SNMP-Based Monitoring of Heterogeneous Virtual Infrastructure in Clouds

Ya-Shiang Peng
Department of Information Management
National Chi Nan University
Puli, Nantou 545, Taiwan
yashiang1986@gmail.com

Yen-Cheng Chen
Department of Information Management
National Chi Nan University
Puli, Nantou 545, Taiwan
ycchen@ncnu.edu.tw

Abstract—**With the rapid growth of cloud computing, a great number of servers are deployed in a data center to provide virtual machines to a variety of users. Several virtualization techniques have been developed in the recent years. Platforms for virtual machines, e.g. VMware, Xen, KVM, and VirtualBox, may adopt different virtualization techniques for particular purposes. This leads to a heterogeneous virtual infrastructure in clouds. For effective management of a heterogeneous virtual infrastructure, there is a demand for a standardized interface for the monitoring of a lot of virtual machines in the data center. This paper will propose the use of SNMP in VM monitoring. The proposed monitoring scheme makes use of two standard MIBs, MIB-II and Host Resources MIB, to provide essential information for VM configuration and performance monitoring. In addition, an enterprise MIB is developed on AgentX to provide detailed VM instance information. A web-based management system is also implemented to demonstrate the superiority of SNMP in monitoring heterogeneous virtual machines.**

Keywords: Cloud computing; Virtualization; SNMP;

I. INTRODUCTION

As internet applications are developed in many areas, there are a variety of requirements of hardware and software in both clients and servers for different applications. Recently, instead of purchasing hardware/software for particular applications, many enterprise customers employ or rent platforms or services provided in the Internet cloud. To meet the various hardware/software requirements, service providers in the clouds provide platforms and services via virtualization techniques. Via virtualization, service providers may allocate the required resources in a virtual machine, run as a process or program in one of the real servers in the data center. As the lease of the resources is terminated, the resources can be easily released and reallocated for another application.

A virtual machine (VM) can be created by a system software or a user program in a real computer, called host machine. Multiple VM instances can be installed on a host machine, if the host machine can offer all the demanded resources for those VM instances. Therefore, virtualization allows service providers to save cost in hardware investment, maximize resource utilization, and fulfill the customization of cloud services. Currently, there have been a number of virtualization platform products with different virtualization techniques. Virtualization is realized by introducing a virtualization layer between hardware and the operating system. The virtualization layer, realized by a VM hypervisor or VMM (Virtual Machine Monitor), enables the creation of virtual machines in different operating systems, and allows their shared access to the real hardware resources, including CPU, memory, and I/O devices. [1]. A number of VMM products have been developed, for example, VMware, Xen, KVM, and OpenVZ [2][3][4]. Depending on the need of modification in the guest OS, i.e. the OS of a VM, there are two major techniques for platform virtualization: *full virtualization* and *paravirtualization*. Without any change in the guest OS, full virtualization fully abstracts the configurations and resources for the guest OS. VMware Server, KVM, VirtualBox are typical full-virtualization VMMs. Paravirtualization requires modification in the guest OS to support better CPU and I/O performance in platform virtualization. Among the platforms by paravirtualization, Xen hypervisor is the most popular one, which supports multiple OS platforms. For example, Amazon EC2 [5] adopts Xen in its VM services.

For better QoS of cloud services, system/network administrators in a data center should be always aware of the current status of the host machines in the server cluster, including their CPU loads, storage usage, network utilization. Furthermore, administrators are more interested in how many VM instances are allocated in a host machine and how well each VM instance is running. As a great number of servers are deployed in a data center to provide virtual machines to a variety of users, it is crucial to the data center to develop an effective scheme for monitoring the health and performance of those virtual machines as well as their host machines. Currently, the monitoring of VMs is performed through the inherent management interface supported by the VM platform over which the VMs are built. Generally, the management interface is proprietary, and the provided features are particular for a VM product. If more than one VM platform are adopted in a data center, administrators should be aware of the management interface of each VM platform. Moreover, it is very possible that the collected management information from different VM platforms cannot be shared and integrated in a consistent fashion. Therefore, there is a strong demand for a unified interface for the monitoring and control of heterogeneous VM platforms. Currently, SNMP (Simple Network Management Protocol) [6] has been widely used in remote monitoring of network devices and hosts. A lot of MIBs (Management Information Bases) have been developed to define the standard management information for various management applications. Therefore, through SNMP, managers can access management

