Performance Evaluation of the CPU Scheduler in XEN

Xianghua Xu[1], Peipei Shan[2], Jian Wan, Yucheng Jiang
Grid and Service Computing Lab
School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
[1]xhxu@hdu.edu.cn, [2]juzi_7@163.com

***Abstract*—Virtual machines with advances in software and architectural support have emerged as the basis for enterprises to allocate resources recently. One main benefit of virtual machine is server consolidation. However, flexible and complex consolidation results in some unpredictable performance problems and introduces new requirements, such as proper configurations for scheduler and reasonable arrangements for services. In this paper, we present a comparative performance evaluation of several different typical application consolidations in different configurations of scheduler parameters (under Credit scheduler) in Xen. We analyze the impact of the configurations of scheduler and mutual impact between VMs which run different types of applications, present proposals for users to adopt an efficient scheduler configuration in applying virtualization, and offer insight into the relationship of performance and scheduler parameters to motivate future innovation in virtualization.**

Keywords- scheduling parameters; performance evaluation; Xen

I. INTRODUCTION

Recent advancements in hardware and software support for virtualization have made a resurgence period for diverse uses of virtual machine. The promotion of high performance of hardware and the urgent need for server consolidation motivate virtualization technology application. Many organizations found that lots of servers only run single tasks with much spare resource. With virtualization, servers could be consolidated into a single machine in a security environment. However, service consolidation and flexibility produce new problems, such as unpredictable application performance, unfair resource scheduling and isolation for service, and also introduce new requirements, such as proper configurations for scheduler and reasonable arrangements for services.

Xen is an open-source virtual machine monitor, on account of its open-source and excellent performance, has been widely adopted by enterprises. As the scheduling policy in Xen is always borrowed from traditional operating system, it is bound to cause some unpredictable behaviors and severe performance problems for processes running in the VM, particularly, for I/O applications. Just because of this, network virtualization is not much attracted by those I/O intensive applications.

While other works studied the fairness of resource distribution between different types of applications concurrently running in domains, and the influence made by VMM scheduling on one type of applications performance, our study focus on the impact of different scheduler parameters configurations on different types of services concurrently executing in different domains, and demonstrate different arrangements of applications and effective scheduler parameters configurations can get improved performance for applications.

The rest of this paper is organized as follows. In section 2, we discuss the related work. Section 3 provides a brief background introduction. Section 4 describes several questions motivate us do the work. Section 5 presents the experimental methodology, and then in Section 6, it discusses experimental results. Finally, we make a conclusion in Section 7.

II. RELATED WORK

As mentioned earlier, the fairness of resource distribution between different types of application concurrently running in several domains has been studied in [6], using Xen, Diego Ongaro et al. figured that the fairness for I/O intensive workloads had not achieved the same level as the compute intensive workloads, also to a certain extent they proposed some extensions to VMM scheduling to improve I/O performance.

Divaker Gupta et al in [7] studied the resource consumption in the hypervisor on behalf of individual VMs for I/O processing, and introduced a SEDF-DC scheduler which derived from SEDF scheduler for Xen, CPU overhead in driver domain not accounted was significant for I/O intensive applications.

Ludmila Cherkasova et al in [8] compared three proportional-share CPU schedulers in Xen, they used a small suite of benchmarks to analyze the influence of the scheduling parameters make on the performance of a single application running in a domain, also did some measurements in the CPU allocation errors. In contrast, this paper focus on how different applications in different domains are impacted by the scheduler parameters, and particularly the configuration relationship among DomUs rather than only the relationship between Domain 0 and domain U.

978-0-7695-3494-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ISISE.2008.123

IEEE computer society

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:14:01 UTC from IEEE Xplore. Restrictions apply.

III. BACKGROUND

A. VMM

Xen is a Virtual Machine Monitor based on open source, including two components: hypervisor and driver domain, the latter is now in domain0 typically. The hypervisor is an abstract layer between the virtual machines and underlying physical hardware. Different to the original model [7, 9], drivers are isolated from hypervisor to become the driver domain, to control I/O access by the guest domains. Each virtual machine has a front-end driver, to communicate with the back-end driver in the driver domain, and then connect to the physical NIC through an Ethernet bridge.

B. Xen's schedulers

Xen offers two schedulers schemes currently, SEDF (Simple Earliest Deadline First) and Credit schedulers.

