Toward a lightweight framework for monitoring public clouds

Kun Ma*, Runyuan Sun*, Ajith Abraham†
*Shandong Provincial Key Laboratory of Network Based Intelligent Computing
University of Jinan, Jinan, China
{ise_mak,sunry}@ujn.edu.cn
†Machine Intelligence Research Labs
Scientific Network for Innovation and Research Excellence, Auburn, USA
ajith.abraham@ieee.org

***Abstract*—Nowadays, the cloud computing owners lack management and monitoring tools to ensure the performance, robustness, dependability, and security. To address this limitation, this paper described our experience with a lightweight monitoring framework using some extra development work. This framework performed end-to-end measurements at virtual machine instances and software in the public cloud. It monitors quality of service parameters of the IaaS and SaaS layer without modifying the implementation of the monitored object. In addition, we discussed the manager-agent and module-centralized architecture in details. All the modules make up the entire proposed framework to improve the performance of applications in public clouds.**

***Keywords*-cloud monitoring; performance evaluation; cloud computing;**

I. Introduction

Along with the cloud's increasingly central role to the services industry, monitoring cloud services as well as the applications deployed on them is becoming a priority. In particular, cloud service providers should closely manage their services [1]. However, currently deployed solutions are minimal at best. Overall, the integration of cloud monitoring and related techniques to effect an end-to-end automated monitoring and provisioning process over cloud environments is a hitherto neglected research area.

In order to solve this problem, this paper describes our experience with a public cloud, and discusses the architecture of a lightweight public cloud monitoring framework. An important finding of this paper is that we design the public cloud monitoring framework with open source solutions and extra significant development work.

II. Related work

A. Cloud monitoring methods

Monitoring of computing resources has been a topic of research interest and development for many years. However, traditional network monitoring approach may not apply to cloud monitoring. So far, there is no widely accepted standard or open source reference implementation for cloud management applications [2]. The topic of this paper is related to public cloud monitoring. Public clouds often have geographically diffuse and large resource pools, which need more investment in monitoring traffic and ensuring the scalability. Public clouds require continuous surveillance across multiple cyber attack vectors.

By exploring the recent literature, we discovered several monitoring systems, each one with its particular characteristics and abilities. Well-known clouds in industry all have their particular monitoring utilities. We will classify the available mechanisms and point out the drawbacks and inefficiencies. Nagios is the industry standard in the infrastructure monitoring, offering complete monitoring and alerting for servers, switches, software, and services about the status of resources. It is designed to run checks on hosts and services using several external plug-ins and return the status information to administrative contacts [3]. Although it includes valuable features and abilities, it does not provide a generic API and perform under small-time interval. Ganglia is a scalable distributed monitoring system for high-performance clusters and grids computing. It leverages widely used technologies and is on the base of a hierarchical design targeted at federations of clusters. The disadvantage of the method is not good for bulk data transfer (no windowed flow control, congestion avoidance, etc.) [4]. Clayman et al. [5] find a monitoring framework for service clouds, which succeeds in the scalability of the monitoring. This framework spreads in different layers (service, virtual environment, physical resources, etc.). The proposed framework offers the libraries and tools to build its own monitoring system. However, they do not present performance metrics and point out the boundaries.

Although many solutions are now available, cloud monitoring technology has not kept pace, partially because of the lack of open source solutions [2]. Those results further motivated us to achieve a lightweight framework that monitors public clouds. To this end, the design of a monitoring system should keep up with the expansion of the flexible ability, when an application or infrastructure scales up or down dynamically.

