DoLen: User-side multi-cloud application monitoring

Do Le Quoc, Lenar Yazdanov, Christof Fetzer
Systems Engineering Group
TU Dresden, Germany
Email: first_name.last_name@tu-dresden.de

Abstract—Cloud computing is a popular platform offering computation, storage and communication resources as a service. Monitoring the performance and behavior of multi-cloud applications in a scenario where applications are deployed on multiple cloud providers is a challenge for cloud computing research community. In this paper, we propose a framework that allows monitoring in near real-time resource utilization including CPU, memory, disk activity and network traffic of applications deployed in multiple clouds. We conducted various experiments using a cluster to simulate three clouds, showing the framework's ability to analyze resource consumption of Hadoop applications deployed in multiple clouds. We also performed VM-to-VM attacks to evaluate anomaly detection capacity of the proposed framework.

Keywords—multi-cloud application, cloud monitoring, DoS

I. INTRODUCTION

Cloud computing has rapidly become a popular platform for delivering services and applications over the Internet. In this paradigm, computation, storage and communication resources are offered as a service to customers. Nowadays, design and deployment enterprise applications on multiple clouds have received much interest in both academic research and industry because of locality of access, optimized costs and disaster tolerance. Monitoring multi-cloud applications is necessary to understand the behavior of the applications for improving performance, making decisions about resource allocation, avoiding overload conditions, checking security, verifying service-level agreement (SLA), etc. However, monitoring the performance and behavior of multi-cloud applications in a scenario where applications are deployed on multiple clouds is a challenge because of the inherent complexity e.g. different technologies and services and different interfaces between cloud providers. Moreover, from the viewpoint of cloud customers, monitoring multi-cloud applications is even more difficult because they have limited information about the infrastructure of public clouds that they can use to measure real performance and bottlenecks of their applications. In fact, they are able to access only the virtual resources, whereas the providers manage cloud infrastructure including hardware and hypervisor software [7].

To convince customers, cloud providers provide several software tools that allow customers to monitor and manage of virtual resources assigned to them. One example is Amazon CloudWatch\(^1\) which allows the customers to supervise the performance metrics of their applications and use these metrics to make management decisions, e.g. applying policies for load balancing or scaling. However, using these tools has several disadvantages: the sampling interval is constrained by providers, the techniques to get the monitoring metrics values are not clear to the customers and the observation processes could introduce non-negligible overheads [8] that the customers cannot measure. Therefore, the customers need their own monitoring framework to continuously measure and analyze the behavior of multi-cloud based applications in terms of performance, reliability, effective resource utilization, SLA fulfillment and security.

Last but not least, as cloud customers maintain their sensitive data and critical applications in a multi-cloud environment, security becomes one of their major concerns due to sharing of computation infrastructure. To reassure customers, cloud providers adopt the same techniques and security tools that are used in traditional data centers [13]. However, the cloud infrastructure has many different features, e.g. its large scale and the wide distribution of computation and communication resources, and it is heterogeneous and completely virtualized. Traditional security mechanisms such as authentication and authorization are not enough for the cloud computing environment [9]. As a result, the customers need to have their own mechanism to protect their data and their applications in cloud environments.

In this work, we define the requirements for monitoring applications deployed in multiple clouds from customer perspective. After that we introduce a fine-grained monitoring framework that helps customers to manage, profile and keep track of their applications in clouds. One of the main ideas in our framework is that the resource usage data is collected as the applications are executed, then the data is stored in distributed in-memory key/value stores to provide real-time data processing, correlation and alerting. We validate our framework by simulating a scenario in which Virtual Machines (VMs) running Hadoop applications are Denial-of-Service (DoS) attacked by compromised VMs on the same tenant network. The framework shows the ability to detect the attacks in near real-time.

The paper is organized as follows. A novel architecture for monitoring applications deployed in multiple clouds is presented in Section 2. The evaluation of the framework monitoring Hadoop application in multiple clouds is described in Section 3. Several related works are discussed in Section

\(^1\)http://aws.amazon.com/cloudwatch/
4. Finally, the conclusion and future work are presented in Section 5.

II. MULTI-CLOUD BASED APPLICATIONS MONITORING ARCHITECTURE

We identify important requirements for a user-side multi-cloud application monitoring system as follows:

First, the system should be programmable to periodically measure both the behavior of application components and performance of the VMs the components are located on. The monitoring is done not only at system level but also at application level. Second, the system should have timely notification functionality because the cloud environment where applications are deployed can be very dynamic. The notification functionality should regularly report the status of the environment and the applications with certain a time interval, so that cloud users can have some adjustments to their system, or their system can adapt automatically with the changing of the environment. Last but not least, the measured results of the distributed monitors should be aggregated into a monitoring data store for providing both durability and accessibility so that it can be archived and queried by analysis applications. Our proposed architecture is illustrated in Fig. 1. The architecture consists of four main components.

