Performance Investigation of Multi-tier Web Applications in Xen Virtualized Environment

Reza NasiriGerdeh[★], UNIVERSITY OF CALIFORNIA, SANTA CRUZ
Negin Hosseini[✦], IRAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
Rohollah Ehsani[✦], IRAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
Keyvan RahimiZadeh[⌘], IRAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
Morteza AnaLoui[⌘], IRAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

[★] rnasirig@ucsc.edu
[✦] {negin.ho3ini, rohollah.ehsani}@gmail.com
[⌘] {rahimizadeh, analoui}@iust.ac.ir

Abstract. Server virtualization comforts deployment of Internet services and enables cloud service providers to improve resource utilization, fault tolerance, and energy efficiency by consolidating virtual servers within a shared environment. Although deploying services in virtualized environment brings benefits to service providers in terms of service agility, functionality, and reusability in comparison to traditional architecture, overhead of mediating hypervisor and contention amongst hosted virtual machines to possess virtualized resources can adversely affect the performance of applications in virtualized environments. In this paper, we provide a detailed profiling-based performance investigation of Web application in Xen virtualized environment to recognize the major sources of performance degradation of Web application in Xen. Our experimental results indicate that the main cause of the performance degradation of a single Web application in Xen is higher CPU utilization not higher disk utilization of the system; detailed profiling of the performance events occurred at the system-level identifies the most CPU consuming and the highest L3 cache miss functions of Xen and Dom0 kernels that contribute the most to this performance degradation. Additionally, the experiments show that by increasing the number of cores in Xen environment, the number of L3 cache misses occurred at the system decreases significantly which leads to response time improvement of the Web application in Xen. Moreover, the results demonstrate that when multiple Web applications are running in a Xen consolidated environment, not only the high CPU utilization but also the high disk utilization causes the performance degradation. The dramatic increase of L3 cache misses of the system is another cause of this performance degradation. The functions of Xen and Dom0 kernels that handle I/O operations of domains have the most contribution to this considerable increase of L3 cache misses.

Keywords: Virtualization, Cloud Computing, Xen, Performance Profiling, Performance Evaluation, Web Application

1 Introduction

Virtualization technology is experiencing a renewed interest to improve resource utilization and energy efficiency. It is a key technology for cloud computing and green IT. The cloud computing platforms use various virtualization technologies such as Xen [1], VMWare [2], and KVM [3] for server consolidation [4]. The cornerstone of virtualization technologies is Virtual Machine Monitor (VMM) or hypervisor that abstracts the physical machine resources into virtual resources and assigns them to a number of isolated execution environments, known as, Virtual Machines (VMs). In this way, VMM provides an environment to consolidate multiple VMs into a single physical machine and leads to more resource utilization, energy efficiency improvement, and cost reduction [5,6].

Although virtualization technology provides the mentioned benefits, it has faced some challenges. First of all, none of the current virtualization technologies can guarantee performance isolation even though they provide resource isolation [4]. The hypervisor that mediates VMs to access resources can affect the performance of applications running in VM; moreover, the functionality of VMs can have an impact on each other performance [6-8]. Therefore, the performance of applications running in virtualized environment might remarkably degrade compared to native environment because of the VMM overhead itself and interference among VMs.

Secondly, none of the current virtualization technologies is alone suitable for all workload types. Thus, it is important to choose a suitable VMM, operating system, and resource allocation scheme regarding each workload type. Resource configuration has a direct effect on hypervisor performance, which in turn affects the quality of service of submitted workloads [9-10]. Thirdly, predicting the performance of applications is more complicated in virtualized environments because of the complexity inherent in these environments; moreover, the impact of hardware events such as cache misses can have a contribution toward the difficulty of performance prediction in such environments [11].

In this paper, we extend our work from [12] to evaluate the performance of consolidated Web applications in Xen. In our previous work [12], we investigated the performance of a single Web application in Xen through detailed performance profiling of hardware events occurred at the system-level to identify the possible causes of performance degradation of Web application in Xen. In this paper, we do the same analysis, but for consolidated Web applications in which multiple Web applications are running concurrently in Xen. Moreover, we evaluate the effect of increasing the number of cores on the performance of a single Web application in Xen.

To these ends, we measure the throughput and response time of Web application(s) under different scenarios. Meanwhile, we monitor the CPU and disk utilizations of system as well as the hardware performance events occurred at the system-level for each scenario. Afterwards, we present the results based on quantitative and qualitative analyses to show the performance degradation and to identify the major sources of performance degradation in Xen.

The remainder of this paper is organized as follows. The next section reviews the previous related work. Section 3 presents a brief overview of the main architectural aspects of Xen hypervisor and describes the experimental environments and tools we have used in our experiments. Section 4 provides experimental results as well as the detailed analysis of these results to elaborate the main sources of performance degradation in Xen. Finally, section 5 concludes the paper and provides direction for future works.

