DriftInsight: Detecting Anomalous Behaviors in Large-scale Cloud Platform

Fan Jing Meng[1], Xiao Zhang[1,2], Pengfei Chen[1], Jingming Xu[1]
[1]IBM Research – China, Beijing, P.R. China
{mengfj, zhxiaobj, pfchen, xujingm}@cn.ibm.com
[2]Xi'an Jiaotong University
zx.0319@stu.xjtu.edu.cn

Abstract*—Detecting anomalous behaviors of cloud platforms is one of critical tasks for cloud providers. Every anomalous behavior potentially causes incidents, especially some unaware and/or unknown issues, which severely harm their SLA (Service Level Agreement). Existing solutions generally monitor cloud platform at different layers and then detect anomalies based on rules or learning algorithms on monitoring metrics. However, complexity of nowadays cloud platforms, high dynamics of cloud workloads and thousands of various types of metrics make anomalous behavior detection more challenging to be applied in production, especially in large scale cloud production environments. In this paper, we present a practical cloud anomalous behavior detection system called DriftInsight. It firstly analyzes multi-denominational metrics of each component and identifies a set of representative steady components based on the convergences of their states. Then it generates a state model and a state transit model for each steady cloud component. Finally, it detects behavior anomalies of these steady components in near real-time and meanwhile evolve behavior models on the fly. The evaluation results of this approach in a commercial large-scale PaaS (Platform-as-a-Service) cloud are demonstrated its capability and efficiency.

Keywords-cloud computing; behavior model, anomaly detection; unsupervised clustering

I. INTRODUCTION

Cloud has become the backbone of modern IT systems. According to Gartner, the worldwide public cloud services market is projected to grow 18% in 2017 to total $246.8 billion and reach $383.4 billion in 2020 with a five-year CAGR of 16.5% [1]. Cloud providers compete not only in differentiated cloud services but also in their SLA fulfillment to sustain and extend their business. However, incidents and unplanned outages caused by malfunctioned cloud components usually lead to cloud SLA violation and therefore revenue reductions and customers' dissatisfaction. For example, AWS suffered a five-hour outage due to errors on read and write operations for the Amazon DynamoDB service in the US-East Region on September 20, 2015[1]. Google reported a two hour and 40 minutes' service outage due to failed virtual networking software in February. 2015[3]. Azure management services were down for about 8 hears on February 29, 2014[4].

To prevent unplanned outages and minimize negative impacts of incidents, cloud providers monitor cloud platform with thousands of or even hundreds of thousands of metrics and alert anomalies on these metrics based on thresholds, pre-defined rules, or learned models. However, complexity of nowadays cloud platforms, high dynamics of cloud workloads and thousands of various types of metrics make anomalous behavior detection more challenging to be applied in production, especially in large-scale cloud production environments.

Generally, majority of existing approaches needs pre-knowledge on systems in terms of understanding semantics of metrics, logs or other collected data. For example, CloudSeer[5] collects logs to induce normal workflows of OpenStack.[6] adopts Principle Components Analysis (PCA) to reduce data dimension and then correlates the most relevant principle components with specific anomalies. Although some approaches are pre-knowledge free, these approaches are more applicable to stable systems. For example, Ubl [7] adopts SOM to train normal models of a system and reports abnormal behaviors that deviate from normal patterns. However, workloads running in cloud are quite dynamic and hard to define a stable rules or models for anomaly detection. As a result, a practical abnormal behavior detection for large-scale cloud production environment with no or less dependencies on pre-knowledge is highly demanded.

In this paper, we proposed a practical behavior anomaly detection approach for large-scale clouds in production. The approach not only dramatically reduces analysis scale and improve detection efficiency, but also removes domain knowledge dependencies. The main contributions of this approach are three-fold:

- We propose convergence-based steady component discovery approach to dramatically downsize the problem scale;
- We use unsupervised clustering algorithms to transform multi-dimensional metrics into single state metric for abnormal state and state transit detection;
- We implement the system and deploy it into a large-scale commercial cloud. The capacity and efficiency of the approach have been demonstrated with real production data.

The rest of the paper is organized as follows. In Section 2, we briefly review the existing work. Section 3 provides the main challenges and the approach overview. Section 4 describes the detailed design of our approach. Section 5 presents the system implementation and evaluates the

2159-6190/17 $31.00 © 2017 IEEE
DOI 10.1109/CLOUD.2017.37

system in a large-scale commercial cloud. Section 6 concludes the paper and discusses future work.

