SCM: A Design and Implementation of Monitoring System for CloudStack

Lin kai, Tong Weiqin, Zhang Liping, Hu Chao
School of Computer Engineering and Science, Shanghai University
Shanghai, P.R. China
ms_newaslan@hotmail.com, wqtong@shu.edu.cn, {zlpssyy, hcleioo}@163.com

Abstract—**Infrastructure as a service (IaaS) is one of the most basic cloud-service models in cloud computing, which provides computers, network and storage. Apache CloudStack (ACS) is an open source software for creating, managing, and deploying infrastructure cloud services. In order to provide high availability, enable Service Level Agreement (SLA) and respond effectively to the Quality of Service (QoS) requirements of computing resources, we proposed a well-designed monitoring system for CloudStack platform named SCM. The SCM monitoring system collects accurate metrics from both physical and virtual resources, including the main components of ACS (system virtual machines, Secondary Storage, Primary Storage and management servers), and makes these data easily accessible and human readable, which is quite friendly to the CloudStack users. Then SCM monitor stores these collected metrics in distributed database for further analysis or to be used by other models of Cloud computing like the billing systems. In this paper, we will describe the architecture of the SCM monitoring system and introduce the technologies used to implement the SCM monitor.**

Keywords—IaaS, monitoring system, Storage, distributed.

I. INTRODUCTION

Cloud computing with the character of optimization the resources utilization of hardware and software, offering on-demand service and reducing the costs of the Cloud users, made it widely used to deliver services over the Internet [1]. In order to provide high availability, enable SLA and respond effectively to the QoS requirements of computing resources, accurate and fine-grained resource monitoring activities is required. In Cloud software stack, job scheduling, dynamic load balancing and billing system also need the data from a resource monitoring system [2].

Apache CloudStack [3] is one of the most popular open source IaaS solutions. In this paper, a monitoring system named SCM is proposed to monitoring the Apache CloudStack platform. In IaaS Cloud environments, two aspects should be considered:

- IaaS hardware and software: In Cloud environment, there are various kinds of hardware and software, including physical hosts, network devices, storage devices and databases. Monitoring system should obtain the performance data of these hardware and software, and report the real-time running status.
- The Cloud users' resources: Everything the user has in the Cloud. In the case of CloudStack, these are instances, disk volumes, guest networks, templates, ISOs, etc. For all these components, the Cloud user needs clear and reliable knowledge of their status.

Our goal is to develop a monitoring system for CloudStack. Since there are still little monitoring systems designed for CloudStack. The monitoring system can collect utilization information from both physical and virtual resources. The monitoring metrics should be accurate, i.e. they are as close as possible to the real value to be measured. This can help the administrators know the status of Cloud system, and give end users a clear view of their resources in Cloud. These data will be stored in a scalable database and can be easily queried. Also it will monitor the components of CloudStack. And a friendly user interface is also very important.

The paper is structured as follows. Section II describes some background knowledge which is important to the SCM monitoring system. In section III we introduce some existing monitoring systems. The SCM monitoring system's architecture is given in section IV. In section V, we discuss the detail of the major components of the SCM monitoring system. In section VI, we give the prototype of the SCM.

II. BACKGROUND

A. Introduce the Apache CloudStack

Apache CloudStack is a highly available, highly scalable IaaS cloud computing platform designed to deploy and manage large networks of virtual machines. CloudStack can offer both public cloud services and private cloud services or a hybrid one. Itcomposes with compute orchestration, Network-as-a-Service, user and account management, a powerful API and a friendly User Interface. It supports the most hypervisors such as VMware, XenServer, KVM and Xen Cloud Platform (XCP) [3]. The architecture of CloudStack is showing in fig. 1.