2 Related Work

Some of the works on performance evaluation of Xen mainly deal with analyzing the performance overhead of Xen or comparing its performance to other virtualization technologies [14,15,16,17]. Barham *et al.* [1] presented Xen as a new virtualization technology and compared its performance and scalability with VMware, native Linux, and User-Mode Linux (UML). They illustrated that Xen performance is near to native Linux and it is scalable so that it is able to host one hundred operating systems simultaneously. Cherkasova and Gardner [13] evaluated the CPU overhead of Web server workloads submitted to DomUs on the CPU overhead of Dom0 in Xen. They indicated that as the workload rate increases and the request size becomes larger, the CPU overhead of Dom0 increases. They argued that Dom0 handles the I/O operations of DomUs and Xen hypervisor employs page-flipping technique to exchange data between Dom0 and DomUs. As the workload rate or request size increases, the number of page-flippings increases accordingly and leads to more CPU overhead of Dom0.

Menon *et al.* [18] modified Oprofile [19] and presented a valuable performance analysis tool Xenoprof that enables the detailed analysis of Xen virtualized environment. They used this tool to recognize certain kernel functions that causes the most overhead for network-intensive applications running on Xen. Menon *et al.* [20] used the results obtained from the work [18] to identify the sources of network virtualization overhead in Xen and proposed three methods to optimize Xen network virtualization. Du *et al.* [21] investigated the requirements for guest-wide and system-wide profiling in virtualized environments. They also developed a guest-wide and a system-wide profiler for KVM, a hypervisor based on hardware-assisted virtualization and a guest-wide profiler for QEMU [22], a hypervisor based on binary translation. Padala *et al.* [23] evaluated the performance of various virtualization technologies for server consolidation. They compared the performance of Xen 3.0.3 and OpenVZ [24] to base Linux on a single-core system. They illustrated that virtualization overhead of OpenVZ is limited while the overhead of Xen virtualization is high. Jang *et al.* [25] explored the performance overhead of processing the networking operations in Xen environment. They also redesigned a networking mechanism to decrease the overhead while keeping the performance level.

Pu *et al.* [26] empirically evaluated the effect of collocating the network-intensive and CPU-intensive workloads on the performance interference among Xen domains under different scenarios and for three different platforms. They also investigated the impact of allocated resources on domains and showed that pinning more cores to Dom0 leads to lower performance of CPU-intensive workloads. Mei *et al.* [27] provided a detailed performance study of network-intensive workloads in cloud virtualized environments.

They illustrated that none of the current virtualization technologies provides acceptable performance isolation in order to guarantee the effectiveness of resource sharing among virtual machines which are running in a single physical machine.

Our work is orthogonal to this group of works and focuses on investigating the performance overhead of Xen. Similar to [18], [20], and [23], we use Oprofile/Xenoprof to monitor the hardware performance events occurred at the system level in order to identify the causes of performance overhead in Xen. Additionally, we carry out a similar detailed analysis at micro level but apply it to different scenarios and configurations in the context of multi-tiered Web application.

Some other works on performance evaluation of Xen deal with the performance prediction and modeling of applications in Xen environment [28, 29, 30, 31]. Menascé [32] used queueing network theory to model the performance of applications in virtualized environments. Hong *et al.* [33] proposed a theoretical performance model to predict the performance of parallel applications in different virtualized environments based on KVM, Xen, and VMware hypervisors. They evaluated the performance prediction model and showed that it is accurate and reliable. We [34] recently evaluated the performance of Web server in Xen-based virtualized environment and proposed some performance models based on queueing network theory to predict the performance of a Web server running on Xen.

3 Overview

In this section, we present a brief overview of Xen architecture. Next, we describe the testbed environment and the tools we used in our experiments. Finally, we explain the methods employed to generate the workload in the conducted experiments.

3.1 Xen Architecture

Xen, named from neXt generation, is an open source, bare-metal VMM which supports two operation modes: hardware-assisted virtualization and para-virtualization. In the former mode, the guest operating system does not require to be modified and guest architecture can be different from the host architecture. In the latter mode, the kernel of the guest operating system requires modification. The modification facilitates faster running of para-virtualized guest operating system.

Fig. 1 demonstrates the general architecture of Xen-based virtualized environment. The Xen VMM runs directly on hardware layer and is able to issue privileged requests to physical resources. It runs a special domain, called Domain0 (Dom0) that is an isolated driver domain (IDD) and has a direct access to the physical resources. The other unprivileged domains (DomUs) can indirectly access to hardware resources through Dom0. In the other words, Dom0 provides access of DomUs to hardware resources. DomUs run device drivers known as Frontend drivers to communicate with Dom0 which in turn runs Backend drivers. Unprivileged domains exchange data with Dom0 over an I/O channel. In order to avoid the overhead of copying I/O data itself to and from DomUs, memory references to buffers are transferred over the I/O channel instead of I/O data [13].

