2015 IEEE International Conference on Smart City/SocialCom/SustainCom together with DataCom 2015 and SC2 2015

A Performance Evaluation Model for Virtual Servers in KVM-based Virtualized System

Jing Yang
School of Computer Science and Engineering
Beihang University
Beijing, China
Email: yang_ann@buaa.edu.cn

Yuqing Lan
School of Computer Science and Engineering
Beihang University
Beijing, China
Email: lanyuqing@buaa.edu.cn

***Abstract*—According to the statistics, there is low resource utilization and high energy consumption in traditional servers. To reduce the cost, more and more companies begin to build virtual servers. Sever virtualization implements the mapping from virtual resources to physical resources and deal with resource contention among all VMs. Because of complexity of virtualized server systems, it is necessary to evaluate the performance of virtual servers. In this paper, we present the major performance-influencing factors in KVM virtualization. Based on factors, then we propose a performance evaluation model based on queuing networking model. In the end, we give a case study to illustrate the use of performance evaluation model on specific KVM virtualized environment and verify the correctness of the model.**

***Keywords*-server virtualization; KVM; queuing network model; performance evaluation;**

I. Introduction

In recent years, virtualization technology has been widely applied in data centers. The CPU utilization of traditional servers ranges from 5% to 40% [1]. Besides, the Gartner report shows, underutilized servers have the high energy consumption, and virtualization can reduce energy use as high as 80%, and save the space consumption up to 85% [2]. In order to resolve the contradiction between the number of servers and resource utilization, many service providers adopt virtualization to integrate server resources and save costs.

Virtualization technology isolates one physical server into several virtual servers, so it can maximize the resource utilization in physical servers. However, server integration implemented by server virtualization increases complexity and dynamics of the system. Whether applications in virtual servers can or can't meet user demand and how to avoid performance bottlenecks are emphases in area of virtualized servers.

Now, the number of Linux servers is growing rapidly and new servers will be up to 72 million in 2017. It provides a great opportunity for use of KVM that is part of Linux kernel. The IDC's tracker report of server virtualization shows, since 2011, the number of deployed KVM is over 278,000 and a GAGR is nearly 42% [3], so it is necessary to study the performance of KVM virtual servers. However, there are few related performance studies and lack elaborated analysis of KVM performance.

In this study, firstly we analyze the performance metrics of virtual servers, and then, based on metrics, we study influence factors of performance combining with KVM implementation. Finally, we build the performance model by queuing network model that can calculate performance of virtual server by practical data produced during the server use. This method can avoid setting up complicated benchmark environment and improve the efficiency of performance evaluation. On the other hand, many performance benchmark tools only give the final evaluation results. This method we proposed can get performance of main parts of virtual server. It is helpful to find the performance bottleneck for service providers.

The remainder of this paper is organized as follows. Section 2 summarizes related research works. Section 3 presents the performance metric and performance-influencing factors of KVM virtualized environment. Section 4 analyzes the architecture of KVM and describes how to get queuing network model based on virtualization implementation of KVM. Section 5 illustrates a case study, including experimental environments and detailed calculation procedures of performance evaluation. Section 6 concludes our research and discusses the future work.

II. Related Work

The method used for performance evaluation of virtual servers can be classified into the simulation method and analytical method, so the study mainly focuses on these two aspects.

Simulation method: This method needs to build practical application environments. The benchmark produces requests for virtual servers and gives the performance results. Huber et al. summarized performance-influencing factors showed in presented virtualization platforms, and designed related experiments by benchmarks to quantitative analysis influence factors in Citrix XenServer5.5 [4]. In [5], Ibidokun et al. conducted an experiment in Oracle virtualized servers by

978-1-5090-1893-2/15 $31.00 © 2015 IEEE
DOI 10.1109/SmartCity.2015.49

benchmarks. According to data obtained from experiments, they analyzed the scalability of the server and operability for high workloads. In [6], on the basis of [4], they made further study in VMware ESX 4.0. They built the simple mathematical performance prediction model by comparing data obtained in VMware ESX 4.0 and Citrix XneServer5.5. In [7], Hui et al. analyzed performance characteristics and problems confronted with server virtualization based on Xen virtualization platform by SPECvirt_sc 2010. And they gave an improvement method.