In the current version schedulers, Credit [5] is the default scheduler in Xen (since version 3.0) while SEDF is gradually phased out. Under Credit scheduler, each CPU manages a run queue of runnable vCPUs. This queue CPU manages is in ordered by vCPU's priority which can be in one of two states: OVER and UNDER. OVER represents the fair share of the vCPU's CPU resource has exceeded, while UNDER is opposite. The scheduler picks the head of the run queue which is in UNDER state.

Each domain maintains a value of *credit*, which determines the state of the priority for the domain. Every 10ms, the scheduler ticks, and then subtracts credits the domain owns. When a vCPU consumes all its allocated credits (the value of the *credit* is negative), the state of its priority changes into OVER, and then the vCPU can not be scheduled. Every 30ms, the value of credit each domains owns is to be accounted again, and all domain will get new value of *credit*.

IV. QUESTIONS ABOUT SCHEDULER PARAMETERS

A. *From* weight *to* credit

Under Credit scheduler, each domain has two parameters (*weight*, *cap*). The *weight* determines that the proportional share of physical CPU time the domain attains, and the *cap* represents an upper limit on the CPU time the domain can consume. A domain whose *weight* is 512 will get twice as much CPU as another domain with a *weight* of 256(default), whereas the *cap* is an absolute value, it is expressed in percentage of one physical CPU. The default value for *cap* is 0 which means no upper cap for the domain.

When we set the *weight* for a domain, it is transmitted to the value of *credit* described above in the scheduler according to the following formula: $credit\_fair = (credit\_total * weight_i + weight\_total -1)/weight\_total$, where *credit_fair* is the proportional share of CPU resources, *credit_total* is the sum of all domains' *credit* (initialized as 300), *weight_total* is the sum of all domains' *weight*, and $weight_i$ is $domain_i$'s *weight*. More CPU resource will be allocated to a domain when a bigger value of *weight* set to a domain.

B. Impacts of scheduler parameters configurations

In this paper, finding a satisfactory configuration of scheduler parameters for corresponding applications running in a single domain, or different types of applications running in multiple domains is the main motivation. For example, when an I/O intensive application runs in a domain, while a compute intensive application runs in another one, how to configure the scheduler parameters in order to achieve the desirable performance?

After the innovation of architecture for Xen 3.0, the new I/O model results in a more complex CPU resource usage. In [7], CPU usage for I/O intensive applications has two aspects: CPU used by the guest domain running the application, and CPU consumed in the Domain0 performs I/O processing on behalf of the guest domain. But when applications run in a single virtual machine, how misbehaving the domain0 behave in the resource dispatch?

V. EXPERIMENTAL METHODOLOGY

A. Benchmarks

Our study was performed with the following benchmarks:

Cal: This small application tries to use as much CPU times in its domain as it could. It just runs an infinite loop for computation.

Netperf: We measure maximum achievable network throughput with Netperf 2.4.4 between a server host and a client end which runs in the VM [1].

Iozone: A file system benchmark tool to check the performance of disk [2].

Httperf: A tool to measure web server performance. The performance of web server was measured through it[3].

SysBench: We evaluate the performance of database transactions using SysBench [4].

B. Experimental system

Our testbed was consisted of a Dell OptiPlex 755 with a 2.33 GHz Intel Core 2CPU E6550, 3 GB of RAM, 160GB hard disk, and one 1000M Ethernet cards. The driver domain and all guest domains ran the CentOS 5.1 with the Linux 2.6.24-18-53.e18 kernel, and the virtualized system ran Xen 3.2.1+2.6.24-18-53.e18-xen.

C. Experiments

The experiments used the following test suite combines several typical combinations of applications in Table 1.

TABLE I. TYPICAL COMBINATIONS OF APPLICATIONS

Consolidation types	Benchmark
B + C	Netperf + Cal
D + C	Iozone + Cal
W + D	Httperf + Iozone
W + C	Httperf + Cal
T + D	SysBench + Iozone
T + C	SysBench + Cal

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:14:01 UTC from IEEE Xplore. Restrictions apply.

B: bandwidth intensive application
C: compute intensive application
D: disk I/O intensive application
T: transactional database
W: web server

Each character in Table 1 such as B symbolizes one type of applications enumerated above. They will be also used in the following.

