory consumption when calculating the cumulative distribution function of leaves' deviations and clustering based on the deviation score density. As a result, it was not able to finish even on the smallest dataset D3.

It is important to mention that DARC can not only search the entire system to locate root causes, but also start searching from an arbitrary abnormal attribute combination. This may be needed when engineers want to have a closer look at some attributes for they are more important, or more prone to fail. In these cases, percentile values of the specific attribute combinations are monitored automatically. Once an anomaly is detected on one attribute combination, DARC is called to search from this node, saving lots of computational cost. Experiments were also conducted to show its efficiency, where abnormal attribute combination (*a2, b3, c1, d2, e1, f2, g1, h3, i3, j4*) in D1 was set as the starting point. The search took about one minute, and the root cause (*a2, b3, c1, d2*) was successfully located without false positives. This is very useful and convenient in practical cloud computing system maintenance.

7.4 Root Cause Location Efficiency

DARC employs three techniques to reduce the computational complexity while searching for root causes in a bottom-up manner.

First, since the computational cost increases exponentially with the attribute dimensionality, decision tree is used to analyze the attribute importance and filter out less relevant attributes. We used 30 attributes on machine memory consumption of the cloud system as input, and their anomaly state as label, to train a decision tree. Attribute importance analysis was used to select the 10 most anomaly-related ones. As the data were highly imbalanced, the normal class was downsampled, so that 2,262 abnormal samples and 19,156 normal ones were used. It took about 98 minutes to select the 10 most important attributes, which were consistent with the experience of performance maintainers. Since the attributes are relatively stable and will not change for months, this step only needs to be performed periodically.

Second, DARC filters out combinations with low frequency attribute values, since such combinations are rarely abnormal. This significantly reduces the computational complexity. The number of all possible attribute combinations, abnormal ones, and the selected high frequency abnormal ones of different time intervals on D3 are shown in Table 5. There were 28,970 attribute combinations, among which only 289 were abnormal in Time Interval 3. The 58 high frequency abnormal combinations were analyzed, and others with less frequent attribute values were discarded.

Table 5. Numbers of leaves, abnormal ones, and the selected ones in different time intervals of D3.

Time interval	No. combinations	No. abnormal combinations	No. selected combinations
1	28,970	91	34
2	28,970	400	87
3	28,970	289	58

For instance, the frequency of Attribute F among all abnormal combinations is shown in Table 6. The two most common values, $f2$ and $f4$, are caused by two root causes, (*B*='*b2*', *C*='*c2*', *F*='*f2*', *G*='*g3*') and (*C*='*c1*', *E*='*e2*', *F*='*f4*', *I*='*i3*', *J*='*j2*'). Another root cause is (*B*='*b3*', *E*='*e1*', *G*='*g1*', *I*='*i2*'), which is irrelevant to F. In this case, only $f2$ and $f4$ are essential, and other values may lead to falsely located root causes.

Table 6. Values of attribute combination F, their frequency and other statistics when using ADS.

Values of F	freq	$\log_{10}$freq	Split score	Keep
$f2$	80	1.903	–	✓
$f4$	71	1.851	**0.228**	✓
$f1$	42	1.623	0.055	×
$f3$	37	1.568	0.063	×
$f5$	32	1.505	-	×
std$_N$	0.403	**0.397**	-	

We use ADS to automatically select the useful attribute values. ADS ignores the most frequent value "$f2$", because at least one value must exist, even though F is irrelevant to the root cause. This avoids removing all combinations. Then, a normalized standard deviation score (std$_N$) is calculated based on the frequency data and the logarithmic frequency data, individually. The logarithmic ones are chosen to calculate the split score, for they have larger std$_N$. Finally, the split with the largest split score is chosen, i.e., $f2$ and $f4$ in this case.

Third, DARC also applies an MK score guided search strategy, where only the parent attribute combination node with the largest score is searched. This again significantly fastens the search process. The algorithm complexity when searching all abnormal parent nodes is $O(n!)$, and reduces to $O(n)$ in our search strategy, since only one parent node needs to be searched for each node.

The time consumption of DARC detecting anomalies and locating root causes with different computational complexity reduction techniques is shown in Table 7. Experiments without attribute selection were not performed, due to the prohibitive computational cost.

Table 7. Time consumption of DARC with different computational cost reduction techniques. ✓means the corresponding technique is adopted, and × means not.

Attribute selection	Value frequency selection	MK score guided search	Time (min)
✓	✓	✓	33
✓	✓	×	70
✓	×	✓	68

8 Real-World Experimental Results

This section presents real-world experimental results to further demonstrate the accuracy and efficiency of DARC.