Analytical method: This method uses traditional prediction model, such as queuing theory, to abstract server system for building performance model, and combines with server workload characteristics to evaluate server performance. In [8], researchers analyzed performance parameters using queuing network mode at the the server virtualization scenario that had two virtualized servers. They also gave an example to explain how to calculate performance metrics. In [9], Sajib et al. studied key parameters involved in application performance model of virtualized servers. And according to experiments, they proposed an application performance prediction model based on the artificial neural network. In [10], RahimiZadeh et al. also used queuing network model to build performance model for Xen virtualization platform in different workloads. In the end of paper, they validated their methodology by an instance with the benchmark.

The main advantage of the simulation method is that they can get the performance metrics from end to end, and because they simulate the real usage scenarios of virtual servers, so the results obtained by simulation method are accurately. However the configuration of benchmarks used in simulation method is very complex. Besides some of benchmarks can only be used in specific virtualization environments. The analytical method has better evaluation efficiency than simulation method, and can find the system bottleneck easily by getting performance metrics of main components of the virtual system. On the other hand, the analytical method can reduce the cost for users. Although the results obtained by the analytical method may be imprecise, it can improve accuracy by detailed feature analysis of virtual servers and reasonable abstraction. Based on the above analysis, we choose the analytical method to evaluate the performance of KVM virtual machines.

III. Performance-Influencing Factors for KVM

The server virtualization can divide a physical server into several independent and isolated spaces, so it makes multi-servers run on a physical server. At first, service level agreement (SLA) was used for restriction Internet service providers. Now, it is also used for management multiple IT services. The quality of service (QoS) in SLA specifies the service quality. In [11], scholars analyzed QoS measures of network applications. The response time and throughput are necessary metrics among measures. At the user level, the performance of virtual servers is reflected by application performance running on it. Hence response time and throughput are also main performance metrics of virtual servers.

A. Performance-Influencing Factors

The application performance in virtualized environment is related to interactions between layers of the virtualized system. Virtual machine behaviors, operations of virtual machine monitor (VMM) and the feature of workloads are main influence factors of application performances in virtualized systems. VMM is a middle layer between the virtual machine and physical resources. It implements the mapping operation of virtual resources to physical resources, so resources allocation strategy of VMM has important influence on server virtualization performance. On the other hand, the physical resources in virtualized system are shared among VMs. Therefore operations of virtual machines may affect each other and degrade performance of virtual servers.

The different workloads running in virtual servers involve different resources, and they have different performance overhead. For example, due to the support of physical hardware, CPU virtualization performance overhead is lower than I/O virtualization, so different workload types have different influence on performance of virtual servers. Huber et al. studied main virtualization solution and summarized performance-influencing factors of them [4]. Besides, they proposed a hierarchical feature model. In this model, the influence factors are divided into three types, virtualization type, resource management configuration and workload type.

B. the Hierarchical Model of Influence Factors

Now there are two kind of virtualization types: full virtualization and paravirtualization. In full virtualization type, OS can be used without modification. Paravirtualization modifies the OS kernel to replace nonvirtualizable instructions in management interface. In the X86 architecture, performance of full virtualization is generally better than paravirtualization [12]. KVM is part of Linux kernel since 2.6.20. It is hardware assisted full virtualization.

KVM implements CPU virtualization by Intel VT-x or AMD-V. Intel virtualization technology provides new CPU execution modes: VMX root mode and VMX non-root mode. These modes have corresponding user mode and kernel mode. Therefore, the efficiency of CPU virtualization is high. KVM relies on QEMU, a software-based emulator, to implement hardware emulation. It needs many times switch contexts, so its efficiency is low. At present, KVM adopts virtio to reduce the number of switch contexts. The performance overhead between CPU intensive operations and I/O intensive operations is different. In addition, for memory virtualization, KVM uses shadow page tables to map virtual memory addresses in virtual servers to physical

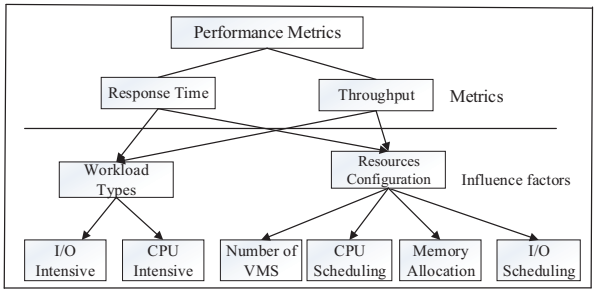

Figure 1. the performance metrics & influence factors.

memory addresses. KVM needs to maintain a set of shadow page tables for all processes. Shadow page tables loaded into memory management unit can be used for virtual servers directly. Intel and AMD also have implemented hardware assisted memory virtualization to improve memory virtualization efficiency.

