AN ENHANCED MONITORING MECHANISM FOR IAAS PLATFORMS

Mohammed Jameel Sadeeq Barwary
Assistant Lecturer, Department of Statistics, Faculty of Administration and Economics, University of Duhok(UoD), Kurdistan Region - Iraq

ABSTRACT

The monitoring mechanisms of open-source IaaS software OpenNebula and monitoring system Ganglia were analyzed. Reduce overload of retrieving resource usage information by deploying the Ganglia monitoring agent. And for improving the robustness of monitoring subsystem, we take measures on the privileged domain of virtualization nodes to hide the agent to prevent regular or malicious users checking, modifying, unloading or killing it intentionally or unintentionally. The mechanism is very helpful to enhance the effectiveness, reliability and sturdiness of the monitoring system of IaaS platforms.

KEYWORDS: IaaS, Monitoring, Consolidation, Agent.

I. INTRODUCTION

With the outstanding advantages of elasticity, economics and etc., cloud computing is embraced by many IT companies, universities and research institutes. According to the NIST (National Institute of Standards and Technology) definition of cloud computing, the cloud model contains five essential characteristics, on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service [1]. Importantly, most of the attractive characteristics rely on the effectiveness and reliability of resources monitoring system. Currently, for IaaS (Infrastructure as a Service), there are many different monitoring systems for commercial or open source IaaS platforms, which have different merits and demerits about overload or robustness [2], many researchers have proposed different solutions to solve performance or robustness problems. For example, G. Katsaros et al designed a multi-level monitoring framework, which is more effective and scalable [3]. Javier Povedano-Molina et al design a high adaptable and scalable monitoring architecture for cloud, which ensures an accurate measurement of resources in cloud keeping a low overhead [4]. And D. Zou et al designed a trusted monitoring framework to ensure the integrity of monitoring environment through trusted computing technology [5].
Most of the previous works focused on the efficiency or monitoring environment security, but lacking the protection mechanism of monitoring system components. With the development of cloud computing and datacenter expansion, the monitoring system should consider both of the above two aspects.

In this paper, we present an enhanced monitoring mechanism for IaaS platform based on OpenNebula, detailing the monitoring agent as the alternative of native monitoring probes and the consolidation measures to protect the monitoring component of monitoring infrastructure.

II. MONITORING MECHANISM OF OPENNEBULA AND GANGLIA

OpenNebula is an enterprise-ready open-source platform to manager cloud data centers and to build IaaS platform. The architecture of Ganglia is shown as figure 1.

![Figure 1. The architecture of OpenNebula](image)

A. The monitoring mechanism of OpenNebula

The native built-in monitoring subsystem of OpenNebula relies on the Xen hypervisor tools and OpenNebula-defined probes. The architecture of monitoring subsystem of OpenNebula as shown in figure 2.

![Figure 2. OpenNebula Monitoring Subsystem](image)

To retrieve the resource information successfully, the OpenNebula Frontend needs to do the following works:
1) EveryNode need to have an identical unix account that will be used by the Frontend to connect and execute commands remotely, such as the monitoring probes and virtual machines management procedures. By default, the account is named ‘oneadmin’.

2) According to the password-less SSH login principle, making the ‘oneadmin’ account in Frontend can password-less login every node.

The above two steps is critical to monitoring subsystem of Frontend. Because any node was added to IaaS platform, the Frontend needs to copy monitoring probes into it remotely.

3) After above operations were performed successfully, the ‘oneadmin’ account in Frontend will remote password-less login every node and execute the probes to gather the resources information periodically.

B. The monitoring mechanism of Ganglia

Analyzing the principle of the monitoring mechanism, there is a fact that the monitoring subsystem overhead is relate to the number of nodes, it will increase along with the increase nodes. To avoid the above problem, we adopt open source distributed cluster monitoring system Ganglia as the alternative, which is developed by Berkeley and with the advantages of scalability, overload lowly.

The architecture of Ganglia [6] as shown in figure 3.

![Ganglia Architecture](image)

**Figure 3.** Ganglia architecture

The Ganglia monitoring system contains three main components, monitor daemon (gmond), metadata daemon (gmetad) and PHP web frontend. Especially, the gmond run in the nodes which we wish to monitor, which collects monitoring data of own, announces its presence on the local network and receive the state of other gmond nodes through muticast or unicast systems.

III. ENHANCED MONITORING MECHANISM IMPLEMENTATION

C. Integrating gmond of Ganglia

According to the principle of Ganglia, we employ gmond as the alternatives of the probes to retrieve the resource information [7][8]. Then the monitoring subsystem can retrieve the whole resource information of IaaS platforms through accessing any one gmond in the multicast systems. The architecture of new monitoring subsystem after integrating gmond as shown in figure 4.
Adopt the new monitoring mechanism, not only reducing the monitoring overhead in a large deployment as the monitor system don’t rely on the SSH connections to the nodes, but also easily extending the user-defined monitoring metrics, such as monitoring current running services or the overload information of specific process.

D. Consolidation monitoring mechanism

As the matter of fact that most of the attractive characteristics of cloud computing rely on the effectiveness and reliability of resources monitoring system, for improving the robustness of monitoring agent, we take measures on Linux guest operating system to hide the agent in privileged domain (which is the Dom0 operating system in Xen [9][10] virtualization environment as shown in Figure 4) to prevent regular or malicious users checking, modifying, unloading or killing it intentionally or unintentionally.

The consolidation basic principles as follow.

1) when user executes commands or call APIs to view information of system processes, the unit for hiding monitor agent will judge the identity of who performs the above operations, whether it is the authorized user or not;
2) If the user is authorized, the request information containing monitoring agent process will be return successfully.