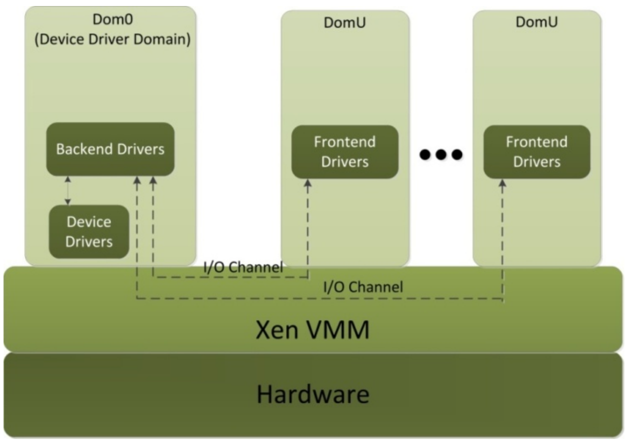

Fig. 1. Xen architecture

3.2 Testbed Environment

In order to investigate the performance degradation of Web application in Xen, we have conducted in-depth experiments in different environments. We have used the hardware testbed consisting of two same HP Proliant DL380 machines, which one of them acts as a server and the other acts as a client. The machines are equipped with two six-core Intel Xeon E5-2620 CPUs running at 2 GHZ with 15 MB L3 cache memory, 64 GB DDR-III of main memory, one 600 GB SCSI hard disk drive, and four Broadcom gigabit Ethernet cards. They are connected together via four Cat6 Ethernet cables. We have performed experiments in four different environments described below.

- **Environment I:** Server machine runs a multi-tier Web application which includes MySQL database system, Apache Tomcat application server, and Nginx Web server in vanilla Centos Linux with kernel version 2.6.18. In this environment, we have activated two CPU cores and limited the main memory of system to 6 GB.
- **Environment II:** In this environment, we have installed the Xen hypervisor in server machine. The Xen hypervisor runs three domains: the first domain acts as backend that runs MySQL database system as well as Apache Tomcat application server, the second domain acts as frontend that runs Nginx Web server, and the third domain is Dom0 which handles the I/O of backend and frontend domains; in addition, we have activated two CPU cores and assigned 4 GB of main memory to Dom0.
- **Environment III:** This environment is the same as the Environment II except that the server machine has four activated CPU cores.
- **Environment IV:** In this environment, Xen runs eight domains as well as Dom0. Half of them are backend domains and the other half are frontend domains; moreover, we have activated four CPU cores and assigned 16 GB of main memory to Dom0.
  Client machine runs Xen hypervisor, too. It runs domain(s) which generate(s) Web requests using Faban [35] workload generator. In environments I, II, and III, it runs only one Faban workload generator whereas in Environment IV, it runs four.

It is worth mentioning that we have installed the latest version of Xen, namely, Xen 4.4.1 in server and client machines and used the Credit-based CPU scheduler of Xen in all experiments; moreover, all domains including Dom0 run XenoLinux derived from Centos Linux with kernel version 2.6.18. All the backend and frontend domains have one virtual core, 1 GB of main memory, and run in the para-virtualized mode. Backend domains have 15 GB of storage while frontend domains have 50 GB of storage. Furthermore, we have disabled the Intel hyper-threading technology. Fig. 2, Fig. 3, and Fig. 4 show the experimental environments. We call Environment I native environment and refer to environments II, III, and IV as Xen environments.

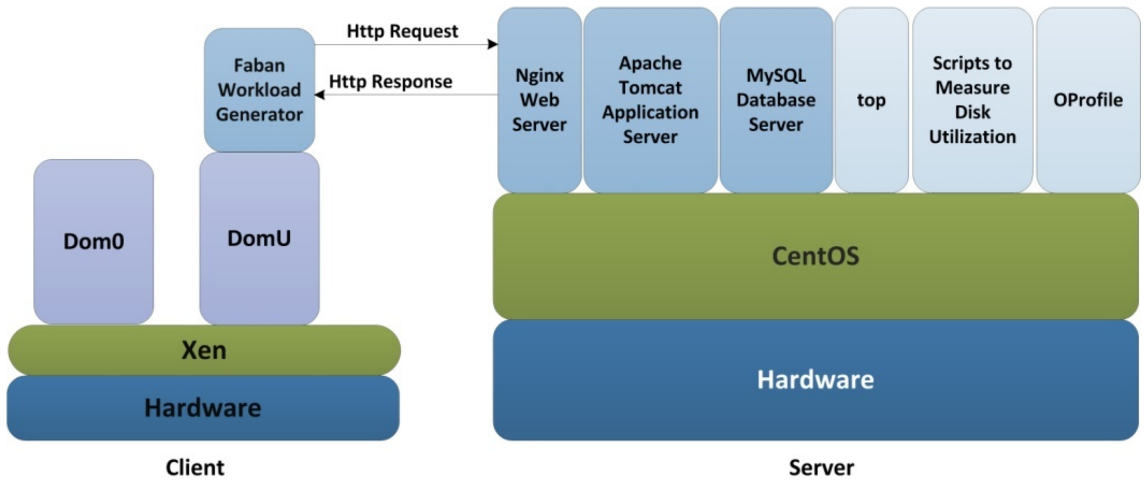

Fig. 2. Environment I

3.3 Workload Description

To investigate the performance degradation caused by Xen, we have employed CloudStone benchmark [36]. CloudStone benchmark consists of Apache Olio [37] as a Web application and Faban [35] as a workload generator. Olio is a Web 2.0 application which represents a social-events website. It allows users to perform several actions such as loading the homepage, adding new events, logging into the site, attending events and browsing events by tag or date. It currently has implementations using three technologies: PHP, J2EE, and Ruby on Rails. In our experiments, we have used PHP implementation of Olio.