The number of virtual machines in the physical server has a direct effect on accessing shared resources. It affects how to process requests made by virtual machines. According to above analysis, the performance metrics and its influence factors in KVM virtualized environment can be showed in the Figure 1.

IV. THE PERFORMANCE MODEL OF KVM

We build performance model of the KVM virtual machines based on queuing network model by the analytical method.

A. Queuing Network Model

Queuing network model is applied to analyze and present resources usage. In queuing network model, both hardware resources and software resources are regarded as service centers. It is interconnected service centers. Each service center has a waiting queue and scheduling algorithm. The elements of the queuing network model can be divided into three parts: service centers, customers and network topology. The customer is workflow passing queuing network. A customer class may have multiple workloads. The requests in a workload class have same features. According to a number of requests that can be fixed or variable, they can be divided into closed queuing network model and open queuing network model. In the actual network environment, the number of requests is variable, so we will build the open queuing network model.

In open queuing network model, a workload class can be characterized with workload intensity and workload service demands on resources. Workload intensity is arrival rate of workload requests. Workload service demand is the total time spent by workload in given resources.

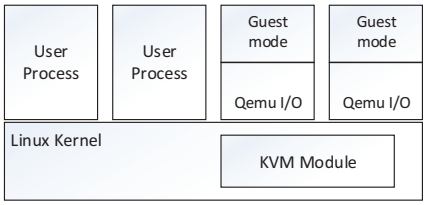

Figure 2. the KVM architecture.

Let K be the number of total service centers in the model with single workload class. The arrival rate of workload requests is λ and the workload service demand is D_m at service center m. V_m is the numbers of total service requests at service center m. The outputs of this model [13]are in the below:

a. The throughput at service center m is:

$$X_m = \lambda * V_m;$$

b. The service demand at service center m is:

$$D_m = \frac{U_m}{X_m};$$

where U_m is the resource utilization at service center m.

c. The response time at service center m is:

$$R_m = \begin{cases} D_m, & \text{IS} \\ D_m[1 + A_m(\lambda)], & \text{FCFS, PS, LCFSPR.} \end{cases}$$

B. Modeling KVM-based Virtual Servers

Based on analysis of performance-influencing factor in 3.2 section, we study the scheduling and implementation of virtualized resources. KVM turns Linux kernel into supervisory program of the bare machine. The process scheduling and memory management of Linux kernel can be applied for KVM virtualized system. The standard kernel uses user mode and kernel mode. KVM introduces the third mode: guest mode. In guest mode, they have own user mode and kernel mode, hence they can run the operating system. The process in guest mode executes native non-sensitive instructions. In user mode, sensitive instructions are executed. Kernel model is responsible for capturing special instructions and performing switches between guest mode and user mode. Figure 2 shows the architecture of KVM [14]:

The guest model runs in VMX non-root mode of CPU. If the guest operating system attempts to perform sensitive instructions, CPU will be changed into VMX root mode. And they may rely on QEMU to execute sensitive instructions. The state-transition in CPU of Intel virtualization technology is showed in the Figure 3 [15]:

Virtio is standard I/O paravirtualization device of Linux. And it becomes a part of Linux kernel since 2.6.25. Virtio is

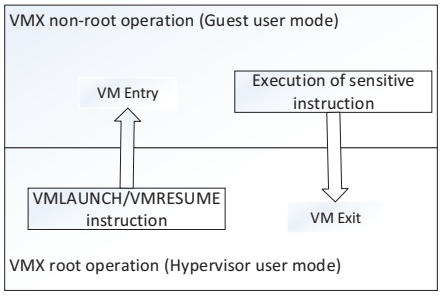

Figure 3. the architecture of Intel VT-x.

