2009 IEEE International Conference on Cloud Computing

Application Performance Isolation in Virtualization

Gaurav Somani and Sanjay Chaudhary
Dhirubhai Ambani Institute of Information and Communication Technology,
Gandhinagar, INDIA
{gaurav_somani, sanjay_chaudhary}@daiict.ac.in

Abstract

Modern data centers use virtual machine based implementation for numerous advantages like resource isolation, hardware utilization, security and easy management. Applications are generally hosted on different virtual machines on a same physical machine. Virtual machine monitor like Xen is a popular tool to manage virtual machines by scheduling them to use resources such as CPU, memory and network. Performance isolation is the desirable thing in virtual machine based infrastructure to meet Service Level Objectives. Many experiments in this area measure the performance of applications while running the applications in different domains, which gives an insight into the problem of isolation. In this paper we run different kind of benchmarks simultaneously in Xen environment to evaluate the isolation strategy provided by Xen. Results are presented and discussed for different combinations and a case of I/O intensive applications with low response latency has been presented.

1. Introduction

Virtual machines are the key blocks of utility computing or on-demand facilities like cloud computing. Modern data centers host different applications ranging from web servers, database servers and high performance computing nodes to simple user desktops. Although the concept of virtualizing resources is three decades old [1] but it is gaining popularity after the term on-demand computing or cloud computing arose. There are types of virtualization technologies available:

- Full virtualization,
- Para-virtualization and
- OS level virtualization

These techniques differ from each other in their internal architecture and how they communicate with guest operating systems. Xen is the open source virtual machine monitor developed at computer laboratory, University of Cambridge, UK. It follows para-virtualization methodology in resource virtualization [2]. Data centers which host these virtual machines on their physical machines follow Service Level Agreements (SLAs), which specifies the service requirements with different constraints and parameters to be fulfilled by service provider or cloud provider [3]. These constraints and parameters include total uptime and downtime, requirement of CPUs, network bandwidth and disk space. While running more than one virtual machine on a single physical server, virtual machine scheduler is responsible for allocating resources as defined by SLAs. This allocation also includes a most demanding and inherent property which is referred as isolation among virtual machines. Isolation is meant for securing and providing the resources to a virtual machine which is co-hosted with other virtual machines on a single physical server. These resources are CPU share, network share, memory share and disk share to each virtual machine. Thus isolation property is forbidding a misbehaving virtual machine to consume other virtual machine resources and providing fairness according to their shares.

In this paper we are intended towards checking isolation using a set of experiments on Xen virtual machine monitor. These experiments are aimed towards getting a perception of scheduling granularity and their effects on applications. Section 2 of this paper discusses the Xen architecture and scheduling algorithms provided. Section 3 elaborates experimental setup and their relevance. Section 4 discusses the results and their significance in the isolation problem. Section 5 discusses related work in research community and Section 6 concludes and directed towards future work.

2. Xen Virtual machine monitor

2.1 Architecture

Xen is the virtualization tool for the x86 architecture which supports paravirtualization. To support full virtualization it requires virtualization technology enabled processors. Xen architecture shown in figure 1 elaborates the basic blocks in it. Xen designates domain-0 which is the host operating system as isolated driver domain (IDD) to provide device driver support to guest operating systems. Thus in Xen architecture the device drivers in host

978-0-7695-3840-2/09 $26.00 © 2009 IEEE
DOI 10.1109/CLOUD.2009.78

operating system will serve guest operating systems. Guest Domain i.e. DomainU can access drivers via backend drivers provided by Domain0 [2].

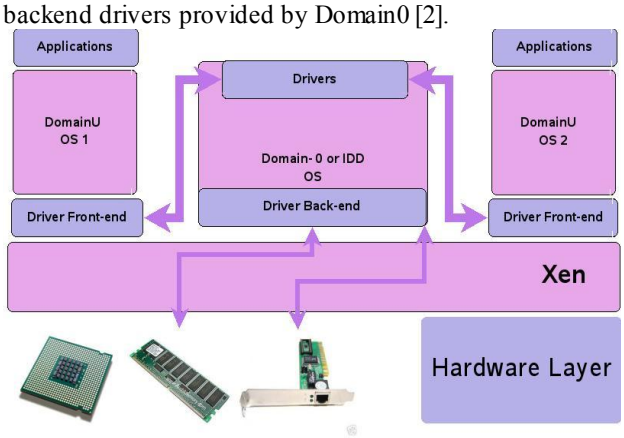

Figure 1 Xen architecture and driver domain

Applications like web servers, database servers or HPC nodes run on the top of these domains.