II. Related Work

Anomalous behavior detection of distributed systems is an essential and challenging topic which attracts many research attentions. The existing approaches can be categorized into three types according to the data source: *metric-based approaches*, *log-based approaches*, *trace-based approaches.*

A. Metric-based Approach

Metric based approaches [1][8][9][10][11] usually apply statistics and machine learning algorithms on collected metrics and detect anomalies on these metrics. These metrics include application performance metrics (e.g. response time, throughput), application monitoring metrics (e.g. thread pool size, heap size, active transaction count) and system metrics (e.g. CPU, memory, disk). In PCA based approach [6], the authors collect 518 metrics and apply PCA to generate multiple PCs for specific anomalies. [12] collects video streaming related metrics to infer whether current video system was suffering from performance anomaly. Huge volume and diversity of metrics require more efforts to define or learn knowledge to detect anomalies. Besides, effective knowledge evolvement mechanism is also required to accommodate dynamic environment of cloud.

B. Log-based Approach

Logs are generated along applications' running and reflect application processing status and execution logics. However, these timestamped logs have many important and self-defined content buried in unstructured textual log messages which are hard to be processed and understood by machines. It always requires pre-knowledge to extract patterns or templates from logs [13]. [14] extracts feature from logs, and then uses PCA to discover the invariances between the features. In CloudSeer[5], the logs of OpenStack are collected and based on those, the normal workflow is mined and used to evaluate current behavior. When the invariances are violated, the anomaly is detected. Pre-knowledge dependency is a key obstacle to apply this type of approaches in abnormal behavior detection of large scale complicated systems.

C. Trace-based Approach

Traces record information about a program's execution, which is typically used by experienced programmers or operators for program debugging or diagnosis purposes. In previous works [15][16][17], system call traces or method invocation [18] traces are collected. Then statistics and machine learning methods are adopted to detect performance anomalies. PerfCompass [17], collects the traces of specific application's system calls and calculates normal ranges of execution time of system calls. It detects anomalies when current system call's execution time beyond the standard criteria. This type of approaches need instrument systems and bring great overheads, which makes them hard to be applied in real-time or near real-time anomaly detection of production systems.

III. Challenges and Approach Overview

In this section, we describe the main challenges to detect abnormal behaviors of large-scale cloud platform and then present an overview to our approach.

A. Challenges

Detecting abnormal behaviors of a large-scale cloud is a critical but challenging problem. The major technical challenges include following three reasons.

1) ***Complexity and diversity of cloud platform.*** Cloud provides standardized visualization of IT resources at different layers and generally offers services as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Every cloud has cloud core components and cloud management components. For example, OpenStack [19] has over 50 types of components including compute (NOVA), networking (Neutron), object storage (Swift), block storage (Cinder), identity (Keystone), image (Glance), dashboard (Horizon), orchestration (Heat), telemetry (Ceilometer) and etc. Cloud Foundry[20] has key components including Router, User Account and Authentication (UAA), UAA Database (UAADB), Cloud Controller (CC), CC Database (CCDB), Droplet Execution Agent (DEA), Diego Cell, Diego Bulletin Board System (BBS), Health Manager (HM9000), Loggreator, BOSH, and etc. Each component presents different behaviors in nature and is impacted by other components as well as requests. Although high availability is usually designed and enabled, failures on components or component clusters at any layer potentially cause incidents, which are hardly detected or diagnosed due to complexity of each component and their dependencies.

2) ***Unpredicted and dynamic cloud workloads.*** Cloud brings agilities to users and meanwhile makes its behavioral detection and prediction much more difficult. Dynamic deployment and diverse of cloud services or applications dramatically impact the behavior of cloud platform. For example, DEAs of Cloud Foundry perform tasks of staging Cloud Foundry applications, running droplets and managing Warden containers. Its behaviors highly depend on the number and characteristics of workloads running in the Warden containers. Similarly, behaviors of Nova node of OpenStack are determined by the number and behaviors of Virtual Machines (VMs) on the machine.

3) ***Large-scale deployment in real-world cloud production environment.*** Cloud providers e.g. Amazon, IBM, Microsoft, Google, Alibaba tend to achieve efficiency of operations and cost with large scale deployment. In result, dramatically increased cloud scale challenge storage and processing capacity of existing cloud operations systems. Massive metrics and logs are generated and requires

sufficient processing and analysis in anomaly detection and problem diagnosis.