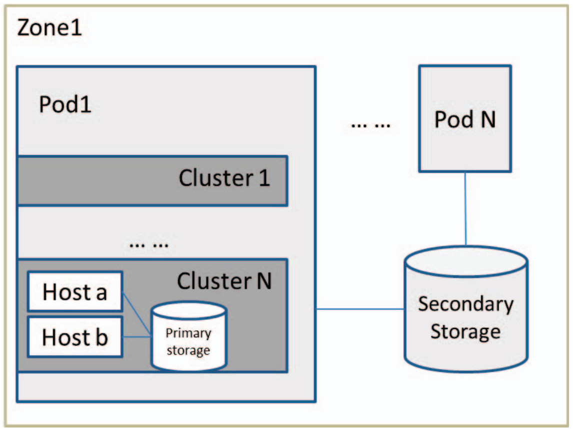


Fig. 1 The architecture of CloudStack

B. Common terminologies of Apache CloudStack

A host is a single computer and is the smallest organizational in the CloudStack platform. A host can be installed with KVM, XenServer or VMware ESXi. The number of guest virtual machines that can be hosted on CloudStack can be determined by number of hosts and capacity of each host. Hosts are not visible to the end user.

Cluster is the second level of physical scaling in CloudStack platform. Cluster provides a way to group hosts. These hosts have the same hypervisor type and share the same primary storage. In one Cluster, virtual machine instances can be live-migrated from one to another without interrupting service to the user.

Pod is a collection of different types of clusters. It often represents a single rack, and it is the second-largest organizational unit within a CloudStack deployment. It is not visible to the end user.

Zone is the largest organizational unit in CloudStack platform. Zones can provide physical isolation and redundancy. Users have option to choose zone while deploying their guest VMs. Admins can setup private zones which is only accessible to specific domain. Each node in a zone shares the secondary storage and network.

Guest VM's root disks and other additional data disks are stored on primary storage. Instances in a particular cluster have same primary storage for them. The speed of primary storage has directly impacts on guest VMs' performance.

Secondary storage is used to store templates, ISOs and snapshots on ACS. Secondary Storage VM communicates with it and retrieves templates and ISO images from URLs.

One or more physical networks can be associated with each zone. CloudStack has four different network traffic (management, guest, storage, public), of all these network traffics, storage traffic is the one most likely to be the bottleneck of the platform.

III. Related work

With the rapid development of Cloud computing, there already have been many works on the Cloud monitoring area. And many of the existing monitoring systems can be used in the Cloud, since Cloud computing evolves from cluster computing and Grid computing.

Ganglia [4] is a hierarchical designed distributed monitoring system for Grids, cluster and other high performance computing systems. It uses XML for data representation and RRDtool for data storage and display. Nagios [5] provides instant awareness of IT infrastructure, detects and repairs problems and mitigates future issues before they affect end-users and customers. Zabbix [6] is a general-purpose enterprise-class open source distributed monitoring solution for networks and applications that can be customized for use with Cloud. CloudStack ZenPack is a plugin for Zenoss [7], which manages both events and alerts and display the resource usage.

DARGOS [8] is a highly adaptable and scalable monitoring architecture for multi-tenant Clouds. It ensures an accurate measurement both of physical and virtual resources and it is easily to defining and monitoring new metrics.

Chukwa [9] is an Apache project for data collecting built on top of Hadoop and inherits Hadoop's scalability and robustness. It has a flexible, dynamically controllable data sources named adaptors. Then adaptors send data to collector process and storage it in HDFS data store.

Kanyun is a OpenStack [10] monitoring and measurement tools developed by sina Cloud computing team. It can track resource usage of tenants, including the virtual machine's performance, and the data can be aggregated and statistics. Kanyun includes three modules as worker, data-server, API-server. The worker module is responsible for collecting and monitoring. Data-server stores the metrics from worker module. API-server provides queries and statistic external interface.

Ceilometer [11] is an infrastructure to collect measurements within OpenStack. Its primary targets are monitoring and metering, and it is easily expandable to collect for other needs. Agents doing data collections of Ceilometer are independent of the overall system.