![Fig. 1: Multi-cloud application monitoring architecture](image)

1. **Probes**: a lightweight program is installed on each VM to measure information about the behavior of hosted applications components such as resource utilization (CPU, RAM and network traffic) and actual performance of each VM. To reduce the number of collected messages, network traffic in each VM is aggregated into 5-tuple flows (e.g. with the same source and destination IP address, source and destination port, and protocol), i.e. Netflow records; and flow records generated by user’s applications are identified using port number. This provides a fine-grain monitoring of the network traffic behavior of applications that enable rapid response to anomalies, whereas providers tools provide only aggregated traffic information. Probes are configured with a measuring period, which is the period between two measures. The value of the measuring period parameter is identified as a trade off between the resolution of monitoring information and the introduced overhead. The monitored information is completely transparent to the cloud provider. The main benefits of using this user-side monitoring system instead of provider-side tools (e.g. CloudWatch by Amazon) are the ability to adjust flexibly for specific purposes of customers and the sampling interval.

2. **Internal Manager**: In a large-scale deployment, lots of monitoring messages exchanged between clouds cause high network overhead. To mitigate this overhead, the collected information from probe programs installed in VMs is emitted to an internal data store in each cloud. The monitoring information is aggregated before being pushed to a centralized repository to provide a global view of the status and behavior of applications deployed on cloud to a central manager component. Moreover, each internal manager is configured with a communication period that is the period to communicate with the centralized manager. If the communication between the internal manager and the centralized manager fails, the monitoring information is stored locally, and it will be sent in the next try.

3. **Centralized Manager**: contains a highly available and scalable database that stores monitoring information. For near real-time analysis, Redis [4] is used to store the monitoring information for a configurable amount of time. We use Redis for both the Internal Manager and Centralized Manager. Redis is a high performance in-memory key-value store and it provides persistence, which means that data is preserved across restarts as on standard databases [6]. Moreover, Redis supports the publish/subscribe paradigm, so the monitoring application can subscribe to receive relevant information. Despite the NoSQL nature, the database provides an API to work with sorted sets. One can collect monitoring data and request the data at any time. The following commands are used to add and retrieve the data: `ZADD` and `ZRANGE` for Redis. For example, to add metric statistics, then the following command is called: `redis> ZADD metric timestamp value`

   To get the history of the metric values over the last 60 seconds the following command is called:
   
   `redis> ZRANGE metric (time_now-60) time_now`

   This method is more effective and convenient than the method proposed in [6] that uses pattern based queries to get a list of keys. For example, the pattern based query:
   
   `redis> KEYS("CLOUDi*)")`

   can be used to get a list of keys starting with `CLOUDi`. For convenient analysis, we design the monitoring information messages as a hierarchy or tree structure (Fig. 2). For long-term analysis and reducing Redis memory usage we periodically execute a back-end script which sequentially retrieves the data from Redis cluster and writes the data to the HBase [2] database. HBase is a NoSQL database runs on top of a Hadoop cluster.

4. **Monitoring and analyzing application**: monitoring and analyzing applications can access the monitoring information from both central and internal data store for near real-time analysis as well as HBase cluster for analyzing long-term
monitoring data. To present the monitoring data in real-time, we use Tornado web server, a scalable and non-blocking web server.

III. EVALUATION

We simulate three clouds using 3 physical nodes that are connected via Gigabit Ethernet. Each node has 2 Intel Xeon E5405 CPUs (quad core) and 8GB of RAM. The hypervisor KVM is installed on each node, and on top of that 5 VMs are deployed to simulate a cloud. Netem [10] is used to simulate WAN connections between three clouds by generating latency between nodes. According to [17], 50ms latency is good enough for simulating WAN connection between clouds.

A. Monitoring Hadoop based application on multi-clouds.

We deploy Hadoop [1] and HBase on the three clouds. Hadoop is an open-source MapReduce framework implemented by Yahoo. We chose Hadoop based applications to demonstrate our monitoring framework because Hadoop has become the standard framework for big data analytics across all industries. By using the framework, users can process their data in parallel by implementing a Map function to separate and spread the data to many machines or computation units for processing, and a Reduce function to aggregate the processing results. The Hadoop framework contains a single Master node that receives Job execution request from clients, distributes Tasks to Slave nodes and monitors the Tasks. The Slaves nodes execute the Tasks on slices of data located on the Slave nodes.