Besides the technical challenges, the non-technical challenges are also the barriers to detect abnormal behaviors. Enterprise IT governance regulations usually constrain the monitoring approaches (e.g. installing software agents or running scripts) on highly sensitive systems. Inaccessibility to critical technical staff or lack of knowledge on misbehaving components makes anomaly detection and/or diagnosis less ineffective.

B. Approach Overview

In this paper, we present DriftInsight, a practical approach to address the above technical challenges in anomalous behavior detection for large-scale cloud. The approach overview is shown in Figure 1. This approach includes three major phases: *warm-up* phase, *abnormal behavior detection* phase, and *model evolving* phase.

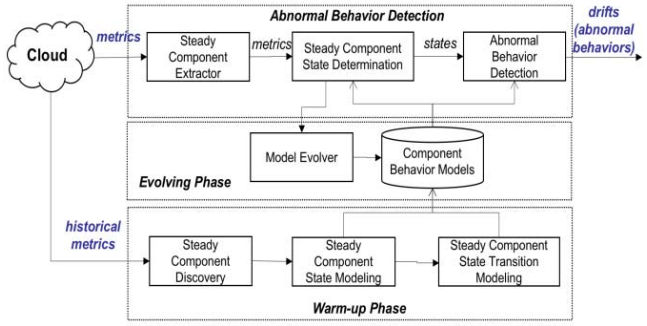

Figure 1. DriftInsight Approach Overview.

Warm-up phase consumes historical monitoring metrics of all the cloud components in last several days and generates steady component behavior models for further anomaly detection. Three modules are designed in this phase. Steady Component Discovery module aims to identify a set of steady components which are relatively stable and indicate the health of the Cloud by analyzing convergences their behaviors. Steady Component State Modeling consumes multi-dimensional metrics as input and applies clustering techniques to identify their typical states of their behaviors. The state serves as the compressed single dimension to detect abnormal behaviors of given components.

Abnormal Behavior Detection phase aims to analyze the real-time/near real-time metrics and apply behavior models to detect abnormal states or abnormal state transits of these steady components. It includes three modules. Steady Component Extractor filters out the multi-dimensional metrics of steady components and feeds them into following detection steps. Steady Component State Determination consumes these metrics and checks current states of steady components based on their behavior models. Then Abnormal Behavior Detection modules checks the current behaviors as well as behavior transits against the behavior models and alerts detected anomalies if any.

Model Evolving phase is designed to make these behavior models always up-to-date by fading out out-of-date data points and adding latest metrics. Model Evolver takes steady component states as input and fuse them into existing behavior model. Based on convergence characteristics of each steady model, it updates steady component state model and state transit model accordingly.

Our approach has following advantages compared to the existing techniques. 1) It **significantly downsizes the problem scale** with compressed one-dimensional behavior detection on carefully selected steady components instead of multiple-dimensional metrics on all the cloud components. This design overcomes the impact of dynamic and fluctuated cloud workloads and large-scale deployment of the cloud, which dramatically improve the efficiency and make it possible to detect the anomaly in near real-time. 2) It applies an **unsupervised learning based approach** which avoids the issues that complexity and diversity of cloud components brings. Moreover, this approach does not rely on pre-knowledge on anomalies and can effectively find unknown issues, which is a big benefit to cloud SREs.

IV. Detailed Design

In this section, we describe the detailed design of the key modules of DriftInsight.

A. Multi-dimensional Metrics Reduction

Dimensionality reduction is one of key technical challenges in system anomaly detection of distributed systems. Higher dimensions bring computation complexity and make traditional approaches hardly to be applied in production. In this paper, we adopt an unsupervised clustering algorithm DBSCAN [21] to aggregate multi-dimensional metrics into a one-dimension component state metric while keep semantics of these metrics for problem diagnosis.

We firstly normalize multi-dimensional metrics data with z-score algorithm to remove different scales. Figure 2. shows the normalization and clustering result of the 3-dimension data, including CPU usage, memory usage and disk usage.