2.2 Scheduling

Xen allows users to choose among number of schedulers available in scheduler set. These schedulers can be chosen at boot time. Xen has included many schedulers in its code over time, but presently it supports only two schedulers, Simple Earliest Deadline First (SEDF) and Credit Scheduler [4] [5]. In the following section introduction about these two scheduling strategies is given.

2.2.1 Simple Earliest Deadline first Scheduler

SEDF scheduler is the extension to classical Earliest Deadline First (EDF) scheduler. This scheduler provides weighted CPU sharing in an intuitive way and uses real-time algorithms to ensure time guarantees. It is a real time scheduler which operates on the deadlines of the domains. Applications with least deadline will be scheduled first to meet their goals on time. Xen operates on SEDF scheduler with two parameters which are decided by system administrator i.e. Time Period P_i and Time slice S_i. So each and every runnable domain will run for S_i time in a period of P_i time. So this kind of scheduler will give the soft real time guarantee to domains. It maintains a per CPU queue to schedule domains according to their deadlines [4] [5]. The deadline of each domain is calculated by the time at which the domain's period is ending. SEDF is a preemptive scheduler whose fairness 1is decided by the parameters chosen by user. SEDF can be a good choice for the application with latency intensive tasks. Domains which host I/O intensive application require very less CPU time but that time is critical in such applications.

2.2.2 Credit Scheduler

Credit scheduler is the latest and default built-in scheduler in Xen [2]. Credit scheduler is a scheduler with Proportional Share (PS) of CPU allocated to each domain. Each domain is assigned number of credits to consume in comparison of other co-hosted domains. The overall CPU time allocation among all the domains will be done with respect to their assigned weights. According to those weights the scheduler assigns number of credits. Over the time Credit Scheduler has incorporated number enhancements. Credit scheduler keeps a domain in one of the three states, UNDER, OVER and BOOST. Domain state is decided on the basis of the number of credits available in a domain's account.

Domain is designated in UNDER state if it has some credits available to use otherwise it is in OVER state where its all credits are consumed. The domains with UNDER state are having higher priority than domains with OVER. The domains in the UNDER state are run simply on First Come First Serve basis [5]. If there is no domain with UNDER state available than the CPU time will be allocated to a domain with OVER priority. Credits are deducted by 100 from each running domain's account on every 10 ms. A Domain will receive maximum 30 ms to run each time it has been scheduled and if it has enough credits to do so. Credit scheduler uses a third state BOOST state which is added to support I/O applications which needs low response latency. BOOST state is assigned to an IDLE domain which has received an event in event channel of Xen. Thus currently running domain with UNDER priority will be preempted and this domain with BOOST priority will be scheduled first. It helps in improving the performance of I/O applications. But this effect will disappear gradually, if multiple domains are expected to perform I/O [5]. BOOST application can only preempt the running domain.

3. Experimental Setup

3.1 Overview

Experiments are designed to evaluate the isolation provided by the Xen virtual machine monitor. Many authors had quantified a number of virtualization platforms with different experiments [5][6][7][8][9]. Our experiments differ from these findings in following ways.

(1) Isolation studies so far show isolation among same type of applications on a single physical server. In

addition to the findings of others experiments, our approach shows the isolation when running different resource intensive applications simultaneously.

(2) Effect of running an I/O application with varying CPU intensive workloads has been executed in different domains using two different schedulers to quantify the performance of latency driven applications.

3.2 Benchmarks Tests

Three kinds of benchmarks are chosen to test the isolation. These tests are inclined to stress three elementary resources i.e. CPU, network bandwidth and disk I/O speed. Following are the reasons for doing such kind of tests:

(1) To characterize the behavior of scheduler running two non-similar resource intensive applications.

(2) Application placement: where to place applications in data center to get maximum isolation and fairness.

Xentop is used to measure real time information about a Xen system and per domain CPU consumption on specified interval. The experiment uses delay of 1 second to measure the consumption. The overhead incurred due to this program is negligible and fair towards all experiments. The three tests are as follows.