In this experiment, A Nutch [3] system is deployed on top of three clouds. Nutch is the current state-of-the-art in terms of web crawling. Nutch is based on Hadoop and MapReduce programming model to execute the algorithm in a highly parallel fashion across the machines in the clouds. A Nutch crawling job starts with an Inject Job to add initial URLs called a seed into crawlDB, after that it performs a loop of four Jobs including Generate Job, which selects a fetching list of URLs to be fetched, Fetch Job, which downloads the raw content from URLs in the fetch list, Parse Job, which reads the content and extracts all out links and UpdateDB Job, which calculates score of all URLs in the crawlDB to prepare for the next crawling round.

We install the probe program on all nodes to monitor network traffic exchange between clouds. The traffic exchange between clouds during Nutch runtime, including HDFS traffic, MapReduce traffic and HBase traffic is presented in Fig. 3. Looping patterns in network traffic that is generated by crawling Jobs are also presented (Fig. 3). CPU, memory utilization, and disk activity are described in Fig. 5. Fig. 4 and Fig. 6 respectively. The Master node VM1 receives the crawling Job requests, assigns Tasks to Slave nodes and monitors the Tasks as well as the status of Slave nodes; so it has different behavior comparing to Slave nodes in terms of CPU, memory utilization and disk activity. The more interesting part of the monitoring data is that the Slave node VM2 in cloud 1 behaves strangely compared to other Slave nodes. After inspecting the log files on the Slave node VM2 and the content of the Crawldb in the HBase database, we recognized that the crawling Tasks encountered pages such as http://wikipedia.org, http://wikipedia.com, http://wikipedia.de, http://www.webwiki.de, http://www.shopwiki.nl, etc which contained a significant number of out links, so VM2 in cloud 1 had to process larger data. That could be the reason for the odd behavior of the Slave node VM2.

![Fig. 2: Message structure](image2.png)

![Fig. 3: Nutch crawling jobs: Network traffic exchange between clouds with the sampling interval is 5s](image3.png)

B. Understanding multi-cloud application behavior under Denial of Service Attacks

In this section, we use the proposed monitoring framework to analyze abnormal behavior of the multi-cloud application.

---

1. http://www.tornadoweb.org
in a scenario where VMs are subjected to Denial of Service (DoS) attacks. In this work, we focus on DoS attack detection because DoS attacks are one of the most serious issues in cloud computing environments [14]. DoS attacks are attempts to obstruct or deny resources to legitimate users by attackers. Although cloud providers have supplied security tools for customers, they cannot protect against attacks between VMs on the same physical infrastructure. Even with light DoS attacks, the I/O and memory access performance of VMs decreases at a significantly higher rate than conventional physical machines [15]. There are many different forms of DoS attacks. In this paper, we investigate on the TCP SYN flood attack against target VMs. This is one of the most prevalent attacks on the Internet today [15]. Fig. 7 illustrates a flooding DoS attack in cloud computing environments. Attacker VMs transmit huge traffic to victim VMs, this significantly impacts the performance of application components residing in the victim VMs, even interrupting communication between components of the multi-cloud based applications.

We perform a real VM-to-VM attack on the simulated clouds running the Nutch application using the open-source Hping3 tool. The tool allows us to generate arbitrary packets to flood a victim host. We set Hping3 on three VMs in a cloud to generate TCP SYN packets and target ports of Hadoop to attack two VMs on the same cloud including one Master node and one Slave node. Table I shows the attacks that we perform.

By using the monitoring framework, we captured abnormal behaviors of the victim nodes. Fig. 8 and Fig. 9 present the number of in network packets on VM1 and VM2 respectively. Since SYN TCP Flood attacks generate massive SYN packets to the victims, Fig. 8 and Fig. 9 show attack patterns on HDFS and MapReduce traffic. By using a simple threshold based method, the attacks could be easily detected. Fig. 10 and Fig. 11 present abnormal behaviors in CPU and memory utilization of NameNode (VM1) and DataNode (VM2) under the SYN

---

TABLE I: TCP SYN Flood attacks list

<table>
<thead>
<tr>
<th>Start</th>
<th>Finish</th>
<th>Attackers</th>
<th>Victims</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18:23:32</td>
<td>VM3, VM4, VM5</td>
<td>VM1, VM2</td>
</tr>
<tr>
<td>2</td>
<td>18:40:14</td>
<td>VM3, VM4, VM5</td>
<td>VM1, VM2</td>
</tr>
<tr>
<td>3</td>
<td>19:20:35</td>
<td>VM4, VM5</td>
<td>VM1</td>
</tr>
<tr>
<td>4</td>
<td>19:36:29</td>
<td>VM3</td>
<td>VM1, VM2</td>
</tr>
<tr>
<td>5</td>
<td>20:09:37</td>
<td>VM4</td>
<td>VM1</td>
</tr>
<tr>
<td>6</td>
<td>20:16:36</td>
<td>VM4</td>
<td>VM1</td>
</tr>
</tbody>
</table>