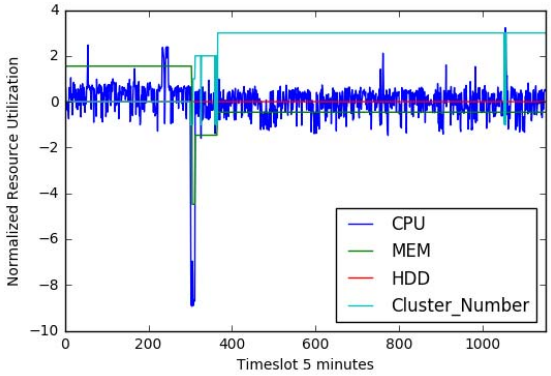

Figure 2. Normalized metrics and cluster state

Figure 3. gives a clustering result of a UAA component in a PaaS cloud. It has seven clusters a.k.a. component states which are generated based on 13,164 metrics data. Therefore, multiple dimensional metrics are represented by a single "State" metric which dramatically reduces computation complexity.

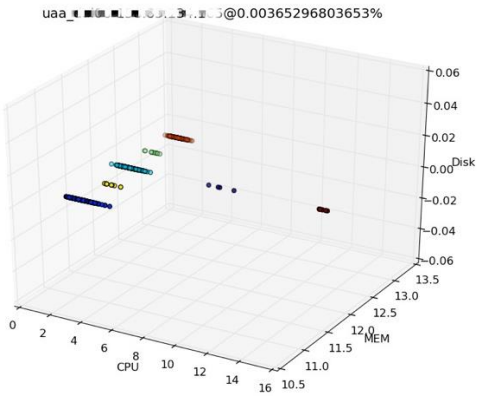

Figure 3. Component State Cluster Example

B. Steady Component Identification

Some cloud component behaviors are very difficult or even impossible to be modeled due to dynamics of cloud workloads. For example, DEAs of a Cloud Foundry cloud keep changing due to applications may deploy or destroy dynamically. Given this fact, we need select a subset of steady cloud components who behaviors are relatively stable and can be modelled for anomaly detection.

We design an algorithm to discover steady components from massive deployed cloud components and then use them to build behavior models. To measure the stability of a cloud component, we define the "*convergence*" to measure how fast a cloud component can have stable state behavior model generated. The detail algorithm procedure is described as follows.

Steady Component Discovery

Input:

Component Set: $\boldsymbol{C}$
Metrics Set: M
Value Set: V
Steady Threshold Timeslot: T

Output:

Steady Components Set: $\boldsymbol{SC}$

Variables:

$\boldsymbol{S}$: Component State Cluster
$\boldsymbol{MCN}$: Maximum cluster number
$\boldsymbol{MCT}$: Minimum convergence time

Procedure:

1. Initialize $SC \leftarrow \emptyset, MCN \leftarrow 0, MCT \leftarrow 0$
2. **FOR** each c_i in $\boldsymbol{C}$
3. **FOR each** $\boldsymbol{m_j}$ **in** $\boldsymbol{M}$
4. $nv_j^i = Normalize(v_j^i)$// normalize values of metric j for component i
5. **END FOR**
6. $S^i = Cluster(nv^i)$//cluster normalized metrics of component i into state cluster
7. MCN = Number (S^i) -1// get the maximum cluster number for component i
8. MCT^i =GetTimeSlot (MCN) // get the minimum time slot for the maximum cluster of component i
9. **IF** MCT^i <T// the component convergence within given threshold
10. $SC \leftarrow c^i$// add this component into steady component set
11. **END FOR**

First, we normalize each dimension with z-score algorithm. Second, we cluster these normalized multi-dimensional metrics into a one-dimensional state for each cloud component. Third, we identify the minimum time slot of the last cluster is generated, which is the time the state model of this cloud component converged. It means that this is no new cluster presented even more data points are added. Finally, we compare the convergence time slot of this component with the convergence threshold of steady components. If the convergence time slot is smaller than threshold time slots, the cloud component is a steady component. Otherwise, this component is not a steady component.

C. Behavior Modeling

We model component behaviors with two types of models: *state behavior model* and *state transit model*. *State behavior model* defines the typical working states with similar metrics presentations. We apply the unsupervised clustering algorithm DBSCAN to learn normal state models. *State transit model* defines statistical characteristics of the given components. In given detection window e.g. 10 time slots, the state transits should follow normal distributions. Significant deviation (e.g. > 3 standard deviations) from normal distribution can be considered as anomalies.