information in a consistent manner from network devices/hosts of different vendors. Essentially, the monitoring of VM hosts are the same as network devices/hosts except the management information supported. In this paper, we would like to discuss the use of SNMP in managing VM services in a data center. Management by SNMP requires the support of an SNMP agent in the managed host, and the MIBs in the agent provide the management information needed for a management application. Currently, there are no standard MIB implementations particular for managing VM infrastructure. In this paper, we will first examine the *managed objects (MOs)* defined in two popular standard MIBs, MIB II [7] and Host Resources MIB [8], to find the set of MOs which can be used in monitoring VMs. These *MOs* are available in current SNMP agents installed on MS Windows and Linux. Since the *MOs* in MIB II and Host Resources MIB are only useful for monitoring host machines, we will further define a MIB module for monitoring the resources consumed by each VM instance. The newly defined MIB requires additional implementation in the agent. We use the Agent Extensibility Protocol (AgentX) [9, 10] to implement the MIB over the *libvirt* API [11], which provides unified programming APIs to access a variety of VM platforms. To demonstrate the heterogeneous features, we will implement a web-based management system for managing three different VM platforms, including VMware, Xen, and KVM, installed on two different OSs, MS Windows and Linux.

II. RELATED WORK

Recently, the monitoring of VMs and hosts receives much attention. A few schemes have been developed to monitor the pool of resources deployed for VM services. SBLOMARS [12][13] is a resource monitoring system for VMs in a Grid computing environment. In a machine managed by SBLOMARS, several sub-agents, called ResourceSubAgents, are implemented for remote monitoring. Each sub-agent is responsible for monitoring a particular resource, e.g. CPU, memory, software, or network. Inside a sub-agent, SNMP is used to retrieve the required management information stored in the standard MIB supported by the managed machine. SBLOMARS can remotely monitor the overall networking and computing resources of a host machine where VM instances are running. However, it is unknown how each individual VM instance in a host machine consumes those resources. For example, when a manager finds a heavy CPU load in a host machine, he/she cannot determine the VM instance which mainly consumes the CPU resource. In [14], a VM monitoring scheme for a grid computing platform is proposed to simplify the management effort and improve the usage of resources for VM services. The proposed scheme is based on a framework with grid information services. The framework is composed by information agent, information collector, information translator, information provider. The retrieval of resource information of a VM is realized by the information agent, which is located in each VM. The information agent retrieves resource information from the CIMOM (CIM Object Manager) if the VM is built byVMware ESX. For VMs on other virtualization platforms without the support of CIMOM, the information agent implemented as a lightweight Java agent on the VMs.

Kyrre M Begnum et al. [15] proposed MLN (Manage Large Networks) to allow managers to easily configure and monitor VM services on Xen and User-Mode Linux. To be managed by MLN, each host machine needs an MLN daemon, a process to receive instructions from the manager. In the context of MLN, a configuration language is used to describe the configuration for a VM, including network devices, memory, storage, and network interfaces. Furthermore, the virtual environment created by the MLN configuration can be one across multiple real machines. This enables the deployment of large VMs. The MLN daemon only provides a little information relevant to VM monitoring, including the number of MLN projects and VMs, and the usage and capacity of memory. It is unknown how the CPU and storage of the host machine is used and how each VM consumes the resources.

The above approaches are proprietary solutions in particular VM products or platforms. For a data center with heterogeneous VM platforms, there is a demand for a unified approach to simplify the management effort across multiple VM platforms. By introducing a middleware over a variety of VM platforms, libvirt [11] provides a unified interface to the upper management plane so that a management application can be developed independent of the actual VM platforms. Currently, libvirt has been supported on VMware, Xen, KVM, and QEMU. As shown in Figure 1, after libvirt is installed, a deamon, called libvirtd, runs in the managed VM server [16]. All operations through the libvirt API is performed locally by the libvirtd in the managed VM server. libvirt is also responsible for receiving remote operation requests via TLS or SSH from the manager server.