8.1 Overall Evaluation

DARC was also applied to three real-world datasets from a cloud computing service provider on CPU, network bandwidth, and network sessions, respectively, to further evaluate its performance. Each dataset included about 1.6 billion time series data of about 275 thousand virtual machines in two months, as summarized in Table 8. Diffusing anomalies were labelled by experts, referring to the records of gray release.

Table 8. Summary of the three real-world datasets.

Dataset	No. Data	No. Machines	No. Anomalies
CPU	1,597,968,311	274,812	90
Bandwidth	1,608,454,839	274,812	70
Sessions	1,602,350,748	274,812	14

DARC and two recently proposed well-performing algorithms (Squeeze and Hotspot) were evaluated on the three datasets. The results are shown in Table 9. DARC almost always significantly outperformed the two counterparts on precision, recall and F_1 score, on all three datasets. This is because DARC specifically targets at detecting diffusing anomalies, whereas Squeeze and Hotspot consider fully established anomalies.

Two specific case studies are presented below to further demonstrate the performance of DARC in details.

Table 9. Performances on three real-world datasets.

Dataset	Approach	Precision	Recall	F_1 Score
CPU	Squeeze	0.73	0.12	0.21
	HotSpot	0.33	0.01	0.02
	DARC	**0.86**	**0.74**	**0.80**
Bandwidth	Squeeze	0.15	0.06	0.08
	HotSpot	1.00	0.03	0.06
	DARC	**0.88**	**0.86**	**0.87**
Sessions	Squeeze	0.07	0.07	0.07
	HotSpot	0.00	0.00	0.00
	DARC	**0.83**	**0.71**	**0.77**

8.2 Case Study 1

On 6/24/2020, the performance maintainers of the cloud computing service provider noticed an unusual phenomenon, where the percentile value of machine memory consumption with attributes (*cluster_id='AY302T', sla_category= 'exclusive', instance_family= 'ecs.g6e', template_name='ecs_moc 2.0_template', controller= '0,6,8', product='MoC2_0', standard _model= 'F65C3', cgroup='tdc'*) kept rising for more than seven days, as shown in Fig. 9(a).

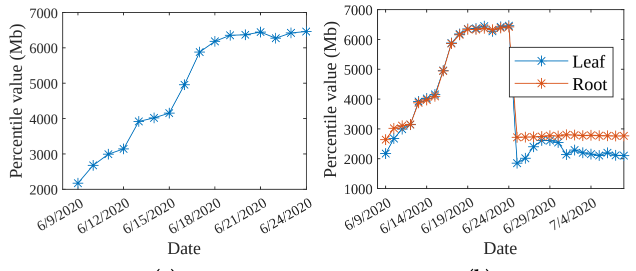


Figure 9. Abnormal percentile values in Case 1. (a) Percentile values between 6/9/2020 and 6/24/2020 of an abnormal node discovered by performance maintainers; (b) Percentile values between 6/9/2020 and 7/8/2020 of the discovered abnormal node and the located root cause node. The bug was fixed on 6/24/2020.

This was quite abnormal, due to the high stability of the cloud service and the steadiness of the machine percentile value during the last month. It took the maintainers two hours to manually search the root causes in the huge attribute space. Finally, nine attributes were identified. Their names and numbers of distinct attribute values were: cluster_id (2250), instance_family (267), template_name (121), vm_num (235), sla_category (7), controller (237), product (8), standard_model (169), cgroup (81). It was confirmed by cloud service operators that the software version in use may lead to heavy memory consumption on some specific machines. After the bug was fixed on 6/24/2020, percentile values of machine memory consumption dropped back to normal and stayed at a low and stable level, as shown in Fig. 9(b).

Although the root cause was eventually located and fixed, this bug had undermined the machine performance and customer experience since 6/10/2020 for 15 days. Automatically monitoring and locating root causes at the early diffusing stages is crucial.

To test DARC in this case, we applied it to 4-day memory consumption data starting from 6/14/2020, each of which contained roughly 14,809 million samples. Two root cause location modes were tested.

First, DARC was tested on this case without prior knowledge. There were 3,000,000 leaf attribute combinations, including 525 abnormal ones. Only 24 leaves with high frequency values were searched. The entire searching process took only 18 minutes, and the manually located root cause (*sla_category= 'exclusive', product='MoC2_0', cgroup='tdc'*) ranked 1st at the end.

Second, since an abnormal attribute combination was observed by the maintainers, DARC started at this node and searched upwards. It took only about 1 minute, and the true root cause again ranked 1st.