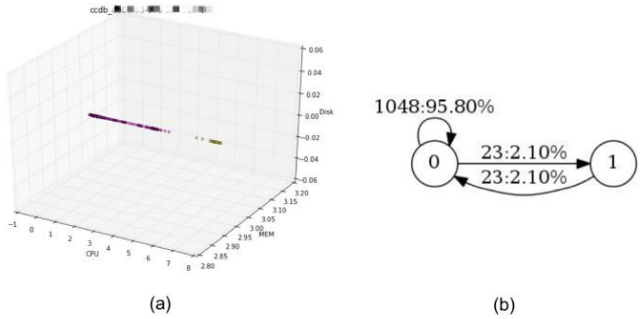

Figure 4. State Behavior Model and State Transit Model

Figure 4. shows the state behavior model and state transit model of a CCDB in a PaaS cloud. This CCDB has two typical working states. The state #0 (pink cluster) with 1071 data

points is the major working state of this component. The state #1 (yellow cluster) with 23 data points is secondary working state of this components. This CCDB has 95.80% probabilities in state #0 and both 2.1% probabilities to transit from state #0 to state #1 and from state #1 to state #0

D. Model Evolving

Behaviors of cloud components are keep changing due to evolving cloud component as well as workloads dynamics in this cloud. The behavior models should be updated to adopt the evolvement and dynamic workloads. To achieve that, we design a Model Evolver to update these state behavior models and state transit behavior models on the fly.

Near real-time steady component state determination results flow into the Model Evolver module. Based on the convergence of a given component, model evolver catch a mini-batch of new data points e.g. 10% time slots of convergence time slots of this component. Once *Model Evolver* gets the mini-batch data, it replaces the same scale of oldest data points with the same size of new data points. Then we run the behavior model generation again and update the state behavior models and state transit behavior models. This design ensures that our DriftInsight always uses the latest models for anomaly detection.

V. EVALUATION

We implemented a system based on above approach and continuously analyzed cloud component monitoring metrics from a large-scale commercial PaaS platform. In this section, we introduce the system implementation and our analysis results based on near real-time analysis to illustrate the capacity and efficiency of our approach works to detect anomalous behaviors in a large-scale cloud platform.

A. System Implementation

Our system is implemented with Python 2.7 [22] and deployed into open-source based near real-time big data processing platform. Its architecture is shown in Figure 5. The near real-time metrics data is flowed and cached into Apache Kafka™ [23]. The near real-time anomalous behavior detector is implemented as a Spark Streaming [24] application and running on a Spark cluster. Raw metrics data and analyzed component state data are stored into InfluxDB [25]. The cloud anomalous behaviors are alerted by Kapacitor [26]. The detailed component drift data and component behavior models are stored in MongoDB.[27] These metrics data, component state data, and component anomalies are then visualized by Grafana [28] and a Django [29] web application.

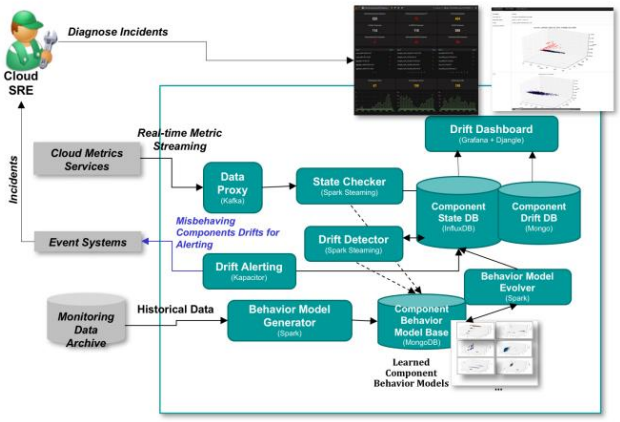

Figure 5. System Implementation Architecture

B. Data Set

We deployed our DriftInsight system into a large-scale commercial PaaS cloud which is built on Cloud Foundry. The cloud is composed of hundreds of Cloud Foundry components and cloud management components (e.g. login, catalog, Business Support System (BSS)) and running on thousands of nodes. Over 100 types of agents are deployed on these nodes and are monitoring millions of metrics from the cloud.

In this paper, we took 12 types of general performance metrics on CPU, memory, disk, and network for each component as input. TABLE I. shows these metrics types, collectors and collection frequency. We were continuously subscribing these metrics data into anomalous behavior detector. We set the analysis window as 5 minutes to balance compute cost with detection efficiency. In each detection window, we processed over 30,000 metrics and check their behaviors against behavior models.