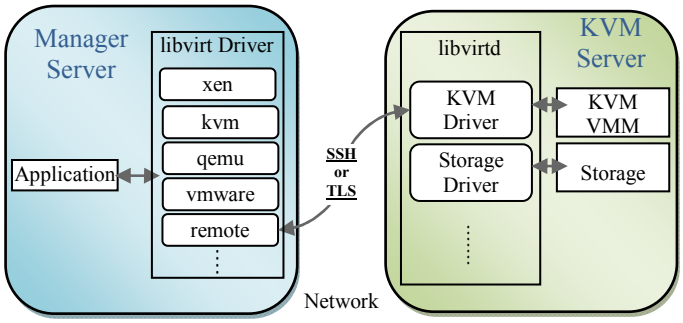

Figure 1 libvirt deamon and its remote access

The effort of libvirt lies in the programming interface, not in the protocol for remote monitoring. Recently, libvirt-snmp [17], a libvirt subproject, is initiated to allow the use of SNMP in monitoring the VM resources under libvirt. The introduction of SNMP in llibvirt-snmp is similar to our approach in this paper. The project is still under development.

III. SNMP-BASED MONITORING FRAMEWORK

SNMP has been widely used in the management of TCP/IP networks. A lot of MIB modules have been developed to define standard managed objects for a variety of network devices and applications. Currently, there is no broad discussion about the use of SNMP in cloud management. In addition, no specific MIB module is proposed especially for cloud management. In this section, we first propose an SNMP-

based monitoring framework for heterogeneous VM platforms, as depicted in Figure 2. Essentially, the framework is not new with respect to the original SNMP model. Due to the lack of specific MIBs for cloud management, the support of additional enterprise MIBs will be required in current SNMP-based monitoring approaches. For ease of adding MIBs in VM servers, we adopt an extensible agent AgentX in each managed host machine. Coexisting with AgentX, the inherent SNMP agent in a host machine is not disabled such that the standard MIBs supported by the host machine can also be retrieved independent of AgentX. Since only one SNMP agent is allowed in UDP port 161, AgentX is configured to use another unused UDP port. As shown in Figure 2, the proposed monitoring framework is built over three different VM hypervisors, including VMware, Xen, and KVM, installed on two different OSs, MS Windows and Linux.

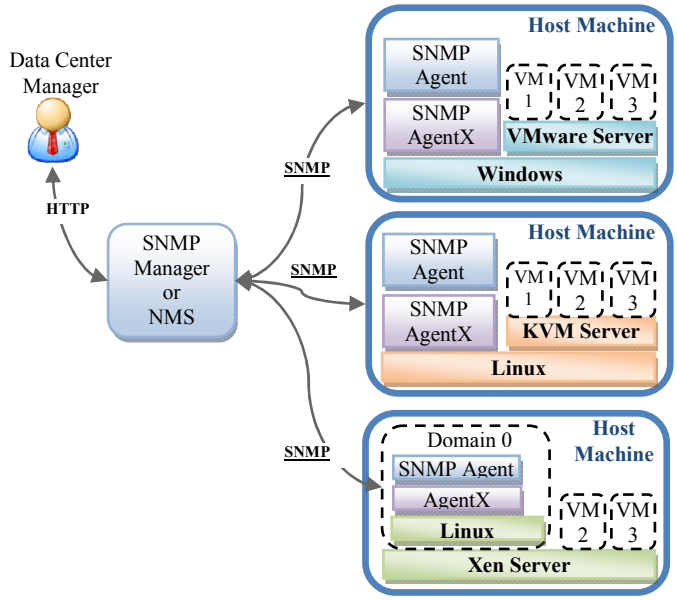

Figure 2 The proposed SNMP-based monitoring framework

Previous approaches didn't consider the use of a standard protocol between the manager and the VM servers. In our framework, a VM manager application is implemented as an SNMP manager. Only SNMP is used to access the management information required for VM monitoring. A web-based NMS (Network Management System) is then developed based on the management information retrieved via SNMP. A detailed description of each component of the proposed framework is given as follows.