(1) 'CPU': CPU intensive program is a computation intensive loop which contains a number of integer and floating point instructions. This is a simple C program with these instructions in a long running loop of 10^8 counts.

(2) 'NET': Network intensive Program, Iperf is an open source measurement tool that can generate UDP or TCP traffic and measure the network throughput. It is written in C++. Here we ran our tests for TCP traffic. The guest domain is running Iperf as a server program and a Test client is running on a separate machine as Iperf client. The test script loops for six iterations to run altogether with other tests. [10]

(3) 'DISK': Disk intensive test, Iozone is an open source disk intensive program which does read, write, reread, rewrite and other variance of these disk intensive tests. Our test perform disk read and write for file sizes ranging from 64 KB to 64 MB with a minimum record size of 64 KB [11].

These tests were run on a server with the configuration as given in figure 2. Server is running two guest domains to start two tests at the same time. The configuration setup of our test bed is described in table 1.

Processor Architecture	Intel Core 2 Duo
Memory	4 GB
Disk	160 GB
Network Connection	100 Mbps
Host OS	OpenSuse 11.0 (x86_64)
Guest OSs	OpenSuse 11.0 (x86_64)
Xen version and change set	Xen 3.2.1- 16881-04-4.2
Cache Size	2048 Bytes
No. of VCPUs to all OSs	2 (with VCPU pinning)
Memory to Guest	1GB
Memory to Host	2GB
Disk (Guest)	8 GB
Disk (Host)	135 GB
Scheduler	Credit
Weight in Credit Scheduler	Same for all the domains
Client Configuration	Intel P4, 1GB, 100 Mbps

Table 1: Configuration setup for experiment set - 1

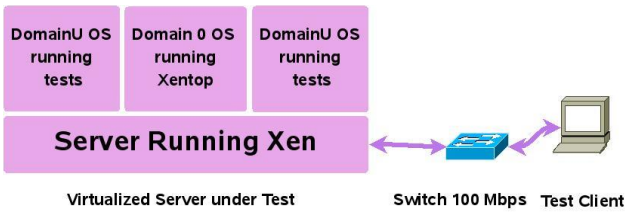

Figure 2: Configuration of experiment Set - 1

3.2.1 Experiment Set 1

Each time we ran different combinations of these tests on two different guest domains and measured different matrices of importance. So the combinations are

(1) CPU and CPU
(2) DISK and DISK
(3) NET and NET
(4) CPU and DISK
(5) CPU and NET
(6) NET and DISK

These tests are designed to run simultaneously, to get an insight into isolation strategy.

3.2.2. Experiment Set 2

After performing the tests in experiment set 1, we have designed one more test which will give a better and correct explanation of Experiment set 2. This test runs on five domains simultaneously. The designed tests are as under:

(1) 'CPU1': CPU intensive program is a CPU intensive loop which contains a number of integer and floating point instructions. This is a simple C program with these instructions in a infinite while() loop

(2) 'PING': Ping latency is designed to calculate round trip time (RTTs).

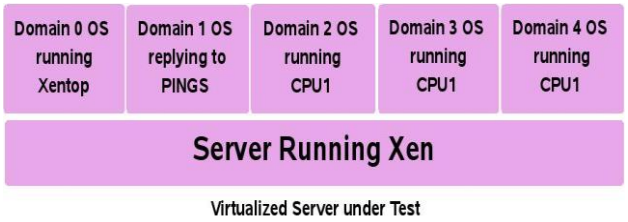

Figure 3: Configuration of experiment set - 2

Processor Architecture	Intel Core 2 Duo
Memory	4 GB
Disk	160 GB
Network Connection	100 Mbps
Host OS	OpenSuse 11.0 (x86_64)
Guest OSs (4)	OpenSuse 11.0 (x86_64)
Xen version	Xen 3.3.0
Cache Size	2048 Bytes
No. of VCPUs to all OSs	2 (with VCPU pinning)
Memory to Guest	512MB
Memory to Host	2GB
Disk (Guest)	8 GB
Disk (Host)	135 GB
SEDF parameters (all)	(P= 10ms and S=1.9 ms)
Weight in Credit Scheduler	Same for all the domains
Client Configuration	Intel P4, 1GB, 100 Mbps