TABLE I. METRICS TYPES

Category	*Metrics*	*Collector*	*Frequency*
CPU	System CPU Percent	BOSH agent	3 minutes
CPU	User CPU Percent	BOSH agent	3 minutes
CPU	Total CPU Percent	BOSH agent	3 minutes
Memory	Memory Percent	BOSH agent	3 minutes
Disk	System Disk Percent	BOSH agent	3 minutes
Disk	Persist Disk Percent	BOSH agent	3 minutes
Disk	Swap Disk Percent	BOSH agent	3 minutes
Disk	Ephemeral Disk Usage Percent	BOSH agent	3 minutes
Network	Receiving Traffic	agent	5 minutes
Network	Transmit Traffic	agent	5 minutes
Network	All Traffic	agent	5 minutes

C. Evaluation Results

We took 3-day monitoring data without critical incidents as training data set. Firstly, we analyzed component behaviors and identified steady components of the cloud platform based on their convergences. Figure 6. shows convergences of the analyzed components. From the analysis results, 334 components out of 820 were converged within first 488 timeslots. These components include 218 DEA nodes and 116 Cloud Foundry core components. 50% of core components and 37% of overall DEA nodes are selected as steady components for further anomaly detection.

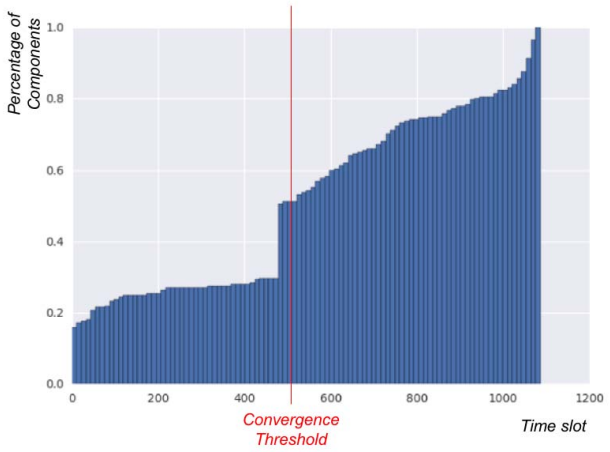

Figure 6. Steady Cloud Component Identification.

For each steady component, we generated behavior models which include a state model and a state transit model. Figure 7. and Figure 8. show examples of the state models and state transit models of these steady components. These models were stored in MongoDB.

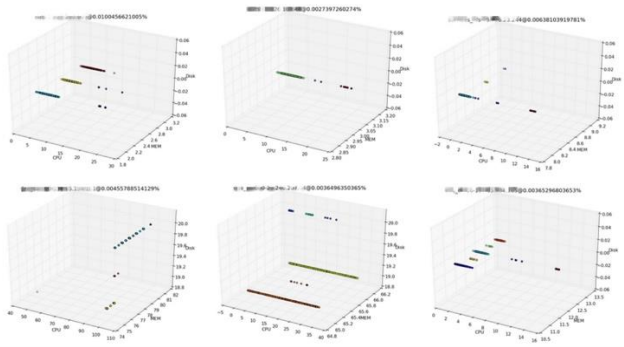

Figure 7. Steady Component State Model Examples

Then DriftInsight checked states of these steady models based on receiving multi-dimensional metrics based on these state behavior models in near real-time. Once states of these state components were determined, Anomaly Detector will check probabilities of state transits in the given windows and reported significant deviations (e.g. > 3 standard deviations) as anomalies.

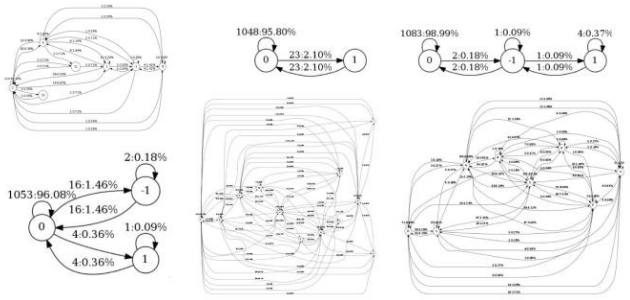

Figure 8. Steady Component State Transit Model Examples

Figure 9. gives the evaluation result how well DriftInsight detected critical incidents caused by abnormal behaviors of cloud components based on learned behavior models. Among the 10 selected critical incidents, DriftInsights detected anomalous behaviors around 115 minutes earlier than incidents reported by previous threshold-based monitoring and alerting system or customers' reports. This earlier anomalous component detection capability gives Cloud SREs more time to verify, diagnose, and fix these problems and therefore improves the SLA.