978-1-4673-4792-1

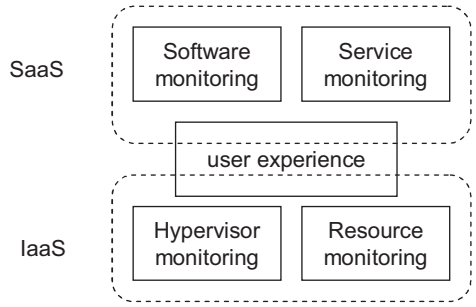


Figure 1. Monitoring mechanism architecture

B. Open source monitoring tools

There are several famous monitoring tools. Round Robin Database Tool (RRDtool) [6] is the open-source industry standard, high-performance data logging and graphing system for time-series data. The data analysis part of RRDtool owns the ability to create graphic representations of the data values collected over a definable time period. Nikto2 is an open-source web server scanner which performs comprehensive tests against web servers for multiple items in the quickest time possible. It has the advantage of frequently and automatically updated method of the scan items and plug-ins [7]. With these monitoring tools, it is possible to deploy a public cloud monitoring solution using extra significant development work.

III. Monitoring framework

A. Monitoring mechanism architecture

The framework that we propose consists of four components, presented in Figure 1, deployed across the different cloud layers. It is divided into two layers: Software as a service (SaaS) and Infrastructure as a service (IaaS) layer. Since the monitoring of Platform as a service (PaaS) depends on the specific platform, we do not discuss it in our paper.

At the bottom is the supporting infrastructure, such as the hardware resource utilization (CPU, memory, disk, networking et al.) on both physical machines and virtual machines, and hypervisor that manages the guest operating systems. This performance metrics, which stays the same regardless of distinguished infrastructures, are extracted to model the runtime infrastructure in the cloud.

On top of the infrastructure is application (such as application servers, Web servers) and service that provides an appropriate run time environment for applications. These applications provide a wide set of services that assist service developers in delivering a professional and commercial service to end users. Commonly, the software in SaaS layer will expose their own interface for monitoring and management, and this metrics is gathered and organized to form the runtime monitoring model of this layer.

Besides IaaS and SaaS monitoring, the user experience is also an important entity to monitor. We put the user experience of cloud tenants as a reference to discover the relationship between users habits and cloud performance. Also it is guidance on the future management of the cloud. Interactive information between end users and clouds are monitored. It mainly contains access frequency, IP distribution, time of staying and so on, are monitored. The service developers may adapt and improve their software to attract more users of public cloud.

B. Hierarchy of resource entity models

Concrete resource entities in a running public cloud are organized in a hierarchy, as shown in Figure 2. This hierarchy can be extended by adding new entities emerged in different cloud layers. Such an organization makes an extension to the model more feasible. New entities can be added in directly by inheriting one of the existing entities. Each entity consists of a set of attribute-value pairs. Children entities can inherit attributes from their parents. And values are obtained from real-time cloud monitoring statistics.

IV. Monitoring framework architecture

A. Manager-agent architecture

In a cloud, multiple entities need to be monitored simultaneously to coordinate their resource utilization to achieve a balance. Therefore, the framework is required to monitor individually and take decisions centralized. This paper achieves a manager-agent styled monitoring framework, as shown in Figure 3. The manager provides the interface between the human network manager and the management system. The monitoring agent is a small daemon which collects system information periodically. In order to analyze the collected data, we provides mechanisms to store and monitor the values in a variety of ways. These agents are responsible for collecting runtime information and sending the UDP packet to the manager. By default, we provide the basic monitoring of common metrics: IaaS monitoring plug-ins, SaaS monitoring plug-ins. Other plug-ins are written to allow you to further extend the default monitoring system for your own applications. All the plug-ins conform to our interface specification. We provide the user interface with the administrators and tenants of public clouds. This will help them view the operating condition of public clouds.

B. Module-centralized architecture

The proposed monitoring framework provides the comprehensive monitoring in forms different plug-ins. These plug-ins include virtualization, availability, SNMP performance, Web security, user experience tracker, and UnixBench. It provides many kinds of alert, information statistics to help the administrator find the problem in time. Each module is a daemon, which collects the data from the monitored object.

