ISSN: 2393-8528

Contents lists available at www.ijicse.in

International Journal of Innovative Computer Science & Engineering

Volume 5 Issue 3; May-June 2018; Page No. 14-19

Improving CPU Performance of Xen Hypervisor in Virtualized Environment

Vishal Pawar[1], Suraj Yadav[2]

[1] M. Tech Scholar, CSE, Jagannath University, Jaipur, Rajasthan (India)

vishal.vicky.pawar@gmail.com

[2] Assistant Professor, CSE, Jagannath University, Jaipur, Rajasthan (India)

er.surajyadav@gmail.com

ABSTRACT

In the large organizations, which are spread across large geographical area, virtualization plays an important role. This is because the amount of resources of a single physical server is large enough to be completely utilized by a single operating system and hence result in wastage of resources. The virtualization of hardware resources allows more than one virtualized servers to share same physical machine. A significant performance drop is observed in a virtualized operating system in comparison to when it runs directly on hardware. This depends on both the CPU technology as well as the virtualization technique used. The performance of the virtual machines also depends on the virtual CPU scheduling technique used. In XenServer, the Credit Scheduler assigns each virtual CPUs to physical CPUs asynchronously. But if the workload is concurrent, there is a need for synchronization. In this paper, we ran our guest operating system two biggest server virtualization platforms of recent times, namely, VMware ESXi and Citrix XenServer, which both use different approaches to virtualization. After analyzing different parameters in concurrent workload, we found that our guest performs better on VMware than XenServer. Then we proposed an improved virtual CPU scheduling algorithm for Xen hypervisor, which supports synchronization of concurrent programs and significantly reduces the CPU waste time in concurrent workload.

Keywords: Cloud Computing, Performance, Virtualization, VMware, Xen, Virtual Machine Monitor, Hypervisor, Credit Scheduler, Virtual CPU Scheduling

1. INTRODUCTION

Virtualization has become an important part of cloud computing environment. This is because of the functions that are provided by a virtualization platform. There is a continuous improvement in CPU architecture and thus the processing power of modern physical servers. These valuable resources will go waste if only one server works on a physical machine. Virtualization becomes necessary in such modern systems, to fully utilize their resources. Virtualization also benefits the data centers by increasing number of users on cloud.

Virtualization provides reliability, scalability, isolation and resource control to data centers. Reliability means that even if one physical resource malfunctions or stops working, the virtual machine should not be affected. For e.g. if one of the physical server stops working, then the virtual machine starts utilizing another physical server on the cloud but does not stops working. Scalability is largely improved using virtualization. As the number of virtual machines can be increased easily, the data centers are now more scalable [1].

When more than one virtual machines run on the same physical system, it becomes important for virtual machine monitor (VMM or hypervisor) to isolate them, especially on a cloud environment, where there are users from all around the world with no mutual co-operation. Hypervisor makes sure that the performance of a virtual machine must not be affected by the execution of applications on other virtual machines. All VMs and their performance must be independent of each other. Resource control is another function of a hypervisor which provides a control of how much resources a virtual machine is allowed to use. This is useful in cases where we know in advance that what kinds of applications will be executed in which virtual machine [2].

Virtualization has many advantages but it also comes at the cost of performance drop. There is a significant overhead in performance in presence of hypervisor. Since hypervisor is at the lowest level in

**Corresponding author: Vishal Pawar*

virtualization architecture, any performance overhead on guest operating system due to hypervisor can not be further reduced in upper layers, it keeps on adding like a tax. Hence it is important to select a hypervisor that provide minimum performance overhead [8].

The virtual machine performance depends on their scheduling on the physical server. Thus, it is important to have a good VM scheduling algorithm. In Credit Scheduler in XenServer, each virtual CPU is assigned to a physical CPU asynchronously. This gives best performance in parallel workload. But if the workload is concurrent, then the synchronization between threads is necessary, because the threads are not independent. The original credit scheduling algorithm in XenServer assign each VCPU asynchronously to PCPUs. This can lead to waste of CPU time. So, it is required that the load balancing algorithm should not modify the scheduling decisions that were made for synchronization [3]. Our proposed algorithm is a modification of the original credit scheduling algorithm.