---

*http://www.hping.org/hping3.html*
Flood attacks compared with their behaviors in Fig. 4 and Fig. 5. It turns out that the victim nodes consume more CPU and memory than usual. However, the CPU utilization does not have significant change, because the performed attacks are light attacks. The attacks affect the execution time of Hadoop jobs, but do not kill the jobs. It’s difficult to detect these kinds of attacks by using monitoring tools supported by cloud providers. There are several techniques to detect DoS attacks such as threshold based methods and statistic-based methods. In a threshold-based detection method, normal behavior profiles of applications are generated using the proposed monitoring system and utilized for future comparisons with newly measured information about applications. The dissimilarity between a new monitoring data record and the receptive normal profile is verified by a proposed detection mechanism. If the difference is higher than a pre-determined threshold, the data record is marked as an attack. Otherwise, it is labeled as a normal data record. Clearly, normal profiles and thresholds have direct influence on the performance of the detecting mechanism. Threshold-based methods are widely used because of simplicity, however they do not work well in highly dynamic cloud environments because of several limitations such as incremental false alarm rate, detection after the fact, and poor scalability [16]. In this work, we use an Entropy-based method to detect DoS attacks. Our DoS detection technique is inspired from [16]. However, in this paper, for the sake of brevity we will not cover DoS detection techniques in detail. Our focus will be on evaluating the proposed monitoring framework.

Fig. 8: The number of in network packets during DoS attacks at VM1 (Master node) in cloud 1

Fig. 9: The number of in network packets during DoS attacks at VM2 (Slave node) in cloud 1

Fig. 10: CPU Utilization during DoS attacks

Fig. 11: Memory Utilization during DoS attacks
IV. RELATED WORK

Cloud-based application monitoring has been a hot topic in research and industry for many years. By investigating the recent literature, we explored several monitoring frameworks that have their own characteristics and capacities. Cloud providers in industry all provide monitoring tools for their customers such as Amazon with CloudWatch. Nagios is widely used in industry for monitoring infrastructures [11]. Ganglia [5] is a scalable distributed monitoring system for high performance clusters. S. Benedict [5] gives a performance comparison of various monitoring tools e.g. CloudWatch, Nagios, Ganglia, and InterMapper.

Yong-min et al [12] proposed a network performance monitoring architecture for multi-clouds. The approach uses a real-time publish/subscribe model and supports both online and offline analysis of the network performance on multi-clouds. However, their work focuses only on network performance, while our work considers also other computation resource metrics such as CPU and memory utilization.

While many cloud-based application monitoring solutions for clients have been proposed, few of them take into consideration multi-cloud environments. For example, A. Di [8] introduce a mechanism to monitor and control the behavior of applications from user side on a cloud environment. However, it considers only applications that are deployed on a single cloud.

Zeginis et al [18] proposed a cloud provider side framework for monitoring the performance of service-based application deployed on multi-clouds. The framework performs monitoring at the IaaS, PaaS and SaaS layers in clouds. The open source OpenTSDB is selected to store monitored events. In the approach, the Amazon Cloudwatch and Nagios are used to monitor resource metrics. In our work, we develop a user-side monitoring framework that processes monitoring data from probes not from tools supported by cloud providers to help cloud customers monitor and manage their application deployed on multiple clouds.

As mentioned in [18], there are a number of EU-funded research projects that investigate cloud monitoring solutions. For example, RESERVOIR 5 presents the Lattice non-intrusive framework for monitoring cloud applications and VISION Cloud 6 introduces a monitoring framework that allows aggregating events, and applying rules on the events. However, not all these related works consider multi-cloud deployment.

V. CONCLUSION

In this paper, we introduce a scalable multi-cloud application monitoring architecture from the user-side that allows cloud users to observe and manage their applications in real-time. The advantages of the proposed framework compared with provider-side tools (e.g CloudWatch by Amazon) are the ability to adjust flexibly for specific purposes of customers and continuous fine-grained monitoring their applications. We also describe the abnormal behavior of Hadoop based application deployed on multiple clouds under DoS attack that is captured by using the framework. The monitoring framework is under active development, but the current source code is open source and available on Bitbucket7.

For future works, we intend to extend the proposed framework for cloud providers as well. We plan to develop an elastic scaling mechanism that automates provisioning of resources to adapt with changing workload by using the proposed monitoring framework.

VI. ACKNOWLEDGEMENT

This work was supported by the LEADS project, the European funded project with grant agreement number 318809.