It is need to emphasis that the authorized user is not the administrator or ‘root’ account in Linux operating system, which is a custom defined user account in privileged domain operating system (Dom0). The purpose of setting the account is to modify or configure the monitoring agent as needed to satisfy the different resource management and schedule requirements of IaaS platforms.

The implementation is detailed below.

According to the principle of process view of Linux operating system as shown in figure 5, it contains the following steps:

1) When an application launch a process view request, the application tools (Such as ps, pstree, top, etc.) will launch a system call ‘sys_open’ function to open the ‘/proc’ directory. If successful, it will return the file descriptors;
2) According to the descriptors, the application will continue calling ‘sys_getdents’ function to traverse the directory entry of ‘/proc’ directory and return the process directory entry set;
3) Finally, according to the process directory entry set, the application will get the process information.
Through intercepting and filtering the above native traverse process, we can hide the specific process. To perform it, we introduce a unit for hiding monitoring agent. The new process of the process view in Linux as shown in figure 6.

![Diagram of process view](image)

**Figure 5.** The native process of process view

**Figure 6.** The new process of process view
For the ‘sys_getdents’ function as shown in figure 7, we modify the ‘readdir’ function to add user identity inspection mechanism. Only return the whole directory entry set to authorized user, otherwise, return the information excluding the monitor agent related information to regular user.

![Diagram](image)

Figure 7. ‘sys_getdents’ function

The flowchart of the whole above process as shown in figure 8.

![Diagram](image)

Figure 8. The flowchart of consolidation
After adding the consolidation mechanism, the enhanced monitoring architecture as shown in figure 9.

![Image](image.png)

**Figure 9.** The enhanced monitoring architecture

### IV. EXPERIMENTAL RESULTS

Based on the enhanced monitoring mechanism detailed in this paper, we take experiments in our IaaS testing platform.

Considering the factors of copyright, effectiveness and performance, we choose the open-source software OpenNebula [11], Xen [12] and CentOS operating system [13] to build our IaaS platform. Xen is an open-source hypervisor, which makes it possible to run many instances of operating systems on a single machine.

The platform architecture as shown in figure 10.

![Image](image.png)

**Figure 10.** Minimal IaaS platform architecture

The related software parameters of the IaaS testing environment are shown in Table I and the physical CPU should be 64-bit capable.

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>5.8, x86_64</td>
</tr>
<tr>
<td>Xen</td>
<td>3.4.4</td>
</tr>
<tr>
<td>Ganglia</td>
<td>3.2.0</td>
</tr>
</tbody>
</table>

**TABLE I. RELATED SOFTWARE PARAMETERS**
Utilizing the new monitoring system, we can also get the resource usage information of hosts and virtual machines. The memory usage information of one virtual machine in our IaaS environment as shown in figure 11.

![Memory monitoring information](image)

**Figure 11.** Memory monitoring information

For the overload and effectiveness of the monitoring system, based on our environment, we assume the number of Nodes is $N_{\text{Total}}$. And according to the principle of the native monitoring subsystem of OpenNebula which needs to remote password-less login every node and execute the probes to gather the resources information periodically, we also assume that every remote login operation consumes $R_{\text{Every}}$ unit’s resource, the number of hosts monitored in each interval is $N_{\text{Interval}}$, the consume time of each interval operation is $T_{\text{Interval}}$ (and assume that every node of one interval finishes data acquisition operations within time of $T_{\text{Interval}}$ by parallel mechanism), the total time and resource consumption of one traverse are $T_{\text{Total}}$ and $R_{\text{Total}}$. Thereinto, $N$, $R$, and $T$ are positive integers. We can deduce the following equations:

1. $R_{\text{Total}} = N_{\text{Total}} R_{\text{Every}}$
2. $T_{\text{Total}} = (N_{\text{Total}} / N_{\text{Interval}}) T_{\text{Interval}}$

First, according to “(1)”, the total resource consumption of one traverse is increase linearly along with the increase nodes. And the whole overload of data acquisition is bore by Frontend, while according to Fig.4 the alternative mechanism lets nodes and Frontend to share the overload of data acquisition. A period of time of the overload of gmond agent on Frontend as shown in figure 12.

![The overload of monitoring agent](image)

**Figure 12.** The overload of monitoring agent
Second, according to “(2)”, the waiting time of retrieving the latest resource information of one node in IaaS satisfies $T_{Time} \in [T_{Interval}, (N_{Total}/N_{Interval})* T_{Every}]$. While after adopting the gmond of Ganglia, the above waiting time is a fixed value and only depends on the specific parameters of gmond, which it is more effectiveness and real-time.

For the robustness of monitoring components of monitoring system, the verification process of the consolidation mechanism is detailed as below.

When unauthorized user including ‘root’ account who launches a process view request, the hiding unit will filter the returning results excluding the gmond monitoring agent as shown in Figure 13. While the authorized user (Built-in ‘admin’ account) can retrieve the whole process information as shown in Figure 14.

![Figure 13. Unauthorized user views monitoring agent process](image13)

![Figure 14. Authorized user views monitoring agent](image14)

V. CONCLUSION

We have present an enhanced monitoring mechanism for IaaS platforms: 1) Adopt the ganglia monitoring agent as the alternatives of native probes to reduce overload; 2) Take measures to hide the monitoring agent to consolidate to prevent it from being killed. We construct a more efficient and robust monitoring subsystem for IaaS platforms. Actually, the mechanism also adapts to the PaaS or SaaS environments, and the regular cluster or grid environments.

ACKNOWLEDGMENT

This work has been supported by the project of cloud computing network security of the Research Institution of China Mobile.
REFERENCES