A. *SNMP Agent*

The SNMP Agent, in the host OS of each VM server, provides the access to the standard MIBs, regarding the monitoring of resources from the viewpoint of the host OS. If the host OS is MS Windows, the inherent SNMP agent of the OS is used. For Linux-like OSs, Net-SNMP is installed and the *snmpd* daemon is enabled to act as the SNMP agent of the host OS. Both SNMP agents support MIB-II and Host Resources MIB. The *ifTable* of MIB-II is used to collect network traffic statistics for the host machine as well as each VM instance. The usage statistics of CPUs, memory, and storages of the host machine are available in the Host Resources MIB. In addition, the *hrSWRunTable* and *hrSWRunPerfTable* of the MIB may provide status and performance data for each VM instance. Details in the use of those MIB objects will be described in sub-section D.

B. *AgentX*

Standard MIBs in the SNMP Agent of the host OS are useful for monitoring the resources of the host machine. If each VM instance runs as an individual process of the host OS, it is possible to monitor the VM instances from the *hrSWRunTable* and *hrSWRunPerfTable* in the Host Resources MIB. However, some VM platforms adopting paravirtualization, e.g Xen, don't fork a process for each VM instance. In addition, the detailed information about the resource consumption of each VM instance is not available in current standard MIBs. To allow the support of enterprise MIBs designed for monitoring VM instances, we deploy AgentX as an alternative SNMP agent in the host machine. The dual SNMP agent scheme is a compromised solution to the monitoring of both the host machine and virtual machines.

In this paper, we define an enterprise MIB, NCNU-VM-MIB, to provide managed objects for the monitoring the resources consumed by VM instances. An implementation of the MIB is realized via the libvirt API, through which the VM instances of different VM platforms can be monitored in a consistent manner. In NCNU-VM-MIB, a *vmDynamicTable* table is defined. Each entry of the table represents a VM instance activated in the host machine. Each entry, as shown in Figure 3, is composed by seven object types:

vmDynamicIndex: a unique index of the VM instance.
vmDomId: the identifier of the VM instance. The identifier, created by the VM hypervisor, is unique among the VM instances created by the same VM hypervisor.
vmDomName: the name of the VM instance.
vmCPUTime: the CPU time used by the VM instance.
vmDomMaxMemory: memory size allocated to the VM instance.
vmDomMemoryUsage: current memory usage of the VM instance.
vmDomVCPU: number of CPU cores configured for the VM instance.

```
vmDynamicEntry ::= SEQUENCE {
    vmDynamicIndex          INTEGER,
    vmDomId                 INTEGER,
    vmDomName               OCTET STRING,
    vmDomCPUTime            INTEGER,
    vmDomMaxMemory          KBytes,
    vmDomMemoryUsage        KBytes,
    vmDomVCPU               INTEGER
}
```

Figure 3 vmDynamicEntry

C. NMS

NMS (Network Management System), as an SNMP manager, consist of two modules: *Information Gatherer*, and *NM Web Service*.

1) Information Gatherer: This module periodically retrieves MIB objects from the SNMP Agent and AgentX of each VM server. Table I lists the MIBs and their objects adopted for VM and host monitoring. Available management information in these objects includes CPU time, memory usage, memory allocation, number of CPU cores, VM identity, VM name, and network trafiic for each VM instance. Since MIB-II and HOST-RESOURCES-MIB were defined for monitoring hosts, not for VMs, the availability of the above metrics depends on how a VM hyperviosr interacts with the host. Table II summarizes the support of metrics/information in different VM hypervisors.

TABLE I. MIBS AND OBJECTS RETRIEVED BY INFORMATION GATHERER

MIB	Objects
HOST-RESOURCES-MIB	hrSWRunTable hrSWRunPerfTable hrStorageTable
MIB-II	system ifTable
NCNU-VM-MIB	VMDynamicTable

TABLE II. THE SUPPORT OF MANAGEMENT INFORAMTION IN VM HYPERVISORS

	hrSWRunPerfTable		hrSWRunTable				ifTable
	CPU Time	Memory usage	vCPU (s)	Allocation of Memory	VM Name	VM ID ; Domain ID	Virtual Network Interface
Xen *	×	×	○	○	○	○	○
KVM	○	○	○	○	○	○	×
VMware Server	○	○	×	×	×	×	×

* : VM resources cannot monitored through HOST-RESOURCES-MIB if the VM hypervisor is Xen in HVM (Hardware Virtual Machine).

