International Journal of scientific research and management (IJSRM)
||Volume||1||Issue||9 ||Pages|| 488-493 ||2013||
Website: www.ijsrm.in ISSN (e): 2321-3418

Virtual Resource management in Cloud Environment

Yaser Nemati [1],Faramarz Samsami [2],Ali Heydarzadegan[3]

1,3 : Department of Computer Engineering,Beyza Branch,Islamic Azad University,Beyza,Iran
2: Department of Computer Engineering, Kavar center,Islamic Azad University,Shiraz,Iran
Corresponding Author email: Nemati.Y.@Gmail.com

Abstract: Cloud computing is a new model for computing. The resource management tools are important compared to traditional distributed systems like grid computing. In order to better management the virtual resource in cloud, an event-driven push (EDP) monitoring model is proposed. EDP can provide comparatively information about usage and status of the resources. The model can simplify the communication between Master and Workers without missing the important events happened during the push interval. We test it by extend the Libvirt in monitoring of virtual CPU and memory.

Keywords: Resource Management, cloud computing, XEN

Introduction

Cloud computing is a new technology that goes computing and data from PCs and desktops into large scale data centers. The basic idea of cloud computing is to deliver infrastructure as services (IAS), software as service (SAS) and platform as service (PAS) over the Internet [1]. Virtualization technology is the key concept in cloud, and it can provide the cloud computing as a basic underlayment platform [2]. New computers have sufficient power to use virtualization for presenting many small virtual machines (VMs), each VM run a separate operating system instance. This research focuses on the virtual resource management in cloud computing. Cloud is not only a kind of new technology, but it can provide users with service through "pay-as-you-go" model. So it is important to manage the cloud resources and for this management a resource monitoring is required. The resource monitoring tools result to a feedback for cloud to resources discovery, allocation and scheduling [3]. Monitoring tools help cloud computing for pricing purpose also. However, since cloud computing is still in the starting stage but the VMs technology is under development, the monitoring tools for cloud computing based on virtual machine resource monitoring are far from its goals [4]. Some monitoring systems are designed for grid computing such as GridIce [5], Gangila[6] and GridView[7]. These monitoring systems are structured to monitor actual resources in systems and they can't monitor the highly virtualized resources. Another problem is cloud feature with no enough monitoring data and exact pricing model, and these are unacceptable for providers and users. In this paper, we extend a novel resource monitoring model from the platform layer to the virtualization layer.

On the platform, event-driven mechanism is used to optimize monitoring results and in virtualization, the goal is to get comprehensive and enough data for this goal we are used the existing Virtual machine monitor tools and interfaces and improved their weakness.

The reminder of the paper is organized as follow: In Section B, we introduce the background of this issue and related work; in Section C, our design architecture is depicted, as well design details are given of the event-driven model and the data collection. Model analysis is discussed in Section D; last we conclude the paper and discus the future works.

A. Related work

Ganglia[6] proposed by UC Berkley is a resource allocation and resource monitoring system that developed for clusters. It using UDP (User Datagram Protocol) or TCP (Transmission Control Protocol) connection for monitors the cluster. Ganglia has three modules: gmond, gmetad, and Ganglia PHP web front end. NWS[8] is a workload and distributed resource monitoring developed by UCSB. An NWS goal is to predict the computing performance for short period. Nagios[9] is a network and system monitoring software. It monitors servers and services that users specified. Nagios based on the alerting mechanism while Ganglia pay more care to collect measure and trace the system data. All of these systems play important roles in distributed systems and grid computing. However, in cloud computing, virtualization counts a lot: virtualized CPU, virtualized memory, virtualized network, etc. These features make it inappropriate and hinder to applying the above systems into cloud computing. Xen[10] and [11] is an open source virtual machine hypervisor from Cambridge. Its performance is close to the performance of real OS (operating system) that running on real hardware. Figure 1 is the architecture of Xen. It has three main components in Xen: the hypervisor, Domain 0 and Domain Us. The hypervisor is an interchange layer between the VMs and actual hardware. The hypervisor runs on Ring0, has the privilege of scheduling the hardware. It is responsible for the control of virtual resources and access of shared resources, and the scheduling of VMs. Domain 0 runs on Ring 1. It differs from the other Domain Us by the direct access of the hardware granted by the hypervisor. Thus the other domains resource allocation, operations have to pass through Domain 0. By doing this, Xen gains better resource isolation and security benefits. VirtManager [12] is a tool for managing the VMs based on Libvirt[13].