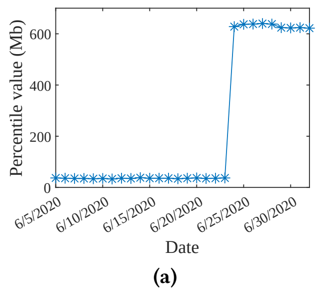
8.3 Case Study 2

On 6/23/2020, the percentile value of machine memory consumption with attributes (*cluster='AY240V', sla_category= 'exclusive', instance_family='ecs.c5, ecs.g5, ecs.r5', template= 'ec_defaul_template', product='normal', standard_model='F52', vm_num='9', cgroup='tdc'*) showed a sudden rise, as shown in Fig. 10(a). The searching space was similar to Case 1, and the root cause was located manually base on the previous experience, i.e., (*sla_category, product, cgroup*). However, after fixing the located root cause (*cluster='AY240V', sla_category= 'exclusive', product='normal', cgroup='tdc'*), the abnormal percentile value remained at a high level in the following a few days.

In order to solve the problem, DARC attempted to locate root causes automatically in two modes based on data in June 23-26, when the anomaly just started diffusing. It took 16 minutes to search the entire system, and about one minute from the specific abnormal node. Both search modes indicated that the node (*cluster_id='AY240V', template_name= 'ecs_default_template', cgroup='tdc'*) was the root cause, whose percentile values were shown in Fig. 10(b). This result was confirmed by the performance maintainers.

9 Conclusions

It is crucial for large cloud computing systems to be able to detect and locate diffusing anomalies at early stages, before they start to affect too many machines and clients. Most existing approaches largely hinge on the assumption that all instances under the root cause are abnormal. This paper has



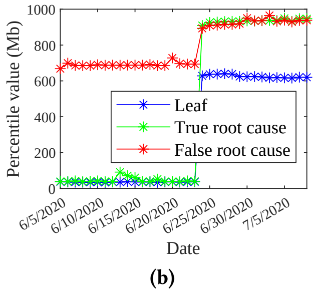


Figure 10. Abnormal percentile values of Case 2. (a) Percentile values between 6/5/2020 and 7/2/2020 of an abnormal leaf discovered by performance maintainers; (b) Percentile values between 6/5/2020 and 7/8/2020 of the discovered abnormal leaf node, root cause node located by DARC, and the falsely located root cause node. The falsely located bug was processed on 7/2/2020 and the anomaly still existed.

proposed DARC to detect and locate these diffusing anomalies in a high-dimensional large-scale search space. It uses first 2-stage percentile analysis plus MK score thresholding to detect the anomalies, and then a bottom-up search approach combined with a series of computational complexity reduction techniques to efficiently locate the root causes. Extensive experiments on semi-synthetic and real-world datasets showed that DARC can accurately and efficiently locate the root causes. DARC is now being used in system of a cloud computing service provider for automatic anomaly detection and root cause location.

References

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*. 207–216. https://dl.acm.org/doi/pdf/10.1145/170035.170072.

[2] Faraz Ahmed, Jeffrey Erman, Zihui Ge, Alex X Liu, Jia Wang, and He Yan. 2017. Detecting and localizing End-to-End performance degradation for cellular data services based on TCP loss ratio and round trip time. *IEEE/ACM Transactions on Networking* 25, 6 (2017), 3709–3722. https://ieeexplore.ieee.org/abstract/document/8077761.

[3] Christoph Arndt. 2001. *Information Measures: Information and Its Description in Science and Engineering*. Springer Science and Business Media.

[4] Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah. 2014. Adtributor: Revenue debugging in advertising systems. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*. 43–55. https://dl.acm.org/doi/10.5555/2616448.2616454.

[5] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time Series Analysis: Forecasting and Control*. John Wiley and Sons.

[6] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (2012), 1–43. https://ieeexplore.ieee.org/abstract/document/6145622/.

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 758–794. https://dl.acm.org/doi/pdf/10.1145/2939672.2939785.

[8] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of ACM Symposium on Operating Systems Principles*. 156–167. https://dl.acm.org/doi/pdf/10.1145/3132747.3132772.

[9] Jerome H Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29, 5 (2001), 1189–1232. https://www.jstor.org/stable/2699986.

[10] Haryadi S Gunawi, Mingzhe Hao, Riza O Suminto, Agung Laksono, Anang D Satria, Jeffry Adityatama, and Kurnia J Eliazar. 2016. Why does the cloud stop computing? Lessons from hundreds of service outages. In *Proceedings of ACM Symposium on Cloud Computing*. 1–16. https://dl.acm.org/doi/pdf/10.1145/2987550.2987583.

[11] Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data Mining: Concepts and Techniques*. Elsevier.

[12] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied Logistic Regression*. Vol. 398. John Wiley and Sons.

[13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of Advances in Neural Information Processing Systems*. https://dl.acm.org/doi/10.5555/3294996.3295074.

[14] Maurice Kendall and Jean D. Gibbons. 1990. *Rank Correlation Methods*. A Charles Griffin Title.