VI. EXPERIMENTAL RESULTS

A. *Impact of Domain0's weight on a single domain*

In Figure 1, we find that the performance appeared in the different configurations when we varied the CPU resources allocated to domain0 relative to domain1 is almost the same. When the weights allocated to domain0 relative to domain1 changed, the throughput attained in domain1 kept unchanged and close to the performance got in native Linux. Though Domain0 consumes CPU resources to perform I/O processing on behalf of the guest domain, the impact of its weights made on domain1 is little due to improved Xen kernel.

B. *Proper weight and impact between domains*

Figure 2 shows that network throughput for Credit scheduler with different values of *weight* allocated to different domains. The x-axis presents eight different configurations where the weights allocated to Domain1 relative to domain2 are 1:1, 1:2, 1:4, 1:6, 1:8, 4:1, 6:1, 8:1, while domain1 ran a client end of Netperf to obtain the maximum throughput and domain2 ran a Cal which is compute intensive. The y-axis presents the throughput obtained in the domain1.

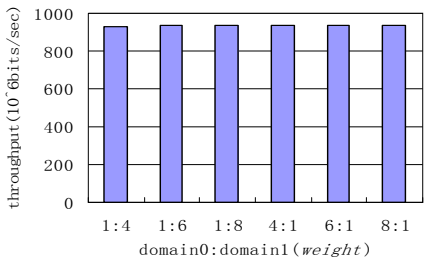


Figure 1. Impact of domain0's weight

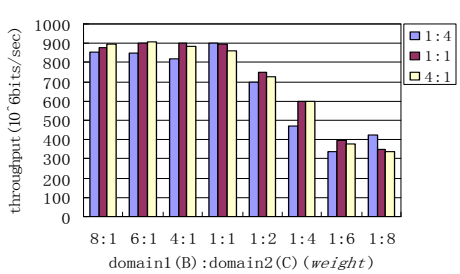


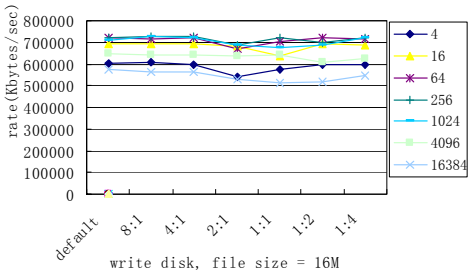
Figure 2. Proper weight for the combination of bandwidth and compute intensive applications

The compute intensive application running in domain2 plays a significant influence on the performance of bandwidth intensive application in the domain1. We first note that when the compute intensive application and the bandwidth intensive application run simultaneously the throughput is quite sensitive to the weight for all domains. Intuitively, when domain1 is assigned more weight than domain2, it gets more credits as we described in the formula in Section 4, then it gets higher priority when scheduled. As compute intensive application performs an infinite compute loop in domain2, it will be scheduled every 30ms (3 slices) and will not free CPU resources until the end of one slice. Though boost optimization in Xen 3.03 is to prevent compute intensive domains from starving I/O intensive domains, it still should wait till the end of 10ms. When bandwidth intensive application needs to be scheduled, it couldn't preempt the computing domain immediately, resulting in decreased bandwidth. Second, the weight for domain1 is not that more higher more better. As Figure 2 illustrates, it is enough to allocate weights to domains when the ratio of the weights for domain 1 to the weights for domain 2 is 1:1.

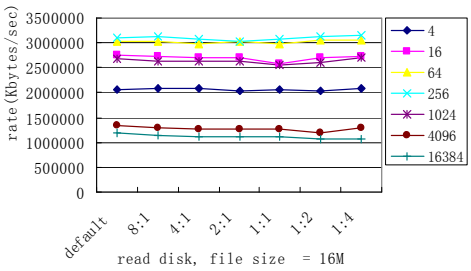
Figure 3 illustrates that the performance of disk writing and reading is almost not influenced by the compute intensive domain. In this test, we ran Iozone to evaluate the performance of disk reading and writing in domain1 while Cal in domain2. The x-axis presents the weights allocated to domain1 relative to domain2 and the y-axis presents the rate of reading and writing disk with different Reclen size from 4KB to 16384KB. In this test, the performance difference for the disk processing in domain1 with different configurations of weight is hardly visible. On account of DMA (direct memory access) in current disk architecture, disk I/O processing is performed by the DMA controller rather than CPU. Thus, the performance of disk I/O processing does not depend much on the weights allocated to domains as shown in Figure 3.