Table 2: Configuration setup for experiment – 2

The configuration setup is shown in figure 3 and corresponding parameters in table 2. We ran five domains simultaneously including domain 0. Each domain will run tests according to the sequence given in table 3. Domain 1 guest is designated as latency driven guest as it is running TEST 'PING' by replying to PING Echo requests coming from the test client for the whole test duration. Main reason behind running this kind of test is to find out the scheduler behavior when we increase or decrease the number of computationally intensive domains. Thus it is a better metric of measuring scheduler behavior when low response latency application is resided with CPU intensive tasks [5].
The experiment set 2 is run for total 400 seconds with making a state transfer of the test in each 50 seconds. These tests are run with SEDF and Credit scheduler available in Xen.

Time (s)	Domain state
0	Domain 0,1,2,3,4 all idle
50	Domain 2 TEST 'CPU1'started
100	Domain 3 TEST 'CPU1'started
150	Domain 4 TEST 'CPU1'started
200	Domain 0 TEST 'CPU1'started
250	Domain 2 TEST 'CPU1'stopped
300	Domain 3 TEST 'CPU1'stopped
350	Domain 4 TEST 'CPU1'stopped
400	Domain 0 TEST 'CPU1'stopped

Table 3: Experiment set – 2

4. Results

Table 4 shows the total time to complete each test. The total time to complete the test is the metric which can be used to see the isolation and fairness. This metric is chosen to see the effect of running two different tests on a single machine

Test	Time to complete
CPU and CPU	57.646s (Domain 1) 57.849s (Domain 2)
DISK and DISK	84.714s (Domain 1) 93.901s (Domain 2)
NET and NET	65.539s (Domain 1, 46.15 Mbps) 65.721s (Domain 1, 47.98 Mbps)
CPU and DISK	56.904s (Domain 1, running C) 58.512s (Domain 2, running D)
CPU and NET	57.258s (Domain 1, running C) 65.248s (Domain 2, running N)
NET and DISK	65.820s (Domain 1, running N) 51.690s (Domain 2, running D)

Table 4: Time to complete each test

4.1 CPU and CPU

In this test, both the guest domains are running CPU intensive test 'CPU'. The total time taken to complete the whole loop of 10^8 by both the domains under consideration is nearly same. The Credit Scheduler's behavior for compute intensive applications is in proportional share. Xentop output for both the domain's CPU share is given in figure 4. Both these tasks are given

same amount of credits each credit allocation interval and they both run for the same amount of time to complete the task. Domain 0 consumes very little amount of CPU for tasks like Xentop for measurement of CPU consumption. The upper limit of Y-axis in each plot in the paper is 200%, as there are two processors available.

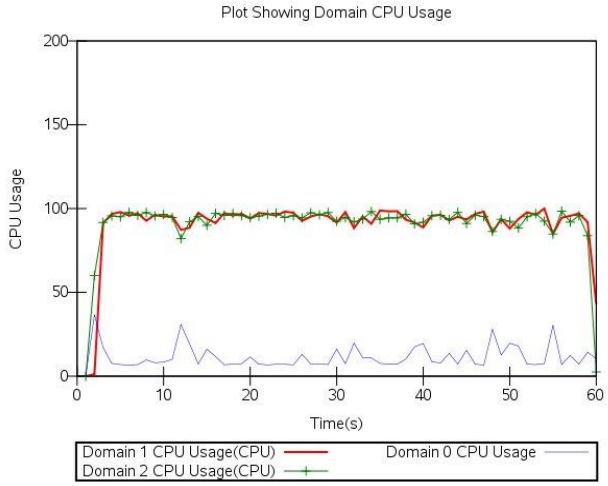

Figure 4: CPU usage by domains (CPU and CPU)

4.2 DISK and DISK

In this test both the guest domains are running file system benchmark, Iozone as a disk intensive program.