2) NM Web Service: This module provides a web interface to the managers of the data center. The performance of VMs in all the hosts with hetergeneous VM platforms can be minitored through the single web interface with consistent views.

D. Interpretation of Object Values in HOST-RESOURCES-MIB and MIB-II

As indicated in TABLE II, the support of objects for VM monitoring is different among the three VM hypervisors studied in this paper. In addition, we found that some valuable information can be obtained from the parameters supplied to the VM software when it was initially loaded. These parameters are stored in the *hrSWRunParameters* object the *hrSWRunTable* table. However, different VM software programs have their own parameters. Therefore, a particular interpretation of parameters is required for each VM software program. In the following, the interpretation of parameters for three VM platforms is described respectively.

1) Xen: Xen adopts two virtualization approaches: paravirtualization and hvm (Hardware Virtual Machine). In a VM in hvm mode, there is a process corresponding to domain 0 of the VM. Moreover, an entry with *hrSWRunName* = "qemu-dm" in the *hrSWRunTable* table denotes a running VM instance. Therefore, we can identify all the entries corresponding to VM instances. For each of these entries, the hrSWRunParameters object of the entry are interpreted to obtain VM identity, VM name, number of CPU cores, and memory allocation, as shown in Figure 4. However, the CPU time and memory usage of each VM created by Xen is absent in the augmented *hrSWRunPerfTable* table. In this case, AgentX is used to provide the detailed information.

In Xen's VMs, only domain 0 can access the physical network interface directly. Other VMs, in domain U, access the network indirectly through domain 0. Thus, Xen creates a virtual interface (vif) for each VM instance. All packets sent to/from a VM instance will be first forwarded to the corresponding vif. Since all the virtual interfaces have a distinct entry in the *ifTable* Table of MIB-II, we can monitor the network traffic of each VM instance through the objects in *ifTable*. Figure 5 shows an entry of *ifTable* corresponding to a virtual interface used by a VM instance. There is a simple naming rule, "vif#.0", in the *ifDescr* of the entry for the virtual interface of a VM in domain #. Therefore, we can easily find all the virtual interfaces for VMs according to the values of *ifDescr*. Then, network traffic statistics for each VM can be obtained in the proposed SNMP-based framework.

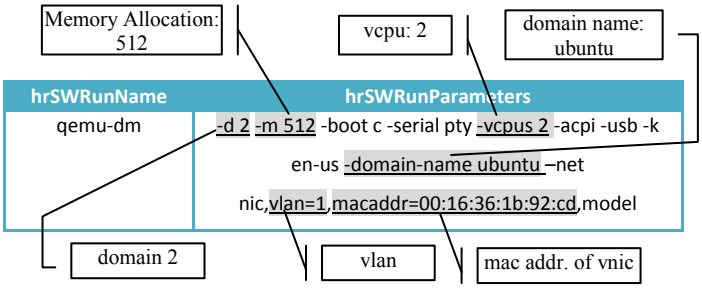

Figure 4 Interpretation of objects in *hrSWRunTable* (Xen)

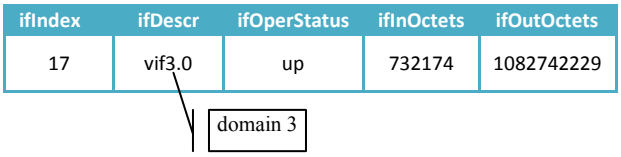

Figure 5 A virtual interface in *ifTable* (Xen)

2) KVM: Since KVM adopts full virtualization, each VM runs as a process in the host machince. Therefore, we can find all the running VMs in the hrSWRunTable table. Each VM runs with a process named "kvm", which appears in the hrSWRunName object of the hrSWRunTable table. From the corresponding *hrSWRunParameters* object, we can further

find VM name, memory allocation, and number of CPU cores configured for a VM, as shown in Figure 6. Moreover, the CPU time consumption and memory usage of the VM can also be obtained from *hrSWRunPerfCPU* and *hrSWRunPerfMem* respectively in the augmented *hrSWRunPerfTable*, as shown in Figure 7.