Figure 4 presents the results of the performance evaluation of web server ran in domain1 when there was a Cal or Iozone running in domain2 respectively. First, it is easy to find the dramatic increase of the reply time for the web server when we allocate more weights to domain2 which runs another application. Second, when Iozone runs in domain2, the performance for web server running in domain1 is to some extent better. In deed, for using Httperf to require masses of web pages frequently from the web server running in domain 1, writing disk is necessary. Running an Iozone need to perform lots of disk I/O processing, however, the rate of disk writing and reading is a bottleneck. The consolidation of web server and disk I/O intensive application leads to a higher overhead in disk processing which limits the performance of the web server.

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:14:01 UTC from IEEE Xplore. Restrictions apply.



(a)



(b)

Figure 3. Impact of weight when disk I/O and compute intensive applications running concurrently

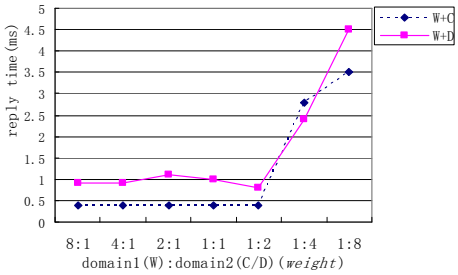


Figure 4. Performance of web server in consolidation of disk I/O and compute intensive applications respectively

Figure 5 shows the performance of SysBench which evaluated the performance of the database running in domain1. We used SysBench to create a table on a MySQL server running in domain1 and fill the table with 1M records. The Y-axis presents the time spending in this processing in domain1 while domain2 runs a compute intensive application and an Iozone respectively. With the decreased weights for domain1, time for database processing is increased accordingly when a Cal is running concurrently. However, with a disk I/O intensive application running in domain2, the database processing time increases till the ratio of domain1's weight to domain2's is 1:1. More weights allocated to another domain performing disk processing have made no impact on degradation of the performance in domain1. In fact, the processing of database transactions involves both disk I/O processing and CPU handling. As disk performs I/O through DMA and the bottleneck problem for the rate of disk processing, provided a certain proportion of CPU resources are enough for it. However, the compute intensive domain consumes as much CPU resources as it is allocated which incurs significant performance degradation of the database transactions while disk I/O intensive domain plays less influence.

Figure 6 presents the performance of consolidation of database transaction and web server. Left y-axis figures the time spending in database transactions while right y-axis introduces the reply time of the web server running in another domain evaluated by Httperf. The performance of database server is a little mixed, fluctuates in different configurations of the *weight*. Nonetheless, allocating two times more weights to the web server domain than that to the database transactions domain is optimal distribution.

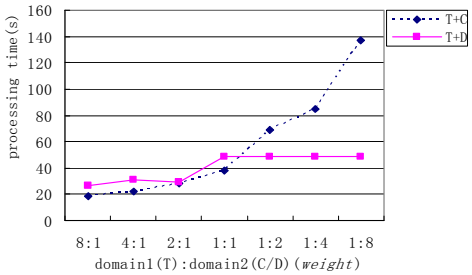


Figure 5. Performance of the database transactions in consolidation of compute and disk I/O intensive applications respectively

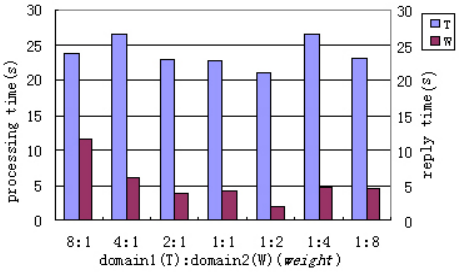


Figure 6. Performance of the web server and database transactions running concurrently in two domains

C. *Impact of both cap and weight*

From Figure 1 we find the performance is not influenced when the weights allocated to domain 0 and domain 1 change. But things are changing when adding cap for Credit scheduler. We used Netperf to get the maximum bandwidth in domain 1 and changed the ratio of weight owned by domain 0 to domain 1. As Figure 7 shows, when domain1's *cap* is changed, the *weight* has also made an impact on the performance. An upper cap of CPU resources for a domain results in significant performance degradation. But when necessary to limit a domain's CPU resources we can set proper weight for the domain to achieve the best performance. We can find in the Figure 7, for bandwidth intensive applications, the best performance achieved when the configuration where weights allocated to domain0 relative to domain1 is 1:1 with corresponding cap.

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:14:01 UTC from IEEE Xplore. Restrictions apply.