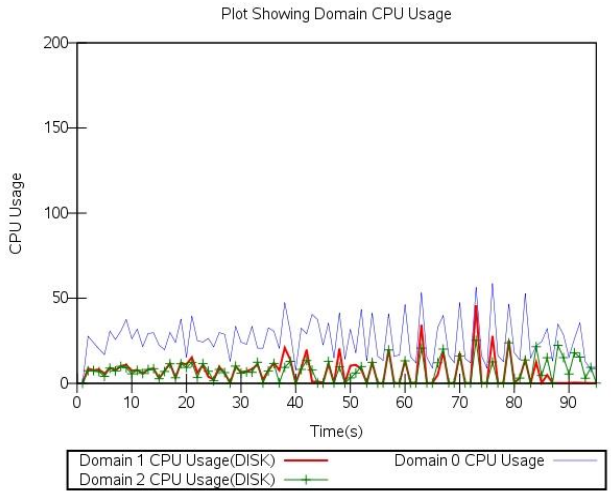

Figure 5: CPU usage by domains (DISK and DISK)

The benchmark is set to give the read and write performance for different file and block sizes (Figure 5). Here both the domains are accessing a single physical disk, thus results into disk resource contention [7]. The credit allocation for both the domains is same. The BOOST state is only assigned to a domain when it is in IDLE state; both the domains are competing to each other for the resource and results in a longer time to complete the test.

4.3 NET and NET

Network intensive tasks using Iperf are run on both the guest domains. The Iperf server runs on these guest domains and two clients are used to send TCP traffic (Figure 6). The bandwidth supported by network hardware is divided in both the domains to run the Iperf tests. It can be extracted from the time to complete whole Iperf test designed is similar to running the same test with other kind of resource intensive tests. The spikes seen in plot are generated after each interval of 10-11 seconds, when each Iperf test completes. The network driver support provided by domain0 shows a high increase in its CPU consumption. Running two Network intensive applications will cut the network bandwidth share by half in this case. So both the domains are experiencing half bandwidth as compare to the bandwidth they have when running the same test with other test like CPU or NET.

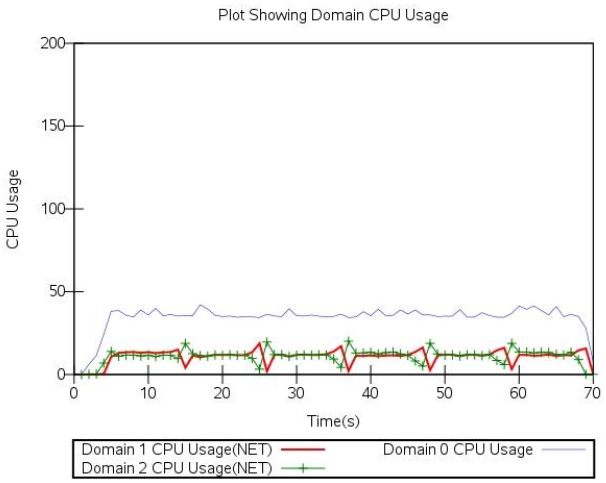

Figure 6: CPU usage by domains (NET and NET)

4.4 CPU and DISK

This test runs a CPU intensive program co hosted with a domain running disk intensive benchmark Iozone. The graph in figure 7 exhibits that disk intensive program running with CPU takes more time than running same with test network intensive test (Figure 7). This is due to no contention of disk resource and less amount of CPU

requirement in disk I/O operation. Clearly Xen's credit scheduler proves itself as a proportional share scheduler in tests where CPU test is available. Experiment 4.5 also shows the same isolation and fairness.

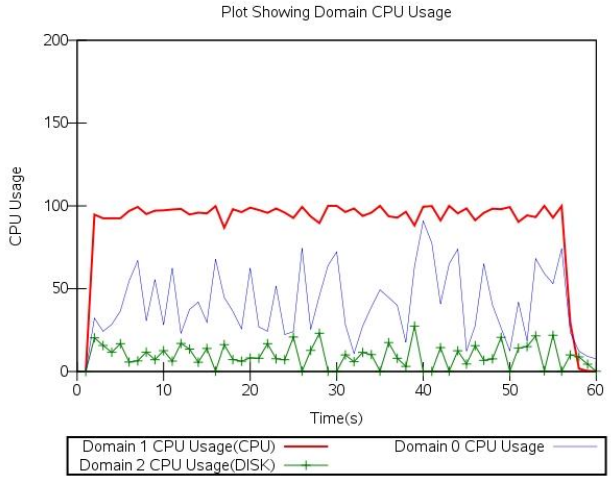

Figure 7: CPU usage by domains (CPU and DISK)

4.5 CPU and NET

This experiment reflects the same output of experiment CPU and DISK.