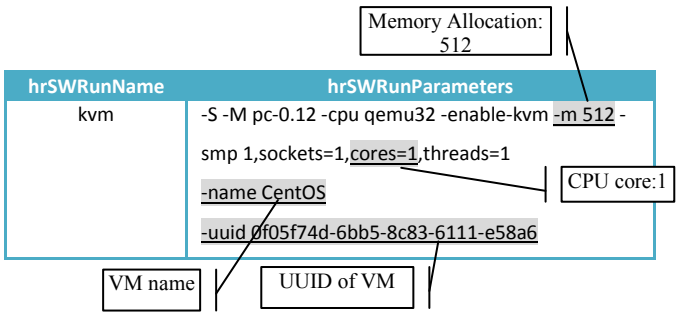

Figure 6 Interpretation of objects in *hrSWRunTable* (KVM)

hrSWRunPerfCPU	hrSWRunPerfMem
327366	110560

Figure 7 The augmented *hrSWRunPerfTable* table (KVM)

3) VMware Server: Similar to the above VM platforms, each VM instance runs as a process with name "VMware-vmx" in the host machine. Therefore, we can find the VM instances in the *hrSWRunTable*, as shown in Figure 8. Their CPU time consumption and memory usage can also be obtained from the *hrSWRunPerfTable*. Due to the lack of explicit parameters in the *hrSWRunParameters* object, we cannot obtain the other information of the VMs.

hrSWRunName	hrSWRunParameters
VMware-vmx.exe	-# "name=VMware Server; version=2.0.2; buildnumber=203138;.....

Figure 8 Interpretation of objects in *hrSWRunTable* (VMware Server)

IV. IMPLEMENTATION

This section presents our implementation of the proposed SNMP-based monitoring framework. Two major components of the implementation are NMS and AgentX. Both are developed in Java. In NMS, we use Tomcat as the web server, on which JSP server-side web pages are developed. In the web-based monitoring interface, the dojo JavaScript tool [18] is used to enhance the presentation of the monitored data. In the implementation of AgentX, we use J.AgentX[10] API to develop the extensible agent, and use libvirt[11] API to access the VM information provided by VM hypervisors. J.AgentX also provides an SNMP protocol entity, which receives requests and sends responses from/to NMS.

To demonstrate the superiority in managing heterogeneous VM environments, we develop the web-based management system for three different VM platforms, including VMware, Xen, and KVM, installed on two different OSs, MS Windows and Linux. As shown in Figure 9, three host machines with different VM platforms are installed in the intranet of our lab. These host machines can be monitored in three aspects, described as follows.

A. Host Machine Monitoring

Figure 9 shows the web interface for monitoring the host machines in a data center. Each host machine is represented by an icon, in which the VM hypervisor and OS are also displayed. After a host machine is selected, the manager can find the basic information of the host machine (see Figure 9). For more details of all the VM instances on the host machine, a VM table is also provided (see Figure 10). To further realize the overall network traffic of the host machine as well as each VM instance, the manager can browser the provided network interface table (see Figure 11).