Faban is an open-source workload generator consisting of a master program that spawns one or more multi-threaded processes to simulate actual users. It provides a web interface to submit customized benchmark runs and observe their results. Faban enables us to change the load level driven against the

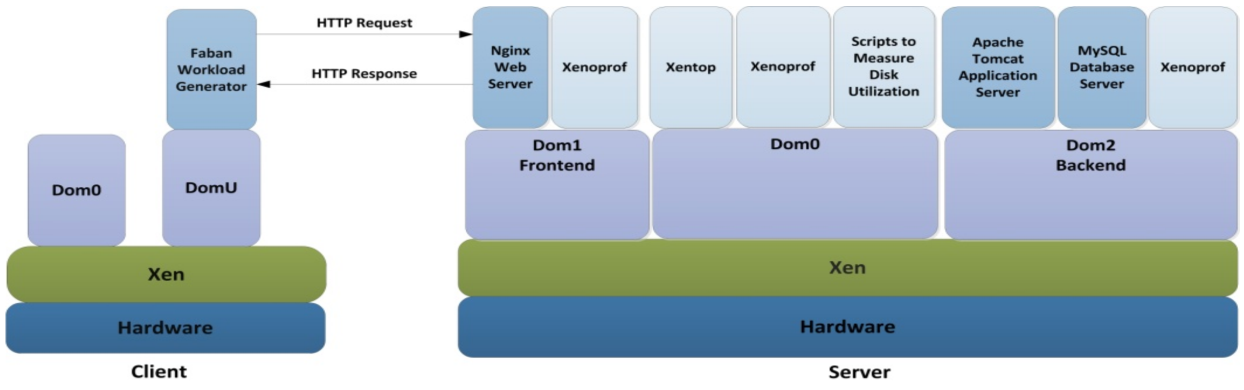

Fig. 3. Environment II and Environment III

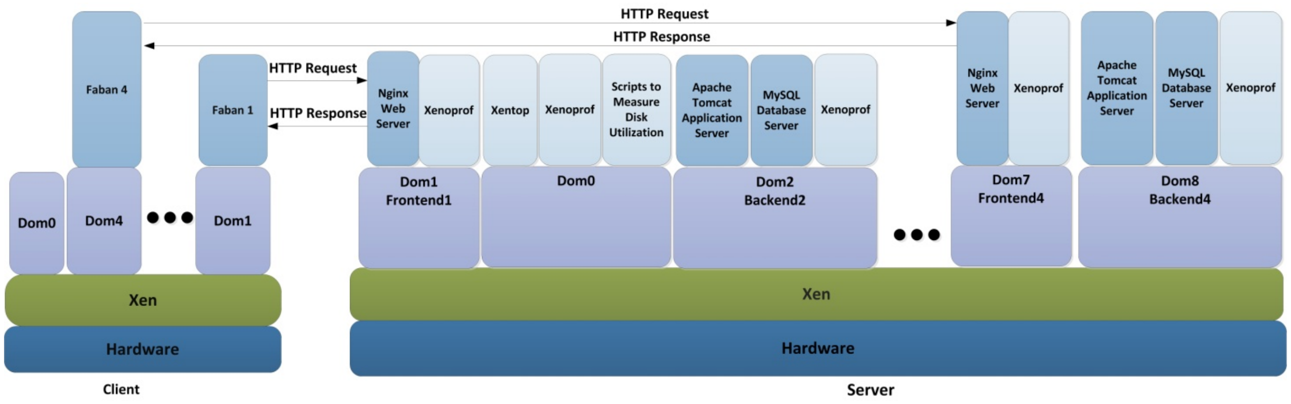

Fig. 4. Environment IV

application by changing the number of concurrent users to be served by the application; furthermore, it allows us to configure total time for each benchmark run by adjusting three different durations: ramp-up, steady state and ramp-down. Faban only takes into account the steady state period to calculate the resulting metrics.

3.4 Monitoring Framework

To evaluate the performance impact of Xen, we require to measure some well-known performance metrics in native and Xen environments. These performance metrics are the throughput and response time of Web application, CPU and disk utilizations of system, and hardware performance events occurred at the system. Faban reports the throughput and response time of Web application at the end of each benchmark run. Therefore, we have relied on the results reported by Faban for these performance metrics.