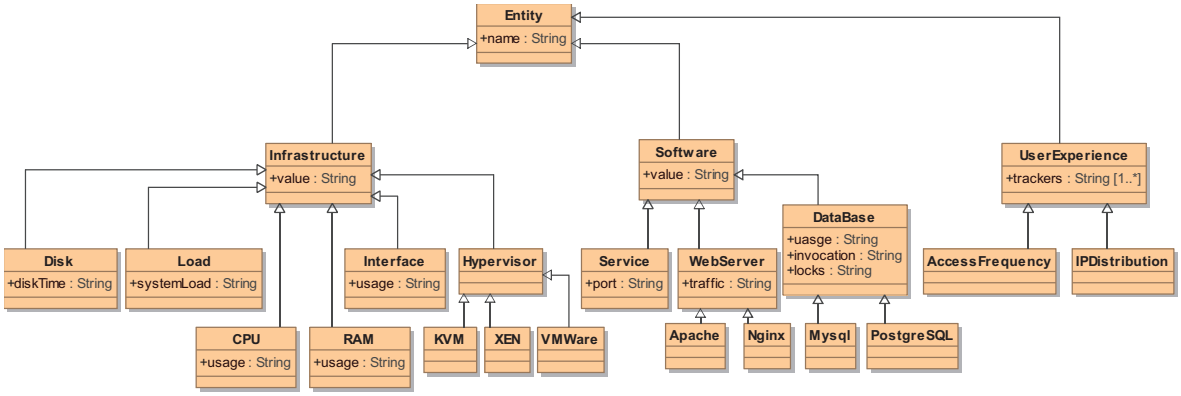


Figure 2. Hierarchy of entity models

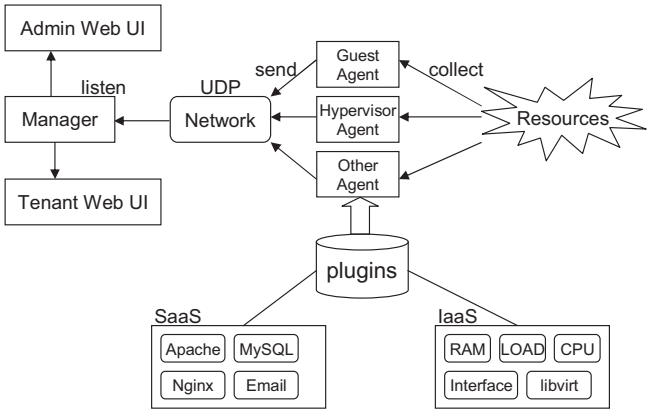


Figure 3. Manager-agent architecture and its plug-ins

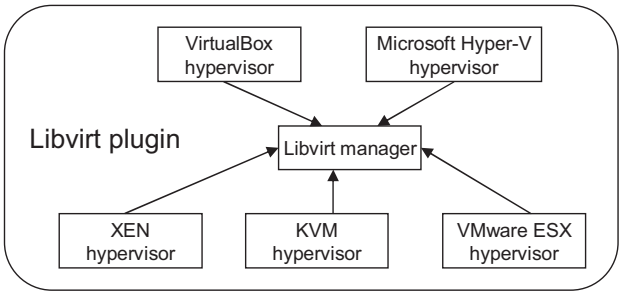


Figure 4. Infrastructure libvirt monitoring architecture

1) Virtualization monitoring module: This monitoring module uses the libvirt plug-in to gather statistics about virtualized guests on a system shown in Figure 4. The libvirt public API supports many commonly hypervisor drivers, such as KVM, XEN, Hyper-V, VMware ESX and VirtualBox [8]. With libvirt plug-in, CPU, memory, networking and device usage for each guest could be collected without