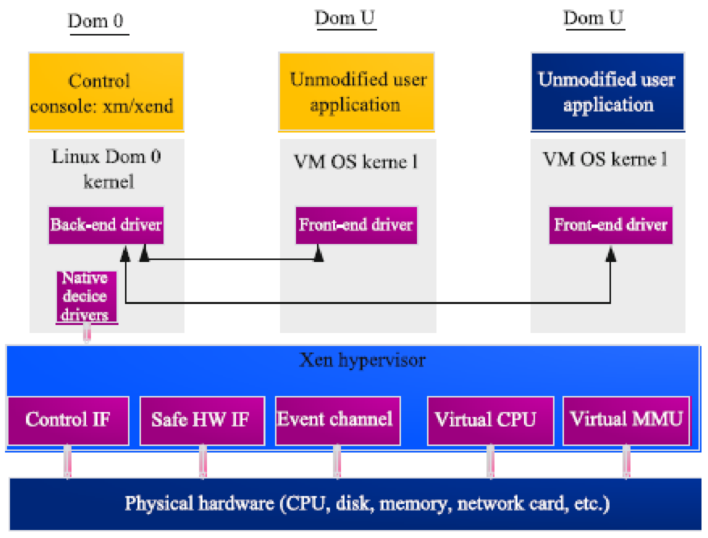

Fig.1 Architecture of Xen

Virt-Manager can monitor the VMs for performance and resource usage. It creates new VMs, allocate and change the VM resources. Virt-Manager is able to manage the VMs that created by Xen, Kvm[14] and Qemu[15]. Virt-Manager also has a list of managing tools such as: Virt install, Virt clone, Virt image and virt machine viewer. Although Libvirt can gather the information of the VMs such as the status, CPU usage rate, VCPU number, virtual memory, etc. But, it has also some problems. Firstly, it could not catch the sight of VCPU status. Because the VCPU information is stored at hypervisor layer, Virt-Manager cannot get the information of lower layers. Secondly, it also cannot get the memory situation in each virtual machine. The number it can get through Libvirt is just the size of allocated memory since the creation of the VMs. this is hard bottlenecks in virtualization technology. This will be a bonus to the overall performance. Thirdly, we can’t get virtual network and virtual disk load monitoring information in Virt-papers dedicated to alleviate the issue. The premise of the solutions is the accurate monitoring of it.

B. Design of the proposed resource monitoring

Two types of mode are often used for resource monitoring in distributed systems: push and pull [16]. However, as the different features between clouds with other distributed systems we cannot apply these monitoring types to cloud computing. So, we introduce these concepts into cloud by modification them. In the pull mode, the worker nodes are in a passive condition and the master node tells them the time that they have to send the monitoring data. In push, the worker nodes send the monitoring data to the master node without being called and the worker nodes are active in this situation.

There are two modes to estimate when to pull or push: event-driven or periodically. Thus four types of combination are available: periodical pull, periodical push, event-driven pull, and event-driven push. Each has its advantages and disadvantages.

In periodical pull mode, the choosing of time interval: short time pull aggravates network load; long interval cannot guarantee the data freshness, and may lose the capture of information during the pull message response time. Meanwhile, the transmission of pull request consumes time and bandwidth.

In periodical push mode, the same problems exist except for the last one. However when some emergency events happen, it would not send it immediately to the master node, that is what we do not expect to see. In event-driven pull mode, obviously it has worse coherency. There is difficulty in the master node to identify the “event”.