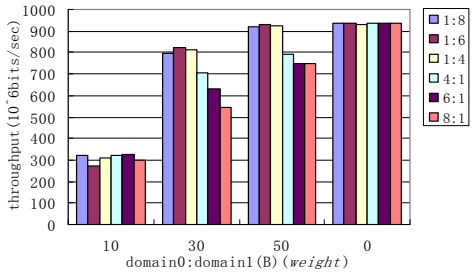


Figure 7. Impact of both cap and weight in single domain

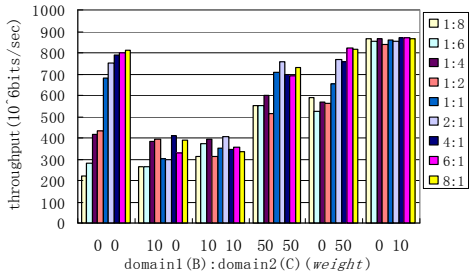


Figure 8. Impact of both cap and weight for an applications consolidation case

Figure 8 presents the throughput for Credit scheduler in different configurations of weight and cap respectively achieved in domain1 when a Cal ran in domain2. Setting different caps for these two domains with different weights leads to a drastic difference in application performance. Since a cap is set to limit the CPU resources allocated to domain2 which ran a Cal, the maximum bandwidth achieved in domain 1 is increased obviously. As shown in Figure 8, weight also plays an important role. We can find in the figure that when the bandwidth intensive domain could use the idle CPU resources and an upper cap (10) for compute intensive domain, the different weights between two domains have not made an impact on the performance.

VII. CONCLUSIONS

The main motivation for enterprises to adopt virtualization is server consolidation. In this work, we make a qualitative comparison about the performance of several typical combinations of applications and identify the impact that scheduler parameters make on the performance of applications.

The parameters of scheduler can make a significant impact on the performance of applications, particularly in the case of consolidation. Furthermore, the mutual influence between domains which run different types of applications is different according to different combinations of applications. First, the study has shown that compute intensive applications have almost made no impact on the disk I/O intensive ones running in another domain. However, if a web server shares CPU resources with disk I/O intensive applications, the performance of web server is degraded significantly. Second, since the rate of disk I/O processing is a bottleneck and DMA in disk architecture, more CPU resources allocated to it can be preempted by other domains. Our work presents a case of a web server concurrently running with a disk I/O intensive application, the experiments results show that provided a certain amount CPU resources is enough for disk I/O processing. Third, the proper scheduler configurations are figured out to schedule several typical types of applications and some proposals are presented for virtualization users to deploy and manage their workloads more efficiently.

Resource allocation and scheduling remains an issue which is understood inadequately. The future work for us are to analyze more concrete server consolidation influenced by the scheduler parameters and explore more complex and integrated relationships among VMs. Efficient scheduler configurations for different types of applications and reasonable arrangements of them are still interesting for future work.

ACKNOWLEDGMENT

This research was supported by State Key Development Program of Basic Research of China (Grant No. 2007CB310900) and National Science Foundation of China (Grant No. 60873023).

REFERENCES

[1] Netperf. http://www.netperf.org/netperf/NetperfPage.html.

[2] Iozone filesystem benchmark. http://www.iozone.org/.

[3] Httperf. http://www.hpl.hp.com/research/linux/httperf/.

[4] SysBench. http://sysbench.sourceforge.net/.

[5] Credit scheduler. http://wiki.xensource.com/xenwiki/CreditScheduler.

[6] D. Ongaro, A. L. Cox, and S. Rixner, "Scheduling I/O in Virtual Machine Monitors", *The International Conference on Virtual Execution Environments (VEE)*, Seattle, WA, March, 2008, pp. 1-10.

[7] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforcing Performance Isolation Across Virtual Machines in Xen", *Proc. of the ACM/IFIP/USENIX 7th International Middleware Conference (Middleware' 2006)*, Melbourne, Australia, Nov. 27 - Dec.1, 2006, pp. 1-13.

[8] L. Cherkasova, A. Vahdat and D. Gupta, "Comparison of the three CPU schedulers in Xen", *ACM SIGMETRICS Performance Evaluation Review (PER)*, Vol. 35, No. 2, September 2007, pp. 42-51.

[9] 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". *In Proceedings of the Symposium on Operating Systems Principles (SOSP)*, Oct. 2003, pp. 1-14.

[10] David Chisnall. The Definitive Guide to the Xen Hypervisor, Prentice Hall PTR, November 19, 2007.

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:14:01 UTC from IEEE Xplore. Restrictions apply.