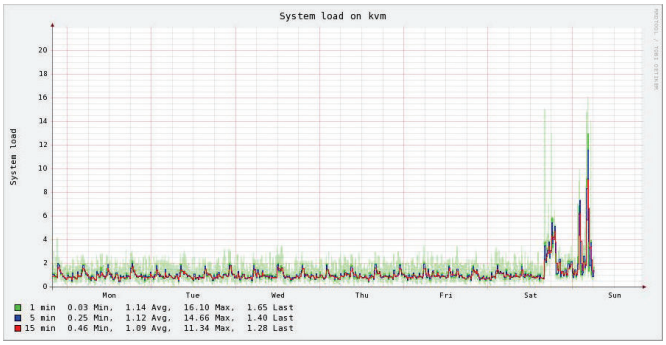


Figure 5. The system load collected from the last one week

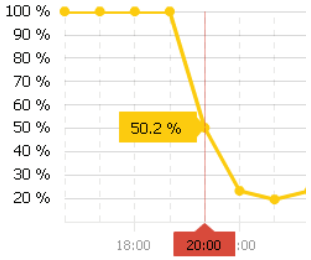


Figure 6. The availability curve graph of a virtual machine in public clouds

installing any software on the guest. As the plug-in receives statistics from the hypervisor directly, it is suitable for the physical and virtualized machines. The system load collected from the over the last one week is shown in Figure 5.

2) Availability monitoring module: Availability monitoring module can refer to the process of collecting data and reporting on the status of the critical website. Availability, in the simplest form, is Availability= Uptime/(Uptime + Downtime). Inherent availability from one probe is as follows: Availability=MTTF/(MTTF+MTTR)=detection success number/detection total number. In this formula, MTTF is mean time to failure, and MTTR is mean time

to repair. In order to get the accurate availability, the probe point distributes around the nation with China Unicom, China Telecom and China CERNET backbone network node. The mean availability is as follows. Availability=(detection success number)/(detection total number). The detail of the availability curve graph of a virtual machine in public cloud is shown in Figure 6 in applying this monitoring module.

3) SNMP performance monitoring module: For the public clouds, we focus on more performance monitoring (service, processes, system users, memory and disk drive usage) more than the availability. Simple Network Management Protocol (SNMP) is tracked for statistics graphing and historical reference of this metrics. This module will appear an alert when thresholds of the performance of a device were exceeded. Services and processes running on devices can be monitored to verify they are running, and not running out of control. An example of the memory and disk monitoring in public cloud is shown in Figure 7.

4) Application monitoring module: Application monitoring module provides the most commonly service monitoring in the software layer, including the Web server and database. For different Web servers, there are different monitoring methods, depending on the different APIs. For example, the mod_status module monitors the throughout of Apache Web server, while stub_status module monitors the throughout of Nginx Web server. Take the database Mysql for example, the mysql proxy is used to intercept the database request (alter, select, update, insert and delete) to be analyzed. The number of issues per second for various SQL-commands is shown in Figure 8.

5) Web content vulnerability scanner: We design a Web content scanner to scan for the known security vulnerabilities of a website. These supported vulnerabilities of the scanner contain Cross-Site Scripting (XSS), SQL injection, command execution, directory traversal and insecure server configuration. It is possible to deploy a vulnerability scanner using only open source solutions like Nikto2, conforming to The Open Web Application Security Project (OWASP).

The whole process of web scanning is as follows. First of all, the file webscanner.html with the security code is put in the root path of the Web application. In addition, the vulnerability scanner checks this file to verify the validity. Finally, the scanner does some automatic verification on websites to tell what vulnerability it is. Because every pattern is in a standardized XML file, the vulnerability test is a flexible and customizable approach.

6) User Experience Tracker: Compared with the availability monitoring, user experience tracker provides the more accurate method to calculate the user access speed, and positioning real experience of each visitor. This module embeds a JavaScript in the webpage of the monitored website. The user access information (IP and location) is automatically recorded after the load of the webpage. An example of user experience tracker is shown in Figure 9. The average response time of visitors from Gansu Province of China is 350.1 ms. It plays an important part in discovering the relationship between end user habits and cloud performance. Furthermore, the metrics from the tracker is a better guide on how to attract more users in the public clouds.