In native environment, we have employed *top* tool to measure the CPU utilization of the system. It reports the average CPU utilization per core for user, kernel, and nice processor states. If $U_{CPU, User}$, $U_{CPU, Kernel}$, and $U_{CPU, Nice}$ are the CPU utilization in user, kernel, and nice states, respectively, then the total CPU utilization per core in native environment is calculated using Eq. 1.

$$U_{CPU,Native} = U_{CPU,User} + U_{CPU,Kernel} + U_{CPU,Nice} \quad (1)$$

In Xen environments, we have used *Xentop* tool to measure CPU utilization. It reports the CPU utilization of all domains running in Xen. If domains Dom0, Dom1, ..., and DomN are running in Xen, then the average CPU utilization per core in Xen environments is calculated using Eq. 2.

$$U_{CPU,Xen} = \frac{\sum_{K=0}^{N} U_{CPU,DomK}}{M} \qquad (2)$$

where $U_{CPU,\,DomK}$ is the CPU utilization of *DomK* and *M* is the number of physical CPU cores.

To measure the disk utilization in native and Xen environments, we have employed the information in *stat* file which is located in */sys/block/sda* directory. This file includes the total busy time of disk from the system boot time. If $B_{Disk,\,T1}$ is the busy time of disk from the system boot time until the beginning of the experiment and $B_{Disk,\,T2}$ is the disk busy time from the system boot time till the end of the experiment, then disk total utilization is calculated using Eq. 3.

$$U_{Disk} = \frac{B_{Disk,T2} - B_{Disk,T1}}{T2 - T1} \quad [12] \qquad (3)$$

where T1 and T2 are the beginning and end times of the experiment, respectively.

To monitor the hardware performance events, we have used Oprofile/Xenoprof tool. Oprofile monitors particular hardware performance events using CPU performance counters. For instance, it can monitor the number of unhalted CPU clocks consumed by a particular binary image or symbol (function). Xenoprof is the modified version of Oprofile tool to support Xen. It is able to monitor hardware events for certain binaries or symbols inside Xen domains as well as Xen kernel. In our experiments, we have monitored three performance counters:

- **CPU_CLK_UNHALTED:** The number of CPU clocks outside of the halt state. It provides an approximate estimate of the CPU time consumed by a certain binary image or symbol.
- **INST_RETIRED:** The number of instructions which are retired. It is an approximate estimate of the number of instructions executed by a particular binary image or symbol.
- **LLC_MISSES:** The number of Last Level Cache (L3) misses. It measures the number of times that a particular binary image or symbol caused L3 miss and must access main memory.

4 Experimental Results

In this section, we present the results of different experiments we have performed. We have conducted the experiments in three different scenarios. Scenario I evaluates the performance degradation of a single web application running in Xen compared to native environment. The target of the scenario II is to elaborate the effect of increasing the CPU cores on the performance of a single Web application in Xen environment. Scenario III investigates the performance degradation of multiple consolidated Web applications running in Xen.

In all scenarios, we first show the performance difference in terms of the web application throughput and response time, then we elaborate the possible sources of this performance difference through the CPU and disk utilization, and most importantly, through the hardware events occurred at the system-level. It is necessary to mention that we have performed the experiments of the scenario I in native environment and Xen environment II, scenario II in Xen environments II and III, and scenario III in Xen environments III and IV, respectively; moreover, the duration of all experiments is 15 minutes and we repeated each experiment three times. The results reported are the average of the results from the experiments (The difference between the minimum and maximum of the results was less than 1%).

4.1 Scenario I: Performance profiling of a single Web application

In this section, we evaluate the performance impact of Xen on a single Web application. Fig. 5 and Fig. 6 illustrate the throughput and response time of the Web application as a function of workload that is the number of concurrent users served by the application in native and Xen environments. It is necessary to remind that Apache Olio Web application provides several functionalities to users and we have reported the results of the throughput and response time only for search by tag functionality herein because of the space limitations. As Fig. 5 illustrates, there is very little difference between the throughput of the Web application in native environment and Xen. This is due the fact that the system is not saturated and is able to process every request arriving to the system. Therefore, no request is lost in the system and the