In this paper, we measured the difference in performance of guest operating system, which is Windows Server, by running it, first on Citrix XenServer and then on VMware ESXi. This experiment suggests that VMware ESXi proves to be better than Citrix XenServer with concurrent workload. We proposed a virtual machine scheduling algorithm which is a slight modification of Credit Scheduling algorithm in XenServer. We introduced a new priority level for concurrent program threads, which allow them to preempt and run at the next time slice. This increases CPU utilization and throughput.

The remaining paper is divided as follows. Section 2 gives a background knowledge on hypervisors, their architecture and Credit Scheduling algorithm. Section 3 summarizes some relevant work in this area. Section 4 contains methodology of our experiment and the proposed algorithm. Section 5 has results and their analysis. Section 6 gives conclusion and directions for future work. Section 7 is References.

2. BACKGROUND

Virtualization is the method to create the virtual version of system components. In server virtualization, the guest operating system does not interact directly with the hardware. There is an addition layer between the two, this abstraction layer is known as hypervisor or virtual machine monitor (VMM). The virtual machine is an isolated environment where we install an operating system. This is operating system is called guest operating system. The hypervisor is responsible for handling low level instructions of guest operating system. The way in which these instructions are handled depends on the virtualization approach used [4].

2.1. Different Virtualization Approaches

Full Virtualization: In this approach, the hypervisor provides a complete virtualized environment to guest operating system. This is called full virtualization because the guest operating system does not know that it is being virtualized. All the guests are isolated from each other. The binary translations are used for converting the instructions made for hardware access. These instructions are given by hypervisor on the behalf of virtual guest. The guest operating systems are not modified in this approach.

Para-virtualization: In this approach, the guest operating systems are modified so that there is a good coordination between the hypervisor and guest operating system and are run in a lower ring than full virtualization. Since the guest operating system is modified, it is aware of it being virtualized. The merit is that the guest operating systems can be optimized specially to run in para-virtualized environment. This limits the guests to be open source like Linux. Although all major operating systems are now configured to run in XenServer, which uses para-virtualization.

Hardware Assisted Virtualization: Virtualization technique is also being supported by hardware vendors like Intel and AMD. Their processors support running of different operating systems on a physical machine. Each operating system manages its own processor independently. Here the virtual processors are real, unlike in full virtualization where those are simulated. Intel-VT and AMD-V are the examples which provide hardware-assisted virtualization [4].

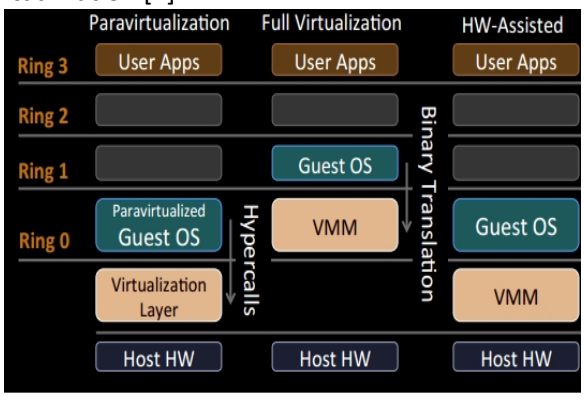


Figure 1: Virtualization Approaches

2.2. Types of Hypervisors

A hypervisor is the virtualization layer between guest operating systems and underlying hardware. The types of hypervisors are as follows: Type 1 and Type 2 [9].

Type 1 Hypervisor: This is also called bare metal hypervisor. This hypervisor installs and runs directly on the hardware. It does not require any other operating system. Type 1 hypervisors generally do not provide graphical user interface. Instead these provide simple BIOS like interface operated with keyboard. These types of hypervisors are mostly used on dedicated servers where better performance is the priority. E.g. VMware ESXi, Citrix XenServer.