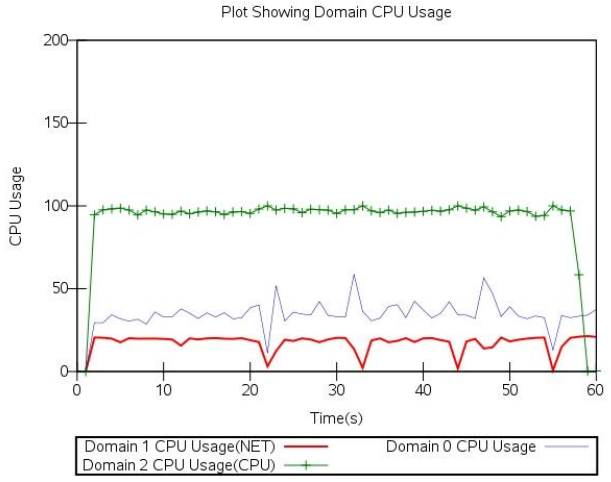

Figure 8: CPU usage by domains (CPU and NET)

The time taken by compute intensive test is also almost same for all experiments. This proves proportional share resource allocation provided by credit scheduler. Domain0 CPU usage count is all due to network driver management. The extra overhead which incurred by Xentop tool is negligible (Figure 8).

4.6 NET and DISK

The time taken by Network intensive test is almost same throughout all the tests. At first time, it shows clear isolation between this application and their combination with other kind of applications (Figure 9). Domain 0 consumes a CPU amount higher than guest domain, which proves the amount of work done by domain-0 on behalf of hosted guests. In this experiment, test DISK consumes least time throughout all the tests which contains test DISK in combination. This is due to no resource contention and Credit's Proportional share property.

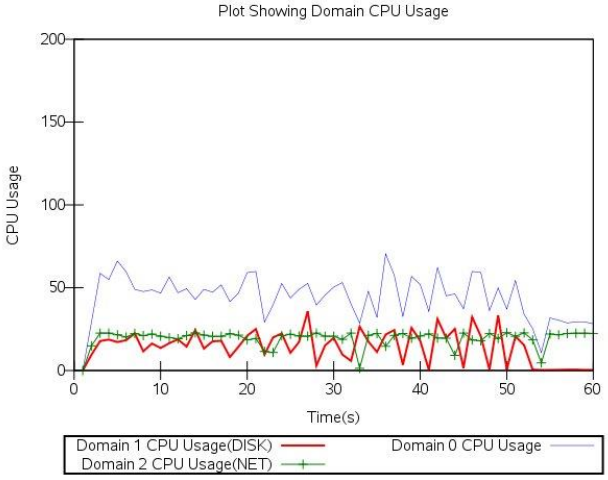

Figure 9: CPU usage by Domains (NET and DISK)

4.7 Experiment Set 2

Experiment set 1 has illustrated the isolation in Xen virtual machine monitor when similar and different kind of resource intensive applications ran on a single server. In Experiment set 2, we illustrate special case where one simple ping application is evaluated on the basis of RTT with CPU intensive domains co hosted.

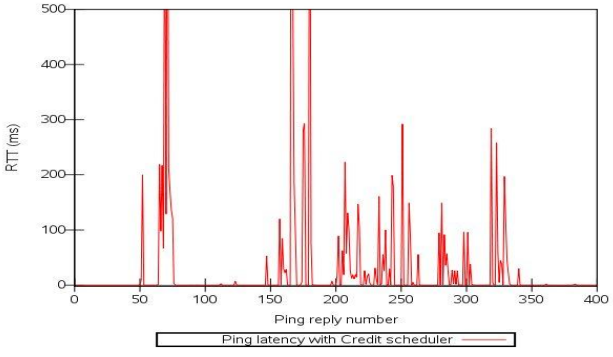

Figure 10: Ping latency with credit scheduler

Although ping is not an application to virtualize in a data center but it will give an insight in to the phenomenon where the I/O latency applications are run with the CPU applications. Table 4 has the experiment run illustrated (Figure 10 and 11).