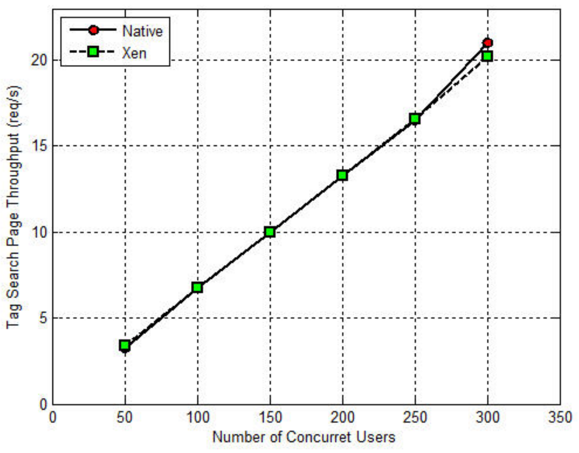

Fig. 5. Scenario I: Throughput for tag by search functionality

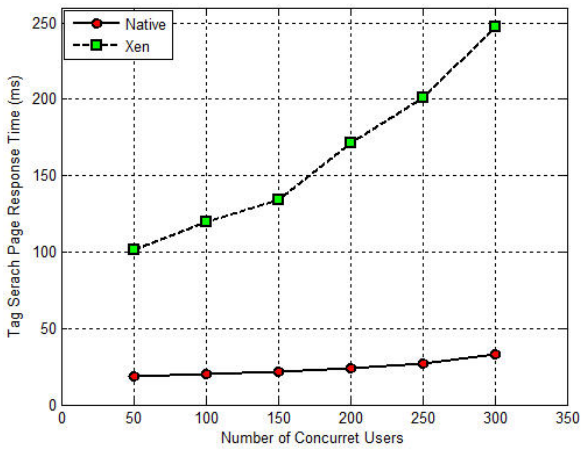

Fig. 6. Scenario I: Response time for tag by search functionality

throughput is equal to the arrival rate which is the same in both environments. However, there is remarkable difference between the response time of the Web application in Xen and native environment, as Fig. 6 shows. For 300 concurrent users, the response time in Xen is approximately six and one-half times of that in native environment. This indicates that the performance considerably degrades in Xen environment; furthermore, as the number of concurrent users increases, the response time in Xen environment grows more quickly compared to the native environment.

Fig. 7 and Fig. 8 demonstrate the disk utilization and CPU utilization per core as a function of workload for native and Xen environments, respectively. Fig. 7 shows that disk utilization is very low in both environments (at most 3.8% in Xen and 0.3% in native). This denotes that Web requests does not engage disk so much. Therefore, the performance degradation of Web application in Xen is not due to the more disk utilization of the system in Xen. Unlike disk utilization, CPU utilization is very high in Xen compared to the native environment as Fig. 8 illustrates. CPU utilization in Xen is almost two times of that in native environment which indicates that CPU is much more engaged in Xen than in native environment to process the Web requests. In addition, CPU utilization increases linearly in both environments as the number of concurrent users increases. However, the slope of increase in Xen is higher than the native environment. This denotes that as the workload increases, Xen leads to more overhead and this can be related to the higher response time and quicker increase of response time in Xen environment.

Now, we take a deeper look at the possible causes of higher response time of the Web application in Xen environment by providing the detailed analysis of the hardware performance counter values from Oprofile/Xenoprof tool. Fig. 9, Fig. 10, and Fig. 11 respectively show the aggregate CPU_CLK_UNHALTED, LLC_MISSES, and INST_RETIRED hardware performance counter values as a function of workload in Xen and native environment. As the figures show, all performance counter values in Xen are higher than native environment. The results indicate that total CPU clocks consumed by processes and total number of instructions executed by CPU are higher in Xen environment. This is consistent with the higher CPU utilization in Xen from Fig. 8; moreover, higher number of L3 cache misses in Xen environment is in agreement with the higher CPU consumption in Xen. This is due to the fact that more L3 cache misses causes more memory reference overhead and puts more pressure on the CPU.