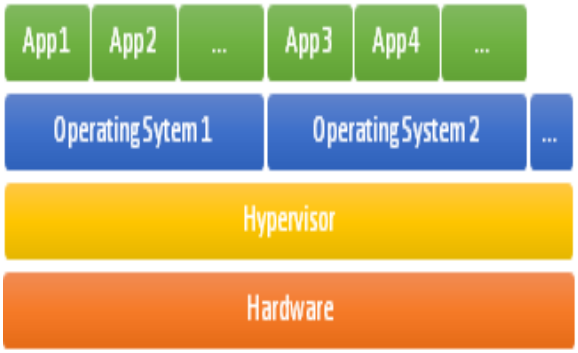


Figure 2: Type 1 Hypervisor

Type 2 Hypervisor: In Type 2 hypervisors, there is a host operating system on which the hypervisor installs and runs. This hypervisor looks like any other application program with a graphical user interface. These hypervisors give lower performance than the type-1. Hence these are not used on big servers. These can be used by any personal computer user to test different operating system. E.g. VMware Workstation, Oracle VirtualBox.

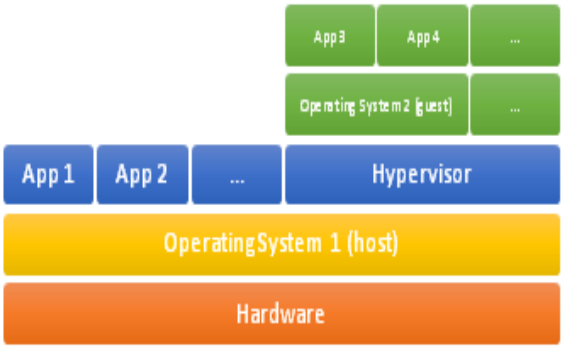


Figure 3: Type 2 Hypervisor

2.3. VMware ESXi and Citrix XenServer

VMware ESXi: VMware ESXi is a commercial type 1 hypervisor which provides full virtualization. VMware provides simple interface for administrator. It provides the guest operating systems with virtual hardware. The guest operating system is not modified so it remains unaware of it being virtualized. All the virtual machines are isolated from each other. Virtual machine monitor tool VMware vSphere is used to configure virtual machines [1].

Citrix XenServer: Citrix XenServer is also a type 1 hypervisor, but it is free and open-source. It provides paravirtualization. XenServer modifies the guest operating system, hence the operating system is aware of it being virtualized. XenServer runs in ring 0, and the guest runs in ring 1. The guest cannot directly access the hardware. The virtual machine monitor tool used to configure virtual machines is Citric XenCenter [7].

2.4 Xen Credit Scheduler

The default scheduler in XenServer is Credit Scheduler. It works as follows: The scheduler maintains weight and cap for each domain. The weight is the amount of CPU time a domain can get. The cap is the maximum amount of CPU that can be consumed by the domain. The credit value of every domain in calculated as:

credit_fair = (credit_total * $weight_i$ + weight_total – 1) / weight_total

Here, credit_fair is proportional share of CPU resources.

credit_total is the sum of all domain's credit. $weight_i$ is $domain_i$'s weight. weight_total is sum of all domain's weight [5].

There is a domain of every virtual machine. This domain consists of all its processors called VCPUs. There is a processor in each domain which monitors the operation of all other processors. This is called Virtual Bootstrap Processor (VBSP). All remaining processors are called Virtual Application Processors (VAP).

The credit scheduler assigns each VCPU to a PCPU. Each PCPU maintains a queue of runnable VCPUs. Each VCPU in the queue is given a priority. These VCPUs are sorted by their priority and not by their credit. The next VCPU to run is chosen from the head of the queue. As a VCPU starts executing, its accumulated credit value is consumed at a rate of 100 credits per 10ms. Each VCPU is given a time slice of 30ms [3].