7) UnixBench: In the area of IaaS, over-commit refers to the practice of committing more virtual resources to customers than the actual resources available on the underlying physical cluster. Most of the prevailing hypervisors, such as Hyper-V, VMWare ESX, KVM, and XEN, support both CPU and memory over-commit When an IaaS service provider practices over-commit, the over-commit parameters are usually unknown to the end user. When an end user creates a VM that is labeled as 1 CPU core and 1 GB memory, the CPU usage limit for that particular VM might be 1, 0.5 or even 0.1 physical processor core. Similarly, a host server with 16 GB physical memory may be committing 18GB or even 32 GB virtual memory to virtual machines. In a public IaaS environment, system administrators can categorize virtual machines according to their resource consumption pattern. In this study, the performance of the virtual machine is tested with the open-source benchmark suite UnixBench. The experiments show that the system scores of UnixBench rather than the parameters of CPU cores, and the amount of memory indicate the actual performance of the virtual machine from the same hypervisor. Figure 10 shows the UnixBench test results of virtualized guest. For virtual machines from the same hypervisor, as the virtual machine gets bigger, the performance gets better in the none over-commit circumstance; for virtual machines from the same hypervisor, as the virtual machine gets bigger, the performance gets worse in the over-commit circumstance

V. Conclusion

In this paper, we first listed the monitoring methods and tools. Currently, we focused on the popular metrics of the guest in the public clouds. Towards our goal of building such a lightweight and scalable framework, we integrate some open-source monitoring tools, and do some extra significant secondary developments work to archive some modules. In addition, we discuss the manager-agent and module-centralized architecture to perform end-to-end measurements at virtual machine instances and software in the public cloud. Our proposed framework could improve performance of applications in public clouds.

Acknowledgment

This work was supported by the Technology development Program of Shandong Province under Contract Number 2011GGX10116.