Lastly, in event-driven push mode, it has a quick response to the “event”, but the real-time coherency cannot be guaranteed. Over all, in our model, we adopt push and event-driven push modes both to each other. Here, we define this model as event-driven push (EDP). By means of it, we can assure the coherency as well as the event response time.

Figure 2 is the architecture of proposed monitoring model. In the master node, the data collector module is responsible for collection of monitoring data, and the GUI displays the monitoring data to user. In each worker node, within the Virtual machine manager (VMM), we choose the widely used Libvirt tool to monitor the status of virtual resources. The push module transfers data to master node under two considerations: It pushes periodically based on the time interval (the time has been set before); in emergency case (when event trigger) it pushes the urging data to the master node immediately.

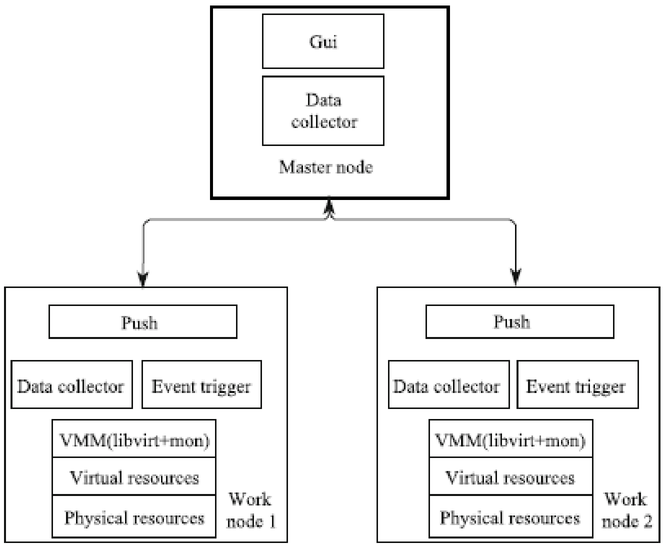

Fig.2 Architecture of proposed monitoring model

C. EDP Algorithm

As Fig.3 shows there are two key points in EDP algorithm: the value of push interval (T) and the definition of "event". The former determines by the monitoring status: a large value of T may result in the "freshness" of information, while a short T cause to data transmission pressure. In [14], [17] discussed the value of T; experiment results showed that 1 second is an appropriate value. So, we set the T to 1 s.

We select event as a threshold which represents the load fraction of CPU and memory. The nodes' resource load vector has been defined as:

R = (Ucpu, Umem)

(1) Here R is the node's resource load vector, Ucpu, Umem represent the rate of resource utilization of CPU and memory respectively.

```
For each node n
        While (new event)
                If (gettimediff() == T)
                        D= Monitor_Data()
                        Send(D,Master)
                        Timer_Reset()
                Else Waite (New_Interval)
                End If
                Send (D, Master)
        End While
End Procedure
```

Fig.3 EDP algorithm

For example, if the CPU utilization is over 85% or the memory utilization rate is over 85%, or both of them are less than 15%, the event will be trigger, and data has been send to master regardless of the T. So, n the work nodes aren't too busy or too idle.

D. Model analysis

The aims of design of any system are scalability, efficiency, and accuracy. In this section these aspects of the proposed model are studied. From scalability perspective, EDP model is inherently scalable. The cloud computing is heterogeneous. There are a lot of categories of cloud computing such as: private, public and hybrid. However, the virtualization is the base cloud computing technology. Proposed model based on the virtualization level and cover the system heterogeneity. We monitor the status not only solely under specific configuration but also we can apply it to general platforms. In efficiency, DEP model guarantee balances the workload and freshness of data. The Suitable value of T can guarantee the obtaining of monitoring data without imposing too much overhead to the system. With the consideration of efficiency, EDP can adopt threshold control since it will not occupy the computing resource (CPU). Obtaining enough monitoring data is a basic role in any monitoring system. Libvirt gains an upper hand in the scalability since it can be used under Xen, KVM or QEMU. However, the monitoring data it gathers are not adequate enough, especially the virtual resource using status. Xen, itself, provides some tools for collecting the monitoring data, such as: xm top, Xenstore, XenMon [18]. These tools work on the low layers and they can get more accurate data. But the problem is that they can only be used in Xen. EDP uses the Libvirt tool; hence it carries forward the scalability.