The priority levels defined are OVER, UNDER and BOOST. When a new VCPU awakes, its priority is set to BOOST, which is the highest priority, so that it is

put at the head of running queue. OVER is the lowest priority level, which is set when a VCPU exceeds its fair share of CPU, otherwise its priority is UNDER. This default scheduler is a non-preemptive in nature.

3. RELATED WORK

Many research works have been done in the performance field of the hypervisors. It is seen in many research works that measure and compare the performance overhead imposed on the guest due to the presence of hypervisors.

“Diagnosing Performance Overheads in the Xen Virtual

Machine Environment” [8] introduces Xenoprof, a system-wide statistical profiling toolkit. It quantified the overhead due to the presence of Xen hypervisor by comparing it with a non-virtualized environment. The reasons for this performance overhead were found and ways to overcome them were devised.

“A Performance Comparison of Hypervisors” [6] by VMware also compared the performance of VMware ESXi and XenServer from the viewpoint of a large enterprise infrastructure. It was found that ESXi is a better option than XenServer for a large enterprise.

“A Component-Based Performance Comparison of Four Hypervisors” [9] Compared the performances of four hypervisors, VMware ESXi, Citrix XenServer, Microsoft Hyper-V and KVM. The results showed that no single hypervisor was best in all respects. In fact, the performances of hypervisor largely depend on the types of applications and the types of resources available to it. Different types of hypervisor may be best suited for different workloads.

"Performance Evaluation of the CPU Scheduler in XEN" [5] presented an evaluation of Xen Credit scheduler performance under different conditions. It presents running of Xen scheduler in different VM configurations and different application. It concluded that the CPU performance depends on various scheduling parameters.

4. METHODOLOGY

For our purpose of performance comparison, we took the following components: CPU, memory, disk, and system uptime. These components were analyzed in Microsoft Management Console Windows Performance Monitor tool. We gave same amount of resources to our guest operating system in both the hypervisors. The performances were analyzed by putting identical workload on both of them.

If the workload is concurrent in some domain, then all the VCPUs in this domain needs to be synchronized to achieve concurrency. We call it need_to_sync domain. Our proposed algorithm must make sure that if the VBSP of a need_to_sync domain is picked to run, then pick to run all VCPUs in that domain and assign them to run on different PCPUs in the same time slice.

For XenServer, we modified the Credit Scheduling algorithm by adding a new priority level TURBO which is greater than all other priority levels. A domain is set to TURBO if the VBSP of a need_to_sync domain is picked to run. There is a global variable turbo_domain which keeps track of the domains which are set to TURBO. All the VCPUs of the turbo domain are set to priority TURBO, and hence these are moved to the front of local run queue. Each PCPU sort its local run queue whenever a turbo domain is set.

4.1 Algorithm

Algorithm: Modified Credit Scheduler

Input: The current time

Output: Task to run next

Step 1: Check if the VCPU that is about to end its time slice is runnable, i.e. if it still has positive credit. If it is runnable, then insert it in the local run queue again.

Step 2: Select the next VCPU from local run queue.

If it has TURBO priority then remove it from local run queue and set it as the task to run next.

If it has positive credits available, and turbo domain is not set then also remove it from local run queue and set it as the task to run next.

If it does not have positive credit, i.e. has eaten its credit, then check if there is more important task on another PCPU using turbo_domain setting.

If it is, then set it as the task to run next. Otherwise, set the next task from the top of local run queue as the task to run next.

Step 3: Set the time slice as follows: If the current VCPU is IDLE then set time slice negative (-1). Otherwise set the default time slice.

Step 4: If task to run next is VBSP of the turbo domain then, clear turbo domain setting.