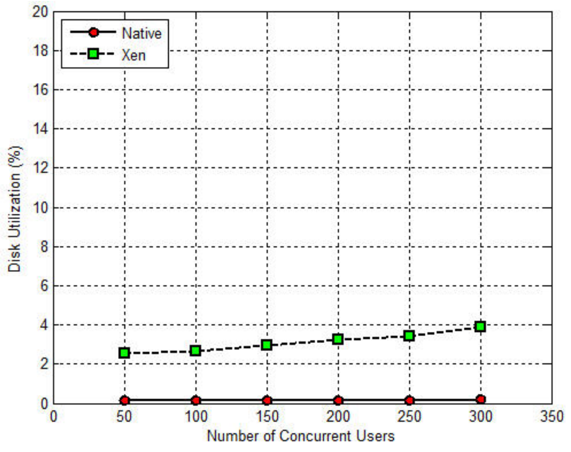

Fig. 7. Scenario I: Disk utilization

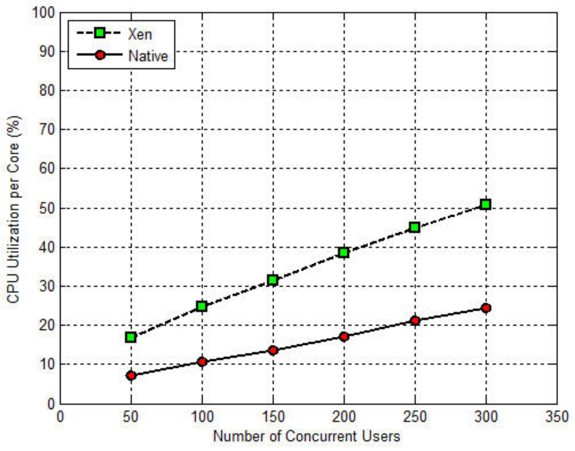

Fig. 8. Scenario I: CPU utilization

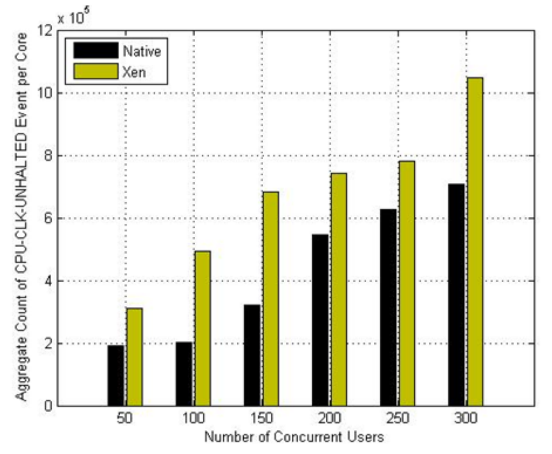

Fig. 9. Scenario I: Aggregate count of CPU_CLK_UNHALTED event

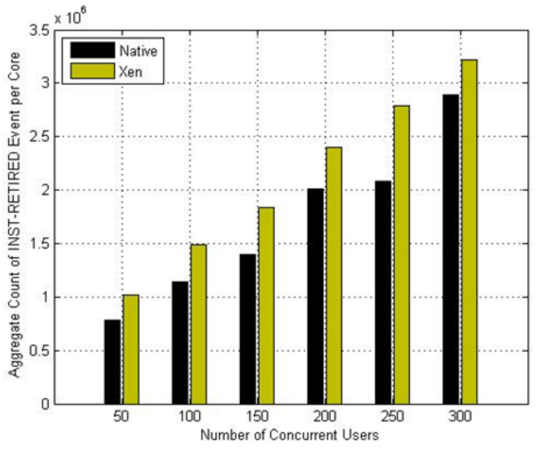

Fig. 10. Scenario I: Aggregate count of INST_RETIRED event

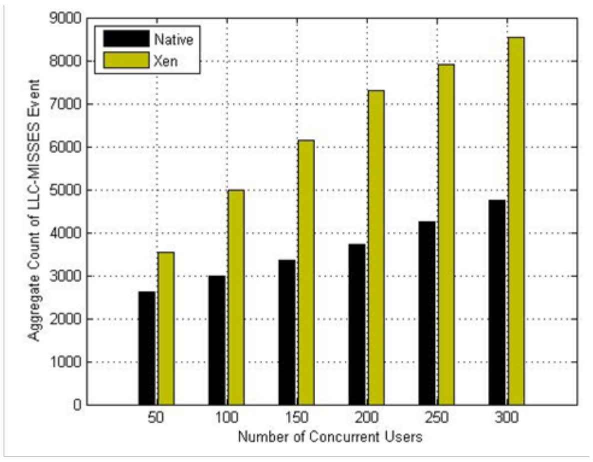

Fig. 11. Scenario I: Aggregate count of LLC_MISSES event

Table 1, Table 2, and Table 3 list the percentage of *CPU_CLK_UNHALTED* counter values for top five CPU consuming binaries in frontend, backend, and Dom0 domains of Xen, respectively. As the tables show, the percentage of CPU clocks consumed by kernel is low in frontend and backend domains (1.6% in frontend and 6.8% in backend) but this value is high in Dom0 (37.1%).

Binary Name	Percentage of CPU Clocks
php-fpm	73.9
nginx	22.6
kernel	1.6
oprofiled	0.6
xennet	0.4

Table 1. Scenario I: Percentage of CPU_CLK_UNHALTED values for top five binaries in frontend

Binary Name	Percentage of CPU Clocks
mysqld	87.4
kernel	6.8
java	2.1
oprofiled	0.9
xennet	0.8

Table 2. Scenario I: Percentage of CPU_CLK_UNHALTED values for top five binaries in backend

Binary Name	Percentage of CPU Clocks
kernel	37.1
netback	9.1
bridge	4.4
tg3	4.9
ip_tables	1.1

Table 3. Scenario I: Percentage of CPU_CLK_UNHALTED values for top five binaries in Dom0

Tables 4, 5, and 6 denote the percentage of retired instructions for top five binaries that executed most CPU instructions in frontend, backend and Dom0. The results are similar to the results for CPU_CLK_UNHALTED counter values. In frontend and backend domains, kernel executes very low number of instructions (1.2% and 2.3%) on CPU whereas it executes high number of instructions in Dom0 (30.9%).

Binary Name	Percentage of Retired Instructions
php-fpm	81.3
nginx	15.1
kernel	1.2
modprobe	1.1
oprofiled	0.4

Table 4. Scenario I: Percentage of INST_RETIRED counter values for top five binaries in frontend

Binary Name	Percentage of Retired Instructions
mysqld	94.4
kernel	2.3
java	1.7
oprofiled	0.4
xennet	0.3

Table 5. Scenario I: Percentage of INST_RETIRED counter values for top five binaries in backend

Binary Name	Percentage of Retired Instructions
kernel	30.9
bridge	7.1
netback	6.3
tg3	2.5
ip_tables	2.4

Table 6. Scenario I: Percentage of INST_RETIRED counter values for top five binaries in Dom0

Table 7, Table 8, and Table 9 show the percentage of LLC_MISSES counter values for top five binaries which caused most L3 cache misses in frontend, backend and Dom0. The kernels of frontend and backend domains cause relatively low number of L3 cache misses (4.5% and 10.8%) while the kernel of Dom0 causes high number of cache misses (29%).