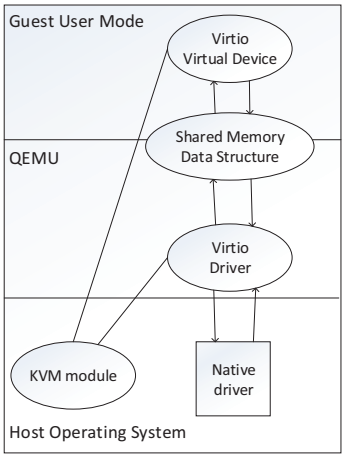

Figure 4. the architecture of Virtio.

divided into front-end driver and back-end driver. Front-end driver is driver module in guest operating system. Back-end driver is implemented in QEMU. Front-end driver and back-end driver interact with each other by KVM module that can access the memory of virtual machines, like in Figure 4. There is the shared memory between two drivers that used for store related data. By this way, each data block only performs once VMExit instruction.

In the virtual server, when applications need to read disks, guest operating system will write requests into shared memory by front-end driver. Firstly, front-end driver calculated needed buffer. And then, it adds requests and its description information into shared memory. Meanwhile CPU executes VMExit command. KVM module parses the reason why VMExit command is executed. After they make sure that it is due to I/O operations, they will quit root mode of CPU. In user mode, the device of virtio parses data in shared memory to get operation instructions and objects. Subsequently, they call corresponding system calls to finish I/O requests. Besides, they store results in shared memory [16]. In the end, back-end driver calls KVM to execute VMEntry. In guest mode, the front-end driver gets results

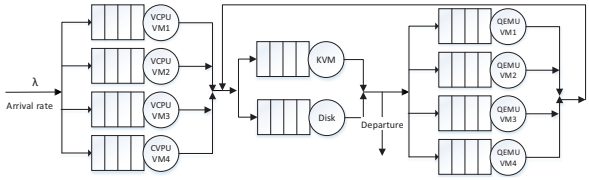

Figure 5. queuing network model.

from shared memory and submits to applications.

In KVM virtualized environment, user requests reaching virtual server are handled by virtualized CPU that is the physical CPU in non-root mode. If they involve I/O operations, KVM kernel module exits back to user mode. The QEMU process running in user mode calls system call to get physical resources. Based on the above analysis, we propose a queuing network model of KVM.

Each virtual machine corresponds to one QEMU process, including one or more CPU and I/O threads. In Figure 5, when user requests reach to the virtual server, the CPU thread running non-root mode handles them, and residence time is R_{vcpu}. If it involves sensitive operations, the CPU will enter into root mode, and KVM invokes relevant functions to handle it. The residence time is R_{kvm}. Because the virtual CPU and KVM are two different modes of the physical CPU, the residence time can be expression as: R_{vcpu}+R_{kvm}=R_{cpu}. If the sensitive operations are I/O operations, they will "read" or "write" shared memory. It can be equal to read or write virtual disks. Its residence time is R_{vdis}. After CPU turns into root mode, KVM notifies I/O threads to handle requests. I/O threads call system calls to read and write hard disk and its residence time is R_{disk}. In the I/O operations, response time is R_{req}=R_{cpu}+R_{vdisk}+R_{disk}. The throughput of this model is total requests handled by virtualized system per second.

V. A Case Study

In this section, we illustrate the specific performance calculation method based on queuing network model in Section 4.2. In addition, to verify the correctness of method, we conducted two experiments and compared the experimental results.

A. Experimental Environment Setup

We experimented our performance calculation method on two physical servers. In each server, CentOS 7 was the native and guest operating system. Both of host was loaded KVM module and installed virt-manager to manager virtual machines. There were two virtual machines in each physical machine, as web server and database server. Another standalone computer, as client, initiated requests by Httperf and Sysbench tools for virtual servers. The only difference of experimental environments was physical resources. One physical server was Dell optiplex 780 machine with Intel

core 2 Quad CPU, 2.66GHZ. It had 4GB main memory and 500GB SATA HDD. Another server had Intel CPU with 2 core, 2GB main memory and 500GB SATA HDD.

B. the Performance Evaluation Method

In web virtual server, we chose prefork worker mode for Apache. At the beginning, there were 5 httpd processes waiting for user requests. Each process had one thread. If they were used and less than specified minimum, Apache would generate new processes. Httperf produced 5 connection requests per second and each them had one request. MySQL as database in database server opened general query log function. Before the test, Sysbench created a test table with 1,000,000 records in server. And it generated 16 threads to access database producing multiple transaction requests. Besides, each transaction included multiple SQL statements.