Step 5: Check if next member of local run queue is

a VBSP. If it is so, and if turbo domain is not set, then set TURBO domain. Also perform the sorting of running queues on all PCPUs so that the VCPUs are sorted in descending order of their priorities, such that VCPUs with TURBO priority comes to the head of their local run queue.

Step 6: Return task to run next.

5. RESULT

5.1. Experimental Setup

The experiments were performed on both VMware ESXi and Citrix XenServer one by one by installing them on the same system. Putting workload and measuring performance, both were done inside the guest operating system. The proposed algorithm of modified Credit Scheduling for Xen is implemented in NetBeans as a simulation of virtual CPU scheduling. The programming language used is Java.

Hardware: The machine on which hypervisor run has Intel Core i3-5010 CPU 2.10 GHz, and 4 GB memory.

Software: The hypervisors used were VMware ESXi 6.0 and Citrix XenServer 6.5. The performance monitoring tool is Windows Performance Monitor. The IDE used is NetBeans IDE 8.2 with JDK 8.

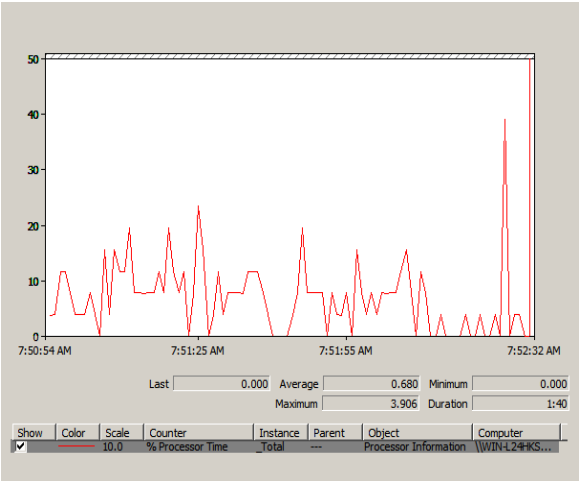
Virtual Machine: The guest operating system that is installed on the virtual machine is Windows Server 2008 r2 (64-bit). The disk image size is 10 GB and memory size is 4 GB. Two virtual CPUs were assigned to the virtual machine.

5.2. Performance Analysis:

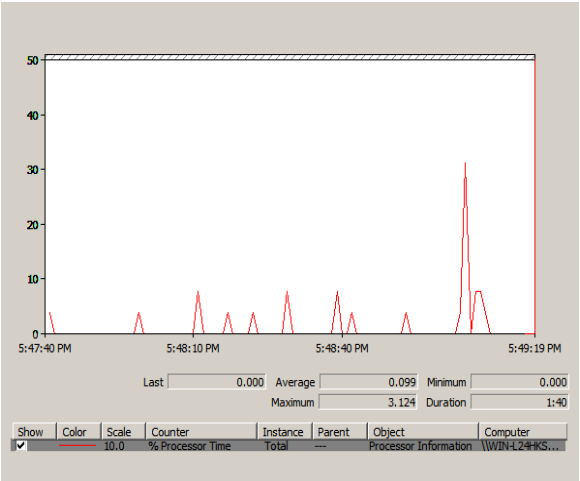
We analyzed the performance from the viewpoint of guest operating system and concluded upon which hypervisor performed well and why. CPU performance is analyzed by setting a counter in Windows Performance Monitor. The separate analysis of CPU performance is as follows:

5.2.1. CPU Analysis:

The counter was set to processor time. We analyzed the CPU utilization by putting general workload on guest operating system in both hypervisor. It was found that VMware performed better than XenServer. It consumed lesser power than XenServer in the same interval of time.



Xen Server



VMware

Figure 4: Comparison of Processor Performance of VMware ESXi and XenServer (lower is better)

5.2.2. Modified algorithm analysis:

After running both algorithms with concurrent workload, we found that the performance is almost same with 1 and 2 virtual machines. This is because, even if the workload is concurrent, both VMs will execute their processes turn by turn. There is a significant improvement in modified algorithm when the number of VMs is further increased. As shown in the graph, the wasted CPU time increases rapidly in default Credit scheduling algorithm of Xen for concurrent workload. While our modified scheduling algorithm shows only slight increase in CPU waste time even with up to 16 virtual machines.