These monitoring systems have different features, some of them have been proposed for many years, such as Ganglia, Nagios. They are robust but less support for CloudStack. Kanyun and Ceilometer are both built for OpenStack, but still many concepts can be used in the SCM monitoring system.

IV. Architecture of the SCM

SCM is a flexible monitoring system supporting for cloud environments, which can monitor both physical and virtual resources. SCM users can choose their interested metrics and set a custom interval. In order to meet these requirements, SCM needs a well-designed user interface, and flexible, dynamic data sources. In Clouds, monitoring metrics are also important to the billing systems, job scheduling and other Cloud components. Because of the characteristics of Cloud environment, the monitoring metrics will be dynamically changed and the volume of data may become very large, a scalable and high performance storage system is needed.

The SCM monitoring system has four main functionalities, which are metric collection, information processing and storage,

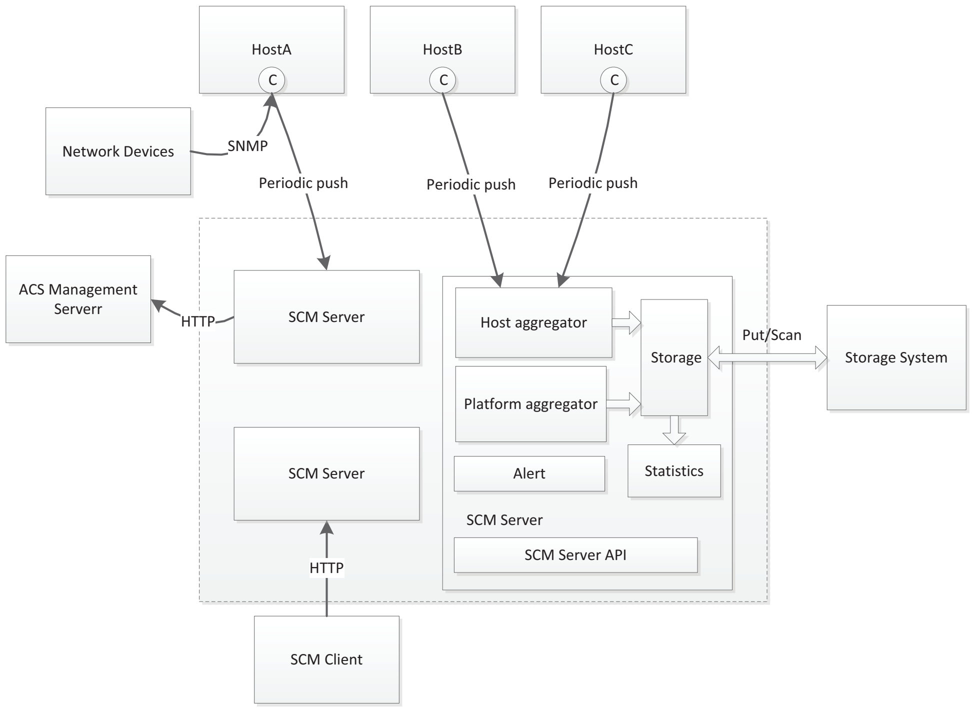


Fig. 2 The architecture of the SCM monitoring system

metric display, alert. The architecture of the SCM monitoring system is shown in Fig. 2.

A. Collectors

In Apache CloudStack environment, the hosts have different meanings. These hosts may be physical or virtual, customer instances or system virtual machines, so the metrics need to be collected vary with the hosts' type. In the SCM monitoring system, we use collectors as the data sources which are deployed on each host. These collectors can easily be configured to collect different metrics. In fact, the collector offers a framework, in which users can develop their own programs to collect metrics they interested in. The collector periodically retrieves performance metric values from the host, e.g. cpu usage, memory usage, disk I/O. When the host becomes management server or storage server, the performance metrics of MySQL, tomcat, NFS and other CloudStack components are also collected. As mentioned above, CloudStack has different network traffics on a host, some of traffics do not to need be monitored. The collector monitors the public and storage traffics. The collector also monitors the network devices through SNMP. These metric values are then pushed to SCM Server.