We obtained CPU time spent by httpd in the output file by top command. Firstly, we counted processes related Apache for web services in total processes. And then we summed the time spent on gathered processes. That was the time that Apache spent on CPU during the experiment. Finally, we could get the average CPU time spent by each user request, as following:

$$T_{cpu,web} = T_{cpu,pid_i} + T_{cpu,pid_j} + \cdots + T_{cpu,pid_k} \quad (1)$$

$$R_{cpu,web} = \frac{T_{cpu,web}}{\lambda_{total}} \quad (2)$$

Where T_{cpu,pid_i} is the CPU time spent by process i. On the other hand, we can get threads calling on disks by iotop command. So we can gather read and write requests. And we can get the process time in sys/block directory. The utilization of disks can be calculated as follows.

$$U_{disk,total} = \frac{T_{disk}}{T} \quad (3)$$

$$Read_{disk,total} = Read^{t_1}_{disk,total} - Read^{t_2}_{disk,total} \quad (4)$$

$$Write_{disk,total} = Write^{t_1}_{disk,total} - Write^{t_2}_{disk,total} \quad (5)$$

$$IO_{disk,total} = Read_{disk,total} + Write_{disk,total} \quad (6)$$

$$IO_{disk,req} = Read_{disk,tid_i} + \cdots + Read_{disk,tid_k} + Write_{disk,tid_i} + \cdots + Write_{disk,tid_k} \quad (7)$$

$$U_{disk,req} = \frac{IO_{disk,req}}{IO_{disk,total}} * U_{disk,total} \quad (8)$$

Where T is total time spent in the experiment. $Read^{t_1}_{disk,total}$ and $Write^{t_1}_{disk,total}$ are amounts of disk reads and writes performed before experiments. $Read^{t_2}_{disk,total}$ and $Write^{t_2}_{disk,total}$ are amounts of disk reads and writes after experiments. According to the model, we can get:

$$D_{disk,req} = \frac{U_{disk,req}}{X_{disk,total}} \quad (9)$$

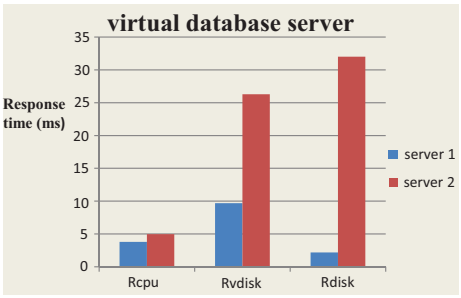

Figure 6. the web server performance comparison.

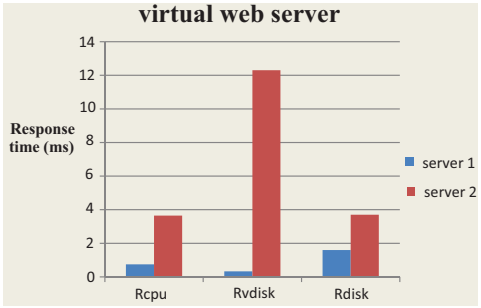

Figure 7. the web server performance comparison.

Table I
PERFORMANCE OF THE FIRST PHYSICAL SERVER

virtual servers	R_{cpu}	R_{vdisk}	R_{disk}	X
web server	0.75ms	0.33ms	1.6ms	5req/s
database server	3.8ms	9.7ms	2.2ms	77.5req/s

Where $D_{disk,req}$ is time spent on disk by web requests. And $X_{disk,req}$ is throughput of disk during the experiment. The throughput and response time of disk can be obtained using the following equations.

$$X_{req} = \frac{\lambda_{total}}{T}, R_{disk,req} = \frac{D_{disk,req}}{1 - U_{disk,req}} \quad (10)$$

The calculation method of response time in virtual disk is same with physical disk. And the performance evaluation method used in web server can also be used in the database server.