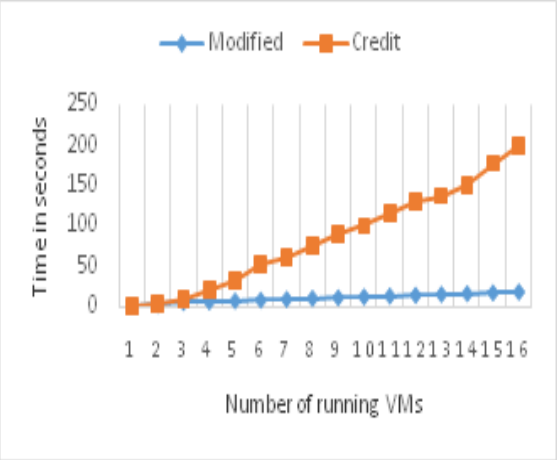


Figure 5: CPU time that is wasted with Credit scheduler and modified scheduler (lower is better)

6. CONCLUSION

Virtualization technology is very important for the Cloud Computing environments. Large enterprises are benefitted from virtualization by reducing hardware costs and increase efficiency and profit. The hypervisor which is to be used at such large scale should be chosen wisely, according to the need and type of workload. In our experiments, we took Windows Server 2008 r2 (64-bit) as the guest operating system and executed it one by one on VMware ESXi 6.0 and Citrix XenServer 6.5. It was concluded that VMware outperforms XenServer. The main factor in CPU performance is its scheduling technique for virtual machines.

Our study could be useful for comparing and deciding on which among these hypervisor gives better performance. VMware becomes a better choice for hypervisor in large enterprises. But performance not only depends on kind of hypervisor, but also on the kind of workload (e.g. sequential, concurrent, parallel, etc.)

We proposed a new virtual CPU scheduling algorithm for XenServer. It is the modification of its default credit scheduler and optimized for concurrent workload. But for parallel workload, the default credit scheduler performs better.

More test could be performed by comparing them on different kinds of workload and more hypervisor could be added to make is study more extensive as the future work. Also, the development of a virtual CPU scheduling algorithm changes its technique itself according to the kind of workload on the virtual machines can be the direction for future work.

REFERENCES

1. Soundararajan, Vijayaraghavan, and Kinshuk Govil. "Challenges in building scalable virtualized datacenter management." ACM SIGOPS Operating Systems Review 44.4 (2010): 95-102.
2. Wood, Timothy, et al. "Profiling and modeling resource usage of virtualized applications." Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware. Springer-Verlag New York, Inc., 2008.
3. Tseng, Chia-Ying, and Kang-Yuan Liu. "A Modified Priority Based CPU Scheduling Scheme for Virtualized Environment." International Journal of Hybrid Information Technology 6.2 (2013): 39-49.
4. Chen, Qian, et al. "On state of the art in virtual machine security." Southeastcon, 2012 Proceedings of IEEE. IEEE, 2012.
5. Xu, Xianghua, et al. "Performance Evaluation of the CPU Scheduler in XEN." Information Science and Engineering, 2008. ISISE'08. International Symposium on. Vol. 2. IEEE, 2008.
6. VMware, “A performance comparison of hypervisors,” VMware White Paper, 2007.
7. Barham, Paul, et al. "Xen and the art of virtualization." ACM SIGOPS operating systems review. Vol. 37. No. 5. ACM, 2003.
8. Menon, Aravind, et al. "Diagnosing performance overheads in the xen virtual machine environment." Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments. ACM, 2005.
9. Hwang, Jinho, et al. "A component-based performance comparison of four hypervisors." Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on. IEEE, 2013.