B. The SCM server

The SCM server is the core of the SCM monitoring system. There are five main modules of the SCM server. Host aggregator is used to aggregate the metric values from the collectors. A host aggregator may receive metric values from a lot of collectors. Apache CloudStack provides an API that gives programmatic access to all the management features. We designed the platform aggregator to communicate with ACS management servers and call the ACS API through HTTP to get the CloudStack related information, such as the version of CloudStack and how many zones, pods, clusters and hosts in the current environment, etc. After a pre-set time, the aggregators send the metrics to the storage module. The storage module is used to communicate with the storage system, putting the metric values into the storage system or getting values from it. The storage module receives the metrics from the aggregators and stores all these data locally, when the metrics file is large enough, it puts the metrics into the storage system. This can reduce the I/O operations on the storage system. The statistics module is a data processing module. It analyses the metric values from the storage module and provides the average, minimal, maximum, performance outliers, etc. To improve the availability of the ACS, abnormal running information should be reported to the Cloud users immediately. The alert module obtains exceptions from statistics and records the information, and then notifies the Cloud user. If the ACS scale is large, there are hundreds or thousands of hosts, multiple SCM Servers may be needed for load balance.

C. The SCM Client

The metric values are organized as a tuple (metric name, timestamp, value, tags), these tuple are not friendly to the Cloud users. So just collecting various resource utilizations information is not enough to explain the observed performance of hosts or applications. In order to let the Cloud users easily to understand the meaning of these metric values, it is very important to display information in a simple and flexible way. The SCM Client gives an overview of the whole system, and displays the metric values in time series graphs with several filters, which is used to help the Cloud users quickly find the minimal or maximal of the current metric value or calculate the average performance in a period of time. Also the Cloud users can customize the graphs by selecting the metric names and tags in tuples. Then only the interested metric values will be displayed in the user interface.

D. Storage system

The metric values need to be stored persistently for analysis as well as displayed on the fly. Resources in the Cloud change dynamically and the deployment of the Cloud is large. Monitoring such distributed system may produce a large amount of metric values. So the storage system should be scalable and flexible, with the ability to collect many thousands of metrics from thousands of hosts and applications at a high rate.

V. Implementation

In this section, we will display some major implementation details of the SCM monitoring system.

The SCM collects metric values from both physical and virtual resources. We modified an open source tool called tcollector, which is a client-side process that gathers data from local host and pushes the data to the TSDs (Time Series Daemons). The collector module is extended from the tcollector. It collects metrics value from both the physical and virtual resources and sends metrics to the SCM server. The collector module is a framework. It executes the scripts to get the monitoring information. We write our own scripts to monitoring the utilization of CPU, memory, disk I/O, network I/O, etc. by parsing files in the Linux /proc file system. We also monitor the components of the applications used by apache CloudStack, such as MySQL, tomcat, NFS and other applications. The ACS currently supports many different hypervisors. We use libvirt [12], an open source API, daemon and management tool, for collecting vCPU number, virtual memory, etc. from hypervisor. The ACS also providers a script to check the running state of SSVM (secondary storage virtual machine), the connection between the SSVM and the ACS management server, and is the SSVM successfully mounted to the secondary storage. We capture the output of the script and send the information to the SCM server. The communication between the collectors and the SCM server are through zeroMQ [13] which is a lightweight message queue and it is fast and easily to be used. Other platform related information is collected by the platform aggregator. The ACS provides a very powerful API which provides administrators access to all of CloudStack's features. We write a java library to call the CloudStack API. Through the ACS API, we can obtain the version of the ACS, and the architecture of current environment. And the list* APIs (listVirtualMachines, listNetworks, listTemplates, etc.) give all the details of the ACS including events and alerts which are very important information for Cloud users to solve problems.