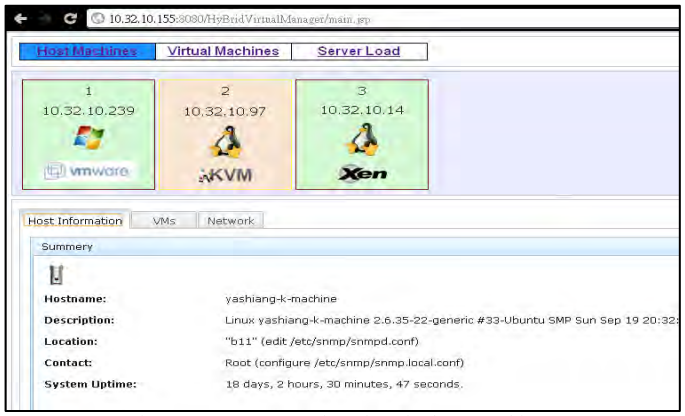

Figure 9 Management interface for host machines

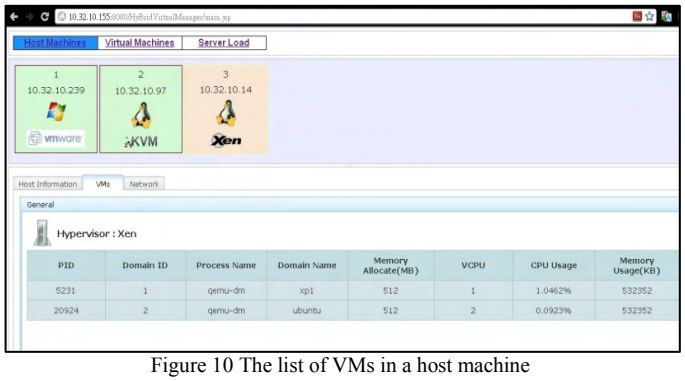

Figure 10 The list of VMs in a host machine

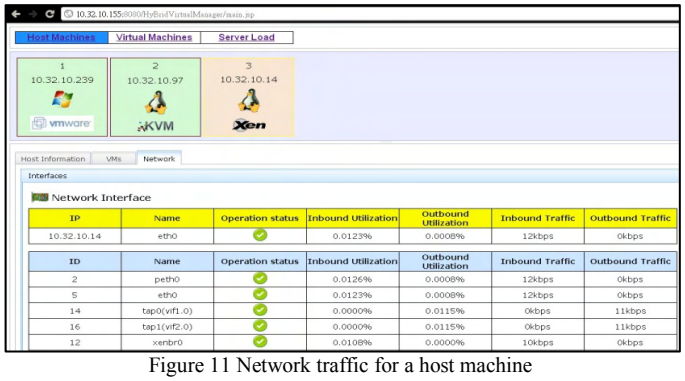

Figure 11 Network traffic for a host machine

B. *Virtual Machine Monitoring*

The manager of a data center may be more interested in monitoring all the VM instances regardless of where the VM instances are installed. Our implementation provides a monitoring interface from the aspect of VMs, as shown in Figure 12. All the VM instances across different VM platforms and host machines are listed in a single web interface. A consistent view of VM information is provided independent of VM platforms.

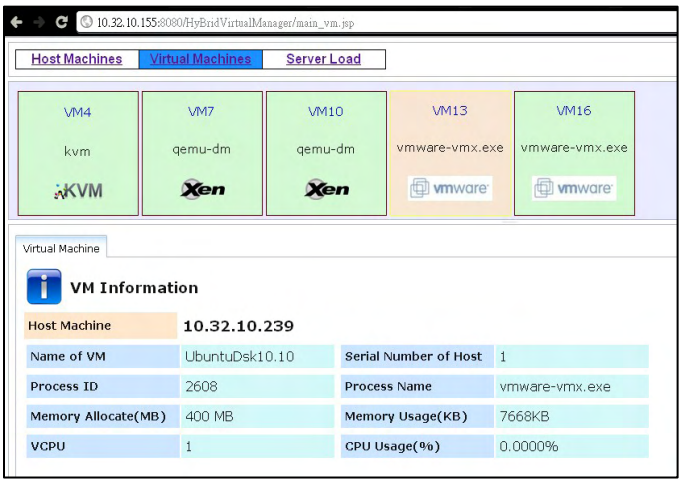

Figure 12 Monitoring of virtual machines

C. *Server Load Monitoring*

The manager of a data center may be more concerned about the balance of loads among the servers of the data center. Then, a new VM instance can be created on a server with a lower load. Our implementation provides a monitoring interface for the aspect of server loads. The host machines are sorted according to their CPU loads and network loads, as shown in Figure 13. Servers with loads over pre-defined thresholds are marked to indicate the status of a heavy load.