C. Experimental Results

We collected data produced during the experiment at process level to calculate response time and throughput. Before the experiment, we gathered initial workload information by top and iotop command. Besides, we got the initial disk operation information as well as activated time in sys/block directory. And then we started up Httperf and Sysbench in client to send requests for server. Meanwhile the top and iotop command were activated on servers. When experiments were end, these commands were stopped. According to calculation method, we can get response time

Table II
PERFORMANCE OF THE SECOND PHYSICAL SERVER

virtual servers	R_{cpu}	R_{vdisk}	R_{disk}	X
web server	3.65ms	12.3ms	3.7ms	5req/s
database server	4.97ms	26.3ms	32ms	54.7req/s

and throughput with collection data. In Figure 7 and Figure 6, we can find that the performance of virtual servers in the first physical server is better than in the second physical server. We can also get same conclusion by comparing physical configuration of the first server to another that can verify the correctness of our performance evaluation method. Besides, as shown in table I and table II, the performance of virtual web server is better than the virtual database server that involves more I/O operations.

VI. CONCLUSION

In this paper, we analyze performance metrics and performance-influencing factors of virtual servers based on KVM. We present a hierarchical metrics structure. Based on metrics, we propose a queuing network model of KVM. Furthermore, we give the specific performance calculation method by a case study. The experimental results obtained from two experiments also verify the correctness of performance evaluation model we proposed.

However, although we give the performance-influencing factors, we don't illustrate specific influence of each factor for performance metrics. We plan to study the quantitative effect of influence factors for performance to help improve performance.

REFERENCES

[1] A. Corporation, "Server virtualization: A step toward cost efficiency and business agility," avanade corporate headquarters, Tech. Rep., 2009.

[2] G. Corporation, "Energy saving via virtualization: Green it on a budget," Gartner corporation, Tech. Rep., 2008.

[3] G. Chen, "Kvm c open source virtualization for the enterprise and openstack clouds," International Data Corporation, Tech. Rep., 2014.

[4] N. Huber, M. Von Quast, F. Brosig, and S. Kounev, "Analysis of the performance-influencing factors of virtualization platforms," in *On the Move to Meaningful Internet Systems, OTM 2010*. Springer, 2010, pp. 811–828.

[5] I. E. Tope, P. Zavarsky, R. Ruhl, and D. Lindskog, "Performance evaluation of oracle vm server virtualization software 64 bit linux environment," in *Security Measurements and Metrics (Metrisec), 2011 Third International Workshop on*. IEEE, 2011, pp. 51–57.

[6] N. Huber, M. von Quast, F. Brosig, M. Hauck, and S. Kounev, "A method for experimental analysis and modeling of virtualization performance overhead," in *Cloud Computing and Services Science*. Springer, 2012, pp. 353–370.

[7] H. Lv, Y. Dong, J. Duan, and K. Tian, "Virtualization challenges: a view from server consolidation perspective," in *ACM SIGPLAN Notices*, vol. 47, no. 7. ACM, 2012, pp. 15–26.

[8] D. A. Menascé, "Virtualization: Concepts, applications, and performance modeling," in *Int. CMG Conference*, 2005, pp. 407–414.

[9] S. Kundu, R. Rangaswami, K. Dutta, and M. Zhao, "Application performance modeling in a virtualized environment," in *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*. IEEE, 2010, pp. 1–10.

[10] K. RahimiZadeh, R. Nasiri Gerde, M. AnaLoui, and P. Kabiri, "Performance evaluation of web server workloads in xen-based virtualized computer system: analytical modeling and experimental validation," *Concurrency and Computation: Practice and Experience*, 2015.

[11] Y. Chen, T. Farley, and N. Ye, "Qos requirements of network applications on the internet," *Information, Knowledge, Systems Management*, vol. 4, no. 1, pp. 55–76, 2004.

[12] D. P.Ahuja, "Full and para virtualization," University Of North Florida, Tech. Rep., 2010.

[13] R. Jain, "Mean value analysis and related techniques and related techniques," Washington University, Tech. Rep., 2008.

[14] T. Hirt, "Kvm-the kernel-based virtual machine," *Red Hat Inc*, 2010.

[15] Y. Goto, "Kernel-based virtual machine technology," *Fujitsu Scientific and Technical Journal*, vol. 47, pp. 362–368, 2011.

[16] S. Grinberg and S. Weiss, "Architectural virtualization extensions: A systems perspective," *Computer Science Review*, vol. 6, no. 5, pp. 209–224, 2012.