Since most of the metric values has the time series properties, we choose the time series database. Compared with Round Robin database and relational database, time series database can handle millions of time series with minimal append and data extraction time and with limited disk space requirements. Account for the scalability, we use HBase [14] as the storage database, which can easily scalable to store thousands of metric values.

The SCM client is a web application built by bootstrap which contains HTML and CSS-based design templates. The SCM presents overview of the ACS and the running state of hosts in the system. It uses graphics to display the system usage and offers events and alerts information. The Cloud users can select which metric value to be graphed.

VI. A prototype of the SCM monitoring system

Cooperating with Shanghai Telecom, we deploy an experimental Cloud environment using Apache CloudStack solution. We set up a KVM cluster with six Dell R720, each of the server has 8 cores, 32G memory and 600G disk space. The CloudStack management server and MySQL are on the same physical machine.

We develop a prototype of the SCM monitoring system and deploy it on the Apache CloudStack. We create an instance to act as the SCM server and collectors are deployed on the physical hosts and VMs. Figure 3 to 5 gives an overview of the SCM Client dashboard.

The SCM is a real-time monitoring system. It collects CPU utilization, Memory usage, I/O throughput and displays these real-time metrics on the fly. Figure 3 shows the Win2008 instance's CPU utilization information, these real-time data let the Cloud users know the performance of their applications and the bottleneck of resources.

The SCM client shows variable message of the system to the end users. In figure 4, it shows the web page under the dashboard menu. There are eight menus such as dashboard, zones, pods, etc. in every page which present the different parts of the Apache CloudStack platform. Figure 4 shows the details of the utilization of system capacity including CPU load, memory usage, public ips, management ips and so on. This can help the Cloud administrators the whole system status. And it displays users of CloudStack, instances, templates and ISOs. .

Events and alerts are usually important to a monitoring system. Figure 5 shows the events which are formatted as (User, type, level, description, account, domain). It helps administrators to understand what happened in the latest period of time.