After each 50 seconds a CPU intensive program is started in each co hosted domain. Both schedulers available in Xen were run to see the difference in running the latency driven applications in conjunction with CPU intensive applications. In the first interval of 0-50 seconds the PINGed domain acknowledges ping requests with RTT~0.300ms. In interval of 50-100s the first domain with a CPU intensive program started and the RTT values become very high. As BOOST condition can be applied when a domain is in UNDER state. After each time when the whole system is having negative credits the I/O domain will be having new credits and it will be in UNDER state. Its credits do not consumed fully because it requires very less amount of CPU and it will goes into inactive state. So PING domain remains in UNDER state. In next interval the other domain with CPU intensive program will be started. For all the intervals the CPU test is resulted into high ping latency. On the other hand if we choose SEDF scheduler in place which gives real time guarantees can perform better for this special case. But that requires careful parameter selection for applications. This does not prove any scheduler better than other. Xen team presently support credit as default scheduler due to its multi processor support or load balancing. On other hand SEDF is a scheduler with real time guarantees and user control over deciding.

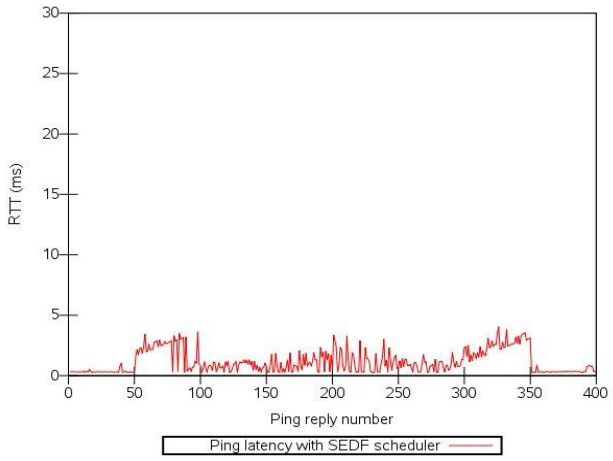

Figure 11: Ping latency with SEDF scheduler

5. Related work

Quantifying isolation among applications in different virtualization environment has been studied by [5][6][7][8][9]. The work done by Jenna et al. has studied effect of a misbehaving virtual machine on other co hosted machine by measuring their performance. They have designed their own set of benchmarks [9]. Ongaro et al. showed results for I/O applications in mixed workload with compute intensive programs and proposed enhancement in Xen credit scheduler [5]. Fabrian et al. has shown the cache and disc interference to see the isolation [6]. Gupta et al. proposed a proper accounting method to improve exact resource allocation and consumption by different guest domains. They developed a scheduler enhancement to calculate exact network share with packet count [8]. Padala et al compared Xen and OpenVZ for resource consumption and scalability and other low level metrics like cache misses and domain-0 consumption [7]. Other than these studies to improve Xen' scheduling methodologies number of approaches has been developed to implement more application oriented scheduling enhancements [13] [14]. Network and I/O virtualization have been discussed in detail by [16] [17]. Menon et al did virtualization performance measurement in [17] with different micro and macro benchmarks. Placement and than load balancing has been seen by Chris Hyser et al. in [21]. They suggested the approach to balance the domains equally among all the physical servers in the whole data center using Simulated Annealing algorithm. Overall the Isolation depends upon the type of application machine is running and their resource requirement with time.

6. Conclusion and Future work

This paper discusses the isolation methodology provided by Xen. Resource contention in terms of disk and network bandwidth is the major consideration for finding a place for a virtual machine to physical host. Xen in many domain environments provide good isolation when running high throughput and non-real time applications with credit scheduler but it becomes difficult to predict the performance and time guarantees when running soft real time applications on it. SEDF has shown relatively good performance than credit scheduler. SEDF requires effective deadline setting and it may have more context switches with smaller slices. In conclusion, high level matrices like time to complete a benchmark test is taken into account while measuring the performance, but more precise and lower level matrices are needed to evaluate scheduler traces for each kind of applications. Work can be continued in the direction while measuring more precise characteristics in Xen environment. It helps in understanding scheduler behavior for different kind of

load they run and their behavior on placement of these applications. Application placement problem is still in its initial phase, it can be seen while running different real time and live benchmarks like for web servers and multimedia applications [19]. Choosing correct parameters and configuring a scheduler is not a trivial task with complex Service level Objectives (SLO). Clear mapping of SLO parameters and scheduler parameter is needed for isolation.