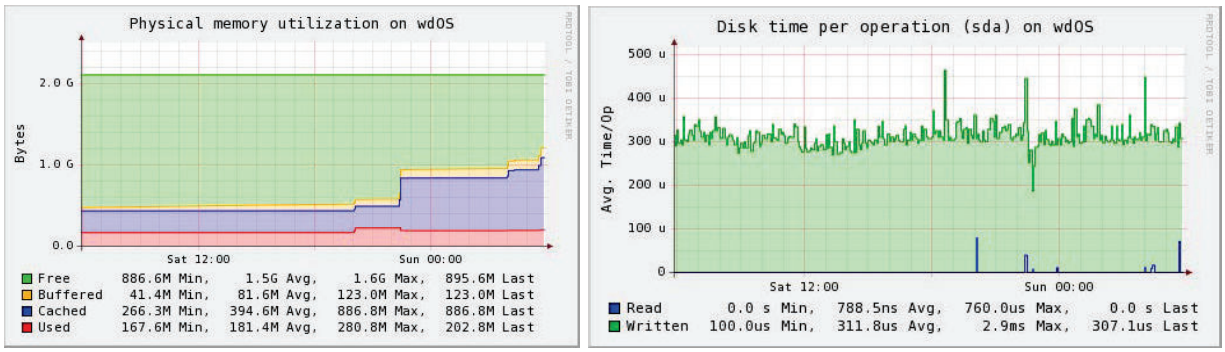


Figure 7. An example of the memory and disk monitoring in public clouds

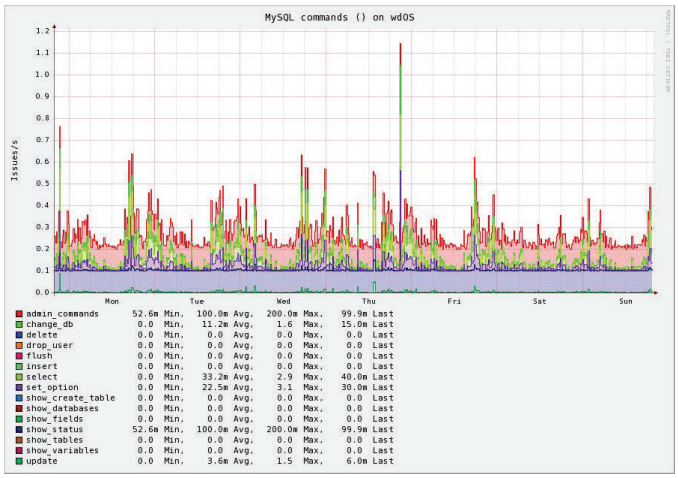
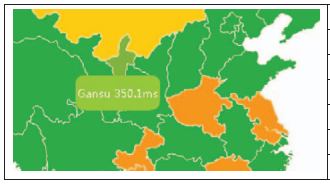


Figure 8. The number of issues per second for various SQL-commands of MySQL



Province	Tracker times	Response time
Ningxia	256	185.6
Gansu	325	350.1
Shanxi	206	50.5
Sichuan	234	20.9
Sinkiang	198	242.3
Tibet	243	252.1

Figure 9. An example of user experience tracker

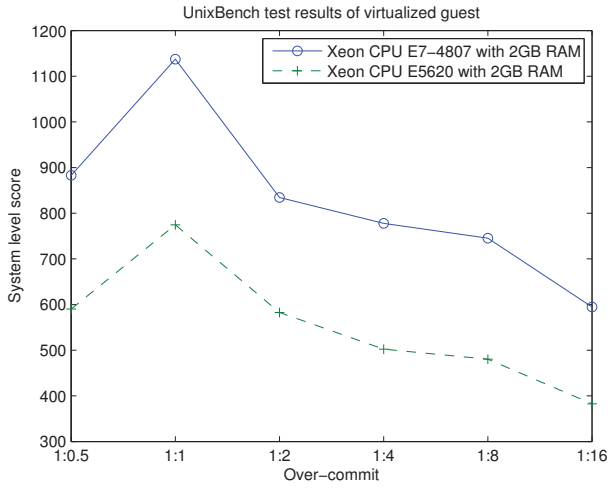


Figure 10. UnixBench test results of virtualized guest

REFERENCES

[1] H. N. Van, F. D. Tran, and J. M. Menaud, "Autonomic virtual resource management for service hosting platforms," in *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, 2009, pp. 1–8.

[2] S. A. de Chaves, R. B. Uriarte, and C. B. Westphall, "Toward an architecture for monitoring private clouds," *IEEE Communications Magazine*, vol. 49, no. 12, pp. 130–137, 2011.

[3] E. Imamagic and D. Dobrenic, "Grid infrastructure monitoring system based on nagios," in *Proceedings of the 2007 workshop on Grid monitoring*, 2007, pp. 23–28.

[4] M. L. Massiea, B. N. Chunb, and D. E. Cullera, "The ganglia distributed monitoring system: Design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, 2004.

[5] G. Tselentis, A. Galis, A. Gavras, S. Krco, V. Lotz, E. Simperl, B. Stiller, and T. Zahariadis, "Monitoring service clouds in the future internet," in *Towards the Future Internet - Emerging Trends from European Research*, 2010, pp. 115–126.

[6] J. Russell and R. Cohn, *RRDtool*. California: Book on Demand Ltd., 2012.

[7] S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic, "Secubat: a web vulnerability scanner," in *Proceedings of the 15th international conference on World Wide Web*, 2006, pp. 247–256.

[8] F. Han, J. Peng, W. Zhang, Q. Li, J. Li, and Q. Jiang, "Virtual resource monitoring in cloud computing," *Journal of Shanghai University (English Edition)*, vol. 15, no. 5, pp. 381–385, 2011.