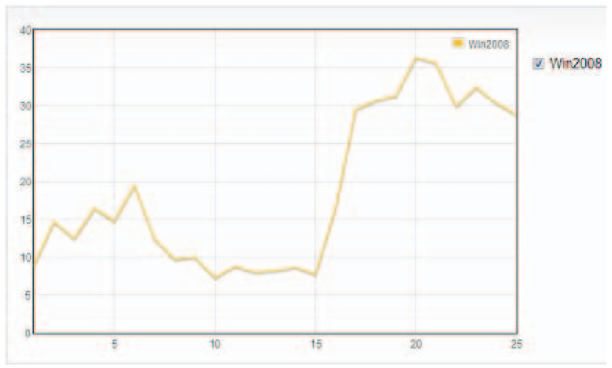
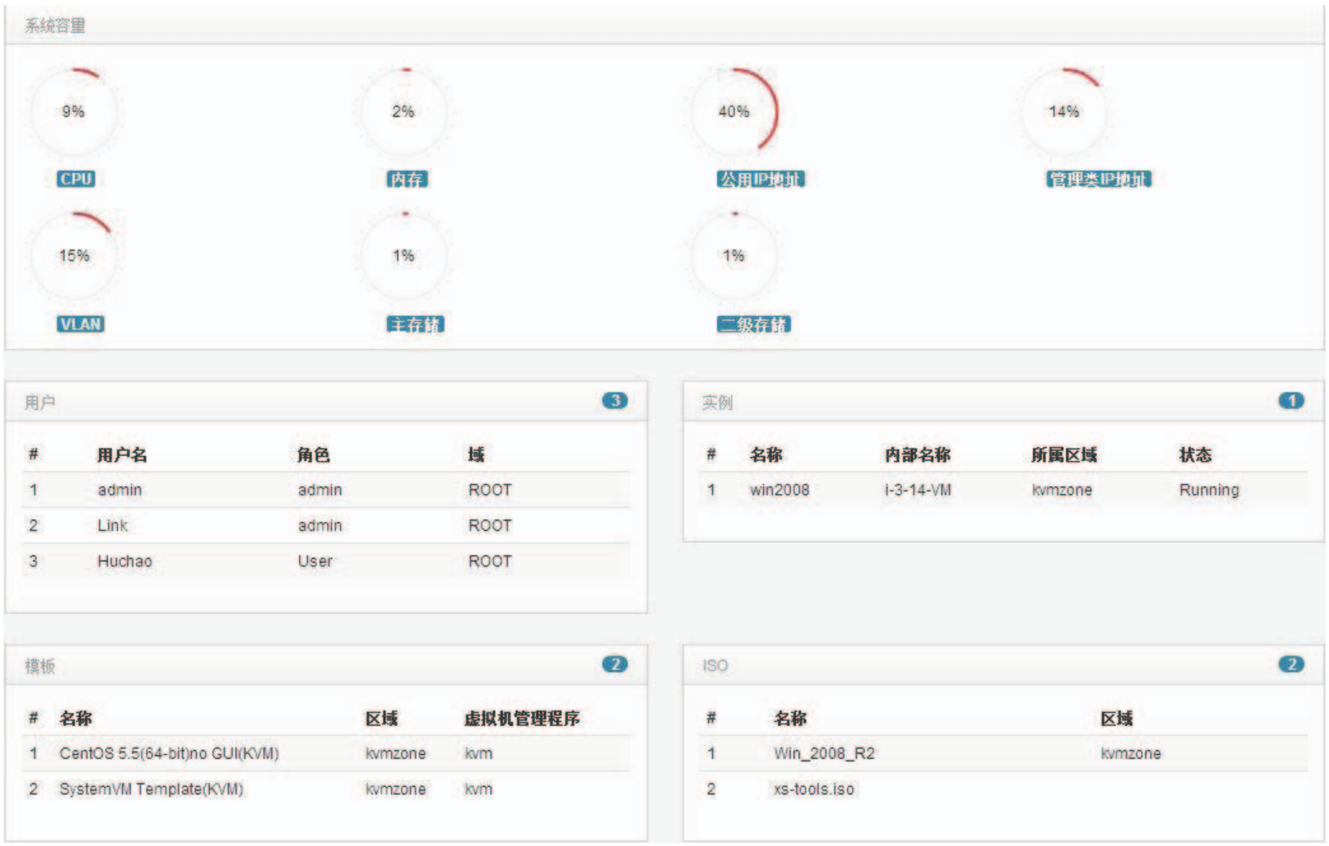