7. Acknowledgement

We acknowledge the comments we received on different topics related to Xen on xen-devel list [20]. We thank Emmanuel Ackaouy, George Dunlap and Zhiyuan Shao for their comments on the things related to experiments.

8. References

[1] Popek, G. J. & Goldberg, R. P. Formal requirements for virtualizable third generation architectures *Communications of ACM,* 1974.

[2] Barham, P.; Dragovic, B.; Fraser, K.; Hand, S.; Harris, T.; Ho, A.; Neugebauer, R.; Pratt, I. & Warfield, A. Xen and the art of virtualization *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, ACM,* 2003, 164-177.

[3] Chen, Y.; Iyer, S.; Liu, X.; Milojicic, D. & Sahai, A. Translating Service Level Objectives to lower level policies for multi-tier services *Cluster Computing, Kluwer Academic Publishers,* 2008, *11*, 299-311.

[4] Cherkasova, L.; Gupta, D. & Vahdat, A. Comparison of the three CPU schedulers in Xen, *SIGMETRICS Perform. Eval. Rev., ACM,* 2007, *35*, 42-51.

[5] Ongaro, D.; Cox, A. L. & Rixner, S. Gregg, D.; dve, V. S. & Bershad, B. N. *(ed.),* Scheduling I/O in virtual machine monitors, *VEE, ACM,* 2008, 1-10.

[6] Fabrian et al., Virtualization in enterprise, *Intel technology Journal*, Volume 10, Issue 3, 2006, 227- 242

[7] Pradeep Padala, Xiaoyun Zhu, Z. W. S. S. K. G. S. Performance Evaluation of Virtualization Technologies for Server Consolidation *Enterprise Systems and Software Laboratory, HP Laboratories, Palo Alto.,* April 11, 2007.

[8] Gupta, D.; Cherkasova, L.; Gardner, R. & Vahdat, A. Enforcing performance isolation across virtual machines in Xen *Middleware '06: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware, Springer-Verlag New York, Inc.,* 2006, 342-362.

[9] Matthews, J. N.; Hu, W.; Hapuarachchi, M.; Deshane, T.; Dimatos, D.; Hamilton, G.; McCabe, M. & Owens, J. Quantifying the performance isolation properties of virtualization systems, *ExpCS '07: Proceedings of the 2007 workshop on Experimental computer science, ACM,* 2007.

[10] Iperf: Network Throughput measurement tool, http://sourceforge.net/projects/iperf, Accessed April, 2009.

[11] Iozone, File System benchmark, http://www.iozone.org, Accessed March 2009.

[12] Schanzenbach, D. & Casanova, H. Accuracy and Responsiveness of CPU Sharing Using Xen's Cap Values *Computer and Information Sciences Dept., University of Hawai at manoa,* 2008.

[13] Weng, C.; Wang, Z.; Li, M. & Lu, X. The hybrid scheduling framework for virtual machine systems *VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, ACM,* 2009, 111-120.

[14] Kim, H.; Lim, H.; Jeong, J.; Jo, H. & Lee, J. Task-aware virtual machine scheduling for I/O performance. *VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, ACM,* 2009, 101-110.

[16]Apparao, P.; Makineni, S. & Newell, D. Characterization of network processing overheads in Xen *VTDC '06: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing, IEEE Computer Society,* 2006, 2.

[17] Menon, A.; Santos, J. R.; Turner, Y.; Janakiraman, G. J. & Zwaenepoel, W. Diagnosing performance overheads in the xen virtual machine environment *VEE '05: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments, ACM,* 2005, 13-23.

[18] Cherkasova, L. & Gardner, R.Measuring CPU overhead for I/O processing in the Xen virtual machine monitor *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference, USENIX Association,* 2005, 24-24.

[19] Pradeep Padala, Kai-Yuan Hou, K. G. S. X. Z. M. U. Z. W., Automated Control of Multiple Virtualized Resources, *HP Laboratories,* 2008.

[20] Xen-developer list, http://lists.xensource.com/archives/html/xen-devel/2009-05/msg00093.html.

[21] Chris Hyser, Bret McKee, R. G. B. J. W. Autonomic Virtual Machine Placement in the Data Center *HP Laboratories,* February 26, 2008.