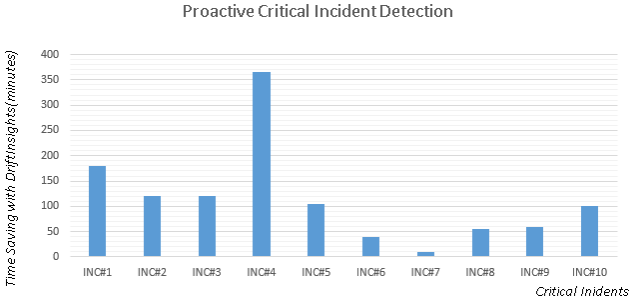

Figure 9. Anomaly Detection Result Evaluation

D. Case Study

We selected incident #2 caused by an abnormal behaving router to demonstrate how cloud SREs use our DriftInsight to accelerate their problem diagnosis and incident resolution. Figure 10. (a) shows the latest behavior models of given router before the incident happened. From its state behavior model in Figure 10. (b), there are three typical states: state #0 in blue, state #1: in light brown, and state #-1 in grey.

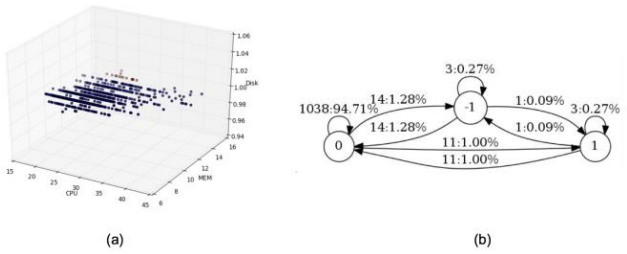

Figure 10. The Router Behavior Models

Figure 10. shows that this router should stay in the state #0 with 94.71% probabilities, and in state #1 and state #-1 with both 0.27% probabilities. It has 1.28% possibilities to transit from state #0 to state #-1 and transit from state #-1 to state #0. It has 0.09% possibilities to transit from state #1 to state #-1 and transit from state #-1 to state #1. It has 1.00% possibilities to transit from state #0 to state #1 and transit from state #1 to state #0.

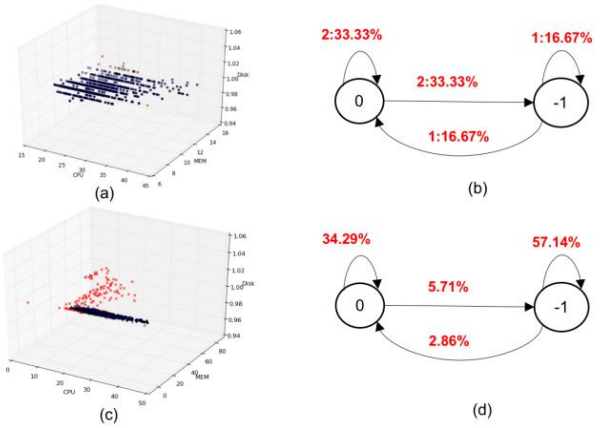

Figure 11. Abnomalous Router Behaviors Detected

Figure 11. shows anomalous behaviors of the router which was detected by its behavior models shows in Figure 10. Figure 11. and (b) shows its states and state transits when the anomaly was firstly reported due to its abnormal state and state transition are heavily deviated from its normal distributions. Figure 11. (c) and (d) demonstrates the cumulative states and state transits when previous rule-based alerting system reported this incident. We reported the anomaly 1 hour and 55 minutes earlier than that was reported by the previous rule-based system.

VI. Conclusion and Futurework

In this paper, we proposed a practical behavior anomaly detection approach for large-scale clouds in production. The approach not only dramatically reduces analysis scale and improve detection efficiency, but also removes domain knowledge dependencies for each metrics. Through the evaluation in large-scale commercial cloud, the capability and efficiency of our approach have been demonstrated.

As future work, we will extend component-based behavior models including state models and state transit models into cross-component behavior models of whole cloud system by considering the correlation of these components. Moreover, the root causes of detected abnormal components can be derived and recommended based on causal relationships among these components. Besides, incorporating of these anomalies with other operations data analysis e.g. logs will provide more contexts of problem diagnosis and accelerate the problem mitigation and resolution.