Fig. 3 Cpu utilization imformation



用户 3

#	用户名	角色	域
1	admin	admin	ROOT
2	Link	admin	ROOT
3	Huchao	User	ROOT

实例 1

#	名称	内部名称	所属区域	状态
1	win2008	i-3-14-VM	kvmzone	Running

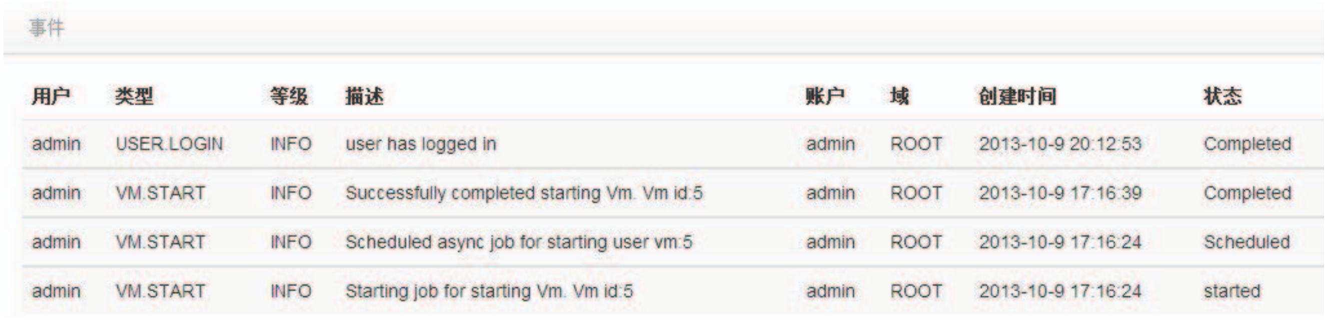
模板 2

#	名称	区域	虚拟机管理程序
1	CentOS 5.5(64-bit)no GUI(KVM)	kvmzone	kvm
2	SystemVM Template(KVM)	kvmzone	kvm

ISO 2

#	名称	区域
1	Win_2008_R2	kvmzone
2	xs-tools.iso	

Fig. 4 System capacity, accounts and other user resources


事件

用户	类型	等级	描述	账户	域	创建时间	状态
admin	USER.LOGIN	INFO	user has logged in	admin	ROOT	2013-10-9 20:12:53	Completed
admin	VM.START	INFO	Successfully completed starting Vm. Vm id:5	admin	ROOT	2013-10-9 17:16:39	Completed
admin	VM.START	INFO	Scheduled async job for starting user vm:5	admin	ROOT	2013-10-9 17:16:24	Scheduled
admin	VM.START	INFO	Starting job for starting Vm. Vm id:5	admin	ROOT	2013-10-9 17:16:24	started

Fig. 5 Events information

VII. CONCLUSION

In this paper we propose a scalable and flexible monitoring system architecture named SCM. It is used to monitor the Apache CloudStack platform. The SCM monitoring system collects metric values from both the physical and virtual resources and stores these data for further analysis and statistics. We also developed a prototype system to validate our thoughts. In the future work, we will complete the architecture and make it more robust. Monitoring virtual resources is much more daunting, libvirt can't satisfy our demand in some situation. We try to make the SCM communicate directly to the hypervisors.

ACKNOWLEDGMENT

This work is supported by Innovation Action Plan supported by Science and Technology Commission of Shanghai Municipality (No.11511500200) and Shanghai Leading Academic Discipline Project (J50103).

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinsky, G. Lee, D. Patterson, A. Rabkin, I.Stoica and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Compuitng", Communications of the ACM, New York, Vol 53 Issue 4, pp 50-58, April 2010

[2] Giuseppe Aceto, Alossio Botta, Walter de Donato, Antonio Pescapè, “Cloud monitoring: A Survey”, Computer Networks, 2013 Elsevier.

[3] Apache Project, Apache CloudStack, 2013 [online] http://cloudstack.apache.org.

[4] M. L. Massie, B. N. Chun, D. E. Culler, “The Ganglia distributed monitoring system: design, implementation, and experience”, Parallel Computing, pp 817-840, 2004

[5] Nagios, http://www.nagios.org accessed 2013.

[6] Zabbix, http://www.zabbix.com accessed 2013.

[7] Zenoss, http://www.zenoss.com accessed 2013.

[8] Javier Povedano-Molina, Jose M. Lopez-Vega, Juan M. Lopez-Soler, Antonio Corradi, Luca Foschini, “DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds”, Future Generation Computer Systems, 2013 Elsevier.

[9] Boulon, J., Konwinski, A., Qi, R., Rabkin, A., Yang, E. and yang, M., “Chukwa: A Large-scale Monitoring System”, In Cloud Computing and its Applications, Chicago, IL. 2008.

[10] OpenStack, http://www.openstack.org accessed 2013.

[11] OpenStack project, Ceilometer, 2013 [online] http://wiki.openstack.org/wili/Ceilometer

[12] M. Bolte, M. Sievers, Georg Birkenheuer, O. Niehörster, A. Brinkmann, “Non-intrusive virtualization management using libvirt”, Proceedings of the Conference on Design, Automation and Test, Europe, PP574-579, 2010

[13] zeroMQ, http://www.zeromq.org accessed 2013

[14] Apache Project, Apahce Hbase, 2013 [online] http://hbase.apache.org