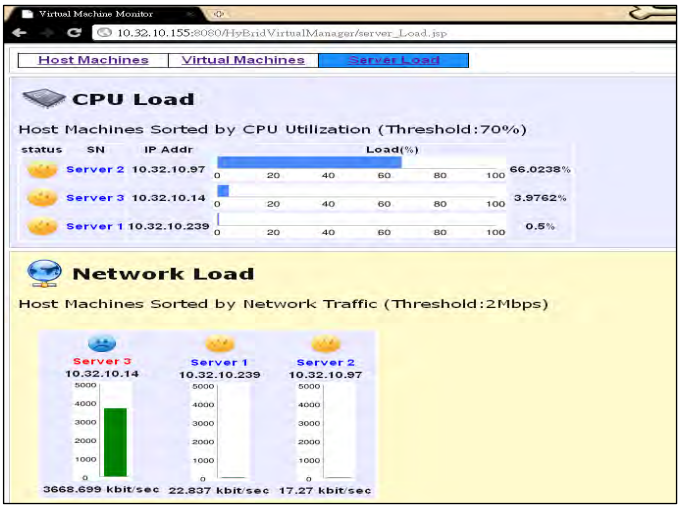

Figure 13 Monitoring of server loads

V. CONCLUSIONS

Currently, the management interfaces are different among VM platforms. There is a need for a standard approach in cloud management. This paper shows the possibility of SNMP for use in cloud management. SNMP has been used for years in managing network devices/hosts in TCP/IP-based networks. Essentially, the manger/agent paradigm of SNMP can be applied in the management of servers in a cloud. Furthermore, SNMP adopts an extensible information model, which allows the development of new MIB modules to manage new devices and services, including ones for cloud computing. In this paper, we have proposed an SNMP-based monitoring framework for heterogeneous VM platforms. By enabling the inherent SNMP agent and introducing an extensible SNMP agent, a host machine and its VM instances become manageable. By utilizing current standard MIBs and defining new MIBs, information for host and VM management becomes available. This paper has successfully demonstrated the use of SNMP in monitoring hosts and VMs. Our future work is to develop more comprehensive SNMP applications on the VM scheduling and dispatching. However, this requires the support of more standard MIB modules for VM management. This is another effort to be taken in the future.

REFERENCES

[1] "White Paper: Understanding Full Virtualization, Paravirtualization, and Hardware Assist", VMware, Nov. 10, 2007

[2] "VMware," http:// http://www.VMware.com/.

[3] "Xen" http://www.xen.org/.

[4] "KVM" http://www.linux-kvm.org/page/Main_Page

[5] "Amazon EC2," http://aws.amazon.com/ec2/.

[6] J. Case et al., "A Simple Network Management Protocol", IETF RFC 1157, May 1990.

[7] K. McCloghrie and M. Rose, "Management Information Base for Network Managementof TCP/IP-based internets:MIB-II", IETF RCF 1213, Mar. 1991

[8] P. Grillo and S. Waldbusser, "Host Resources MIB", IETF RFC 1514, Sept. 1993

[9] M. Daniele et al. "Agent Extensibility (AgentX) Protocol Version 1", IETF RFC 2741, Jan. 2000

[10] "J.AgentX", http://eden.dei.uc.pt/agentx/

[11] "libvirt VIRTUALIZATION API", http://libvirt.org/

[12] Edgar Magana, Antonio Astorga, Joan Serrat, Rafael Valle, "Monitoring of a Virtual Infrastructure Testbed", in Proc. 9th IEEE Latin-American Conference on Communications, Sept. 2009

[13] "SBLOMARS & BLOMERS", http://www.maps.upc.edu/sblomars/

[14] Lizhe Wang, Gregor von Laszewski, Dan Chen, Jie Tao, and Marcel Kunze, "Provide Virtual Machine Information for Grid Computing", IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, vol 40, pp. 1362 – 1374, Nov. 2010

[15] Kyrre M Begnum, "Managing Large Networks of Virtual Machines", in Proc. 20th Large Installation System Administration Conference, 2006

[16] Matthias Bolte, Michael Sievers, Georg Birkenheuer, Oliver Niehörster and André Brinkmann "Non-intrusive Virtualization Managementusing libvirt", in Proc. The Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden,Germany, Mar. 2010

[17] "libvirt-snmp", http://wiki.libvirt.org/page/Libvirt-snmp

[18] "dojo toolkit", http://dojotoolkit.org/