[15] Suk-Bok Lee, Dan Pei, MohammadTaghi Hajiaghayi, Ioannis Pefkianakis, Songwu Lu, He Yan, Zihui Ge, Jennifer Yates, and Mario Kosseifi. 2012. Threshold compression for 3G scalable monitoring. In *Proceedings of IEEE International Conference on Computer Communications*. 1350–1358. https://ieeexplore.ieee.org/abstract/document/6195498/.

[16] Ze Li, Qian Cheng, Ken Hsieh, Yingnong Dang, Peng Huang, Pankaj Singh, Xinsheng Yang, Qingwei Lin, Youjiang Wu, Sebastien Levy, and Murali Chintalapati. 2020. Gandalf: An intelligent, End-to-End analytics service for safe deployment in large-scale cloud infrastructure. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*. 389–402. https://www.usenix.org/system/files/nsdi20-paper-li.pdf.

[17] Zeyan Li, Chengyang Luo, Yiwei Zhao, Yongqian Sun, Kaixin Sui, Xiping Wang, Dapeng Liu, Xing Jin, Qi Wang, and Dan Pei. 2019. Generic and robust localization of multi-dimensional root causes. In *Proceedings of IEEE International Symposium on Software Reliability Engineering*. 47–57. https://ieeexplore.ieee.org/abstract/document/8987454/.

[18] Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, and Dongmei Zhang. 2016. iDice: Problem identification for emerging issues. In *Proceedings of International Conference on Software Engineering*. 214–224. https://dl.acm.org/doi/pdf/10.1145/2884781.2884795.

[19] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, et al. 2018. Meltdown: Reading kernel memory from user space. In *Proceedings of USENIX Security Symposium*. 46–56. https://dl.acm.org/doi/pdf/10.1145/3357033.

[20] Ashraf Mahgoub, Alexander Michaelson Medoff, Rakesh Kumar, Subrata Mitra, Ana Klimovic, Somali Chaterji, and Saurabh Bagchi. 2020. OPTIMUSCLOUD: Heterogeneous configuration optimization for distributed databases in the cloud. In *Proceedings of USENIX Annual Technical Conference*. 189–203. https://www.usenix.org/system/files/atc20-mahgoub.pdf.

[21] Henry B Mann. 1945. Nonparametric tests against trend. *Econometrica: Journal of the Econometric Society* 13, 3 (1945), 245–259. https://doi.org/10.2307/1907187.

[22] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. 2012. *Introduction to Linear Regression Analysis*. Vol. 821. John Wiley and Sons.

[23] Amruta More, Sheetal Vij, and Debajyoti Mukhopadhyay. 2014. Agent based negotiation using cloud–An approach in E-commerce. In *Proceedings of Annual Convention of Computer Society of India*. 489–496. https://arxiv.org/abs/1311.6233.

[24] Jennifer Ortiz, Brendan Lee, Magdalena Balazinska, Johannes Gehrke, and Joseph L Hellerstein. 2018. SLAOrchestrator: Reducing the cost of performance SLAs for cloud data analytics. In *Proceedings of USENIX Annual Technical Conference*. 547–560. https://www.usenix.org/conference/atc18/presentation/ortiz.

[25] Jian Pei. 2020. Data pricing–From economics to data science. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 3553–3554. https://dl.acm.org/doi/pdf/10.1145/3394486.3406473.

[26] Linnea Rudenius and Moa Persson. 2018. *Anomaly detection and fault localization: An automated process for advertising systems*. Master's thesis.

[27] T. Savor, M. Douglas, M. Gentili, L. Williams, K. Beck, and M. Stumm. 2016. Continuous deployment at Facebook and OANDA. In *Proceedings of IEEE/ACM International Conference on Software Engineering Companion*. 21–30. https://ieeexplore.ieee.org/abstract/document/7883285/.

[28] Yongqian Sun, Youjian Zhao, Ya Su, Dapeng Liu, Xiaohui Nie, Yuan Meng, Cheng Shiwen, Dan Pei, Shenglin Zhang, Xianping Qu, and Xuanyou Guo. 2018. HotSpot: Anomaly localization for additive KPIs with multi-dimensional attributes. *IEEE Access* 6, 1 (2018), 10909–10923. https://ieeexplore.ieee.org/abstract/document/8288614/.

[29] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. 2009. Detecting large-scale system problems by mining console logs. In *Proceedings of ACM Symposium on Operating Systems Principles*. 117–132. https://dl.acm.org/doi/pdf/10.1145/1629575.1629587.

[30] Yong Xu, Kaixin Sui, Randolph Yao, Hongyu Zhang, Qingwei Lin, Yingnong Dang, Peng Li, Keceng Jiang, Wenchi Zhang, Jian-Guang Lou, et al. 2018. Improving service availability of cloud systems by predicting disk error. In *Proceedings of USENIX Annual Technical Conference*. 481–494. https://www.usenix.org/conference/atc18/presentation/xu-yong.