References

[1] Ng, Fred, et at. “Forcast: Public Cloud Services, Worldwide, 2014-202, 4Q16 Update.” Gartner Inc. Gartner Report: G00320866, Jan 2017

[2] Smolaks, Max. “AWS suffers a five-hour outage in the US”. [Online]. Available: http://www.datacenterdynamics.com/colo-cloud-/aws-suffers-a-five-hour-outage-in-the-us/94841.fullarticle

[3] Babcock, Charles. “Google Cloud Outage: Virtual Networking Breakdown”. [Online]. Available: http://www.informationweek.com/cloud/infrastructure-as-a-service/google-cloud-outage-virtual-networking-breakdown/d/d-id/1319178

[4] Kanaracus, Chris. “Microsoft's Azure Cloud Suffers Serious Outage”. [Online]. Available: http://www.pcworld.com/article/250983/microsofts_azure_cloud_suffers_serious_outage.html

[5] Yu, Xiao, et al. "Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs." ACM SIGOPS Operating Systems Review 50.2 (2016): 489-502.

[6] Guan, Qiang, and Song Fu. "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures." Reliable Distributed Systems (SRDS), 2013 IEEE 32nd International Symposium on. IEEE, 2013.

[7] Dean, Daniel Joseph, Hiep Nguyen, and Xiaohui Gu. "Ubl: unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems." Proceedings of the 9th international conference on Autonomic computing. ACM, 2012.

[8] Gu, Xiaohui, and Haixun Wang. "Online anomaly prediction for robust cluster systems." Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on. IEEE, 2009.

[9] Tan, Yongmin, et al. "Prepare: Predictive performance anomaly prevention for virtualized cloud systems." Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on. IEEE, 2012.

[10] Dean, Daniel J., et al. "Automatic Server Hang Bug Diagnosis: Feasible Reality or Pipe Dream?." Autonomic Computing (ICAC), 2015 IEEE International Conference on. IEEE, 2015.

[11] Chen, Pengfei, et al. "CauseInfer: automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems." INFOCOM, 2014 Proceedings IEEE. IEEE, 2014.

[12] Krishnan, S. Shunmuga, and Ramesh K. Sitaraman. "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs." IEEE/ACM Transactions on Networking 21.6 (2013): 2001-2014.

[13] Fu, Qiang, et al. "Execution anomaly detection in distributed systems through unstructured log analysis." 2009 ninth IEEE international conference on data mining. IEEE, 2009.

[14] Xu, Wei, et al. "Detecting large-scale system problems by mining console logs." Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. ACM, 2009.

[15] Zhang, Hui, et al. "CLUE: System trace analytics for cloud service performance diagnosis." Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, 2014

[16] Dean, Daniel J., et al. "PerfScope: Practical Online Server Performance Bug Inference in Production Cloud Computing Infrastructures." Proceedings of the ACM Symposium on Cloud Computing. ACM, 2014.

[17] Dean, Daniel J., et al. "PerfCompass: toward runtime performance anomaly fault localization for infrastructure-as-a-service clouds." Proceedings of the 6th USENIX conference on Hot Topics in Cloud EComputing. USENIX Association, 2014.

[18] Fonseca, Rodrigo, et al. "X-trace: A pervasive network tracing framework." Proceedings of the 4th USENIX conference on Networked systems design & implementation. USENIX Association, 2007.

[19] OpenStack. [Online]. Available: https://www.openstack.org/

[20] Cloud Foundry. [Online]. Available: http://www.cloudfoundry.org/

[21] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in *Proc. of KDD,* vol. 96, no. 34, 1996, pp. 226- 231.

[22] Python. [Online]. Available: https://www.python.org

[23] Apache Kafka. [Online]. Available: https://kafka.apache.org

[24] Spark Streaming. [Online]. Available: https://spark.apache.org/streaming

[25] InfluxDB. [Online]. Available: https://www.influxdata.com

[26] Kapacitor. [Online]. Available: https://www.influxdata.com/products/open-source/#kapacitor

[27] MongoDB. [Online]. Available: https://www.mongodb.com

[28] Grafana. [Online]. Available: https://grafana.net/

[29] Django. [Online]. Available: https://www.**django**project.com