E. Conclusions and future work

The rising of cloud computing paradigm has a lot of new problems to be solved. The resource management tool is one of the challenging. In order to manage the virtual resource the monitoring tools are vital in cloud computing. So, to provide comparatively sound monitoring information to users and other components in cloud computing, we proposed EDP monitoring model. It takes advantage of the event-driven and push mechanism, and simplify the communication between master and worker nodes without missing the important events that happened during the push interval. This model is able to capture the virtual resources dynamic usage status very well, compared to other methods such as Libvirt tools only. It does not consume much computing resource and provide more adequate information. Our further goal is to be able to provide this information not only to users but also to the other main components in cloud computing, such as: load balancing tools and pricing. Based on the monitoring data, we can develop easy the ways to reallocate resources and load balance the overload nodes.

F. References

[1] L. Wang, G. Von Laszewski, A. Younge, and X. He, “Cloud Computing : a Perspective Study,” vol. 28, pp. 137–146, 2010.

[2] A. Kaur, “A survey on cloud computing and its techniques,” vol. 1, no. 3, pp. 93–97, 2012.

[3] M. Sharifi, H. Salimi, and M. Najafzadeh, “Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques,” 2011.

[4] H. A. N. Fang-fang and L. I. Qing, “Virtual resource monitoring in cloud computing,” *Comput. Eng.*, vol. 15, no. 200802800007, pp. 381–385, 2011.

[5] S. Andreozzi, N. De Bortoli, and S. Fantinel, “GridICE: a monitoring service for Grid systems,” … *Comput. Syst.*, 2005.

[6] M. Massie, B. Chun, and D. Culler, “The ganglia distributed monitoring system: design, implementation, and experience,” *Parallel Comput.*, 2004.

[7] N. Guangbao, M. Jie, and L. Bo, “GridView: A dynamic and visual grid monitoring system,” … *Comput. Grid Asia Pacific …*, 2004.

[8] V. Poladian and A. Arlan, “Leveraging resource prediction for anticipatory dynamic configuration,” *Self-Adaptive …*, 2007.

[9] E. Imamagic and D. Dobrenic, “Grid infrastructure monitoring system based on Nagios,” *Proc. 2007 Work. Grid …*, 2007.

[10] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the Art of Virtualization Categories and Subject Descriptors,” *Memory*, vol. 59, no. C, pp. 164–177, 2003.

[11] L. Cherkasova, D. Gupta, and A. Vahdat, “Comparison of the three CPU schedulers in Xen,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 2, pp. 42–51, Sep. 2007.

[12] M. Hammel, “Managing KVM deployments with Virt-Manager,” *Linux J.*, 2011.

[13] R. Hat, “libvirt: The virtualization API,” 2012.

[14] A. KIVITY, “KVM: The Linux Virtual Machine Monitor,” *Proc. Linux Symp. Ottawa, Ontario, …*, 2007.

[15] S. Shen, S. Lee, and C. Chen, “Full system simulation with QEMU: An approach to multi-view 3D GPU design,” *Circuits Syst. (ISCAS), …*, 2010.

[16] J. H. III and J. Brown, “From push to pull: emerging models for mobilizing resources,” *J. Serv. Sci.*, 2011.

[17] V. C. Emeakaroha, S. Paulo, and C. A. F. De Rose, “CASViD : Application Level Monitoring for SLA Violation Detection in Clouds.”

[18] G. Diwaker, G. Rob, and C. Ludmila, “Xenmon: Qos monitoring and performance profiling tool,” 2005.