It is worth describing the functionality of some binaries listed in the tables. *xennet* in frontend and backend domains and netback in Dom0 handle the network related events. *bridge* is a software entity in Dom0 that connects the virtual network interfaces of frontend, backend, and Dom0 to the physical network interface. *tg3* is the Broadcom Ethernet card driver installed in Dom0.

Binary Name	Percentage of L3 Cache Misses
php-fpm	63.8
nginx	27
kernel	4.5
xennet	0.9
modprobe	0.4

Table 7. Scenario I: Percentage of LLC_MISSES counter values for top five binaries in frontend

Binary Name	Percentage of L3 Cache Misses
mysqld	74.6
kernel	10.8
java	9.2
xennet	2.8
pcscd	1

Table 8. Scenario I: Percentage of LLC_MISSES counter values for top five binaries in backend

Binary Name	Percentage of L3 Cache Misses
kernel	29
netback	15.7
tg3	2.9
bridge	1.6
python	1.5

Table 9. Scenario I: Percentage of LLC_MISSES counter values for top five binaries in Dom0

Now, we consider the Xen kernel itself. Table 10 denotes the percentage of hardware performance counter values for Xen kernel. As the table results show, Xen VMM consumes low number of CPU clocks and executes low number of instructions on CPU whereas it causes relatively high number of L3 cache misses. Xen kernel and Dom0 which handles the I/O of other domains are overhead sources in Xen virtualized environment. In addition, the kernel of Dom0 consumes the most CPU clocks and causes the most L3 cache misses in Dom0. Therefore, we focus on the Xen and Dom0 kernels to get much more insight into the main sources of performance degradation in Xen environment. Tables 11 through 14 list the top five functions of Xen and Dom0 kernels that consumes the most CPU clocks and causes most L3 cache misses.

Performance Counter	Percentage for Xen Kernel
CPU_CLK_UNHALTED	4.9
INST_RETIRED	6.1
LLC_MISSES	11.6

Table 10. Scenatio I: Percentage of performance counter values for Xen kernel

Symbol Name	Percentage of CPU Clocks
_spin_lock	6.5
page_get_owner_and_reference	4.4
flush_area_local	3.8
_set_status	3.5
csched_schedule	3.3

Table 11. Scenario I: Percentage of CPU_CLK_UNHALTED counter values for top five functions in Xen kernel

Symbol Name	Percentage of L3 cache misses
set_status	12.2
evtchn_set_pending	12.2
memcpy	11
_put_page_type	10.4
page_get_owner_and_reference	8.7

Table 12. Scenario I: Percentage of LLC_MISSES counter values for top five functions in Xen kernel

Symbol Name	Percentage of CPU Clocks
hypercall_page	4.1
kmem_cache_zalloc	3.9
nf_iterate	3.8
evtchn_do_upcall	3.6
_copy_from_user_ll	2.9

Table 13. Scenario I: Percentage of CPU_CLK_UNHALTED counter values for top five functions in Dom0 kernel

Symbol Name	Percentage of L3 cache misses
skb_copy_bits	13.7
kmem_cache_zalloc	5.6
spin_lock_irqsave	4.8
netif_rx	4.5
evt_chn_do_upcall	4

Table 14. Scenario I: Percentage of LLC_MISSES counter values for top five functions in Dom0 kernel

We observe that the most CPU consuming and the highest cache miss functions of Xen kernel relates to the scheduling of domains by Xen. For instance*, _spin_lock* function implements the spinlock mutual exclusion mechanism to schedule virtual CPUs on physical CPUs, *page_get_owner_and_reference* function gets domain which is the owner of a particular memory page, and *set_status* function set the status of a particular domain. The functions of Dom0 kernel are related to the processing the I/O of DomUs by Dom0. For instance, *skb_copy_bits* copies data from socket buffer to kernel buffer or *_copy_from_user_ll* copies data from user-mode memory pages to kernel mode memory pages.

In conclusion, the major sources of performance degradation of a Web application in Xen environment relates to more CPU clocks that consumed by processes, more instructions executed by CPU, and most importantly, more L3 cache misses that occurred at the system in this environment; moreover, the most percentage of CPU clocks and L3 cache misses in Xen and Dom0 kernels are associated with the functions that schedule the domains on physical CPUs and perform the I/O operations of DomUs.

4.2 Scenario II: The effect of increasing CPU cores

In this section, we evaluate the impact of increasing CPU cores on the performance of a single Web application in Xen environment. Fig. 12 and Fig. 13 show the throughput and response time of the Web application for search by tag functionality in dual core and quad core configurations of Xen environment, respectively. As we can see, the throughput is equal in two configurations but the response time in quad core configuration is almost 11 percent lower compared to dual core configuration. This indicates that increasing CPU cores in Xen environment has no effect on the throughput of the Web application but it decreases its response time.

Fig. 14 and Fig. 15 illustrate the disk and CPU utilizations of the system for dual core and quad core configurations, respectively. Both configurations have the same disk utilization as we expect. However, the CPU utilization in quad core configuration is approximately half of that in dual core configuration. This is because the degree of concurrency in quad core configuration is higher than that in dual core configuration. This allows Xen to schedule the domains on more physical CPU cores concurrently and leads to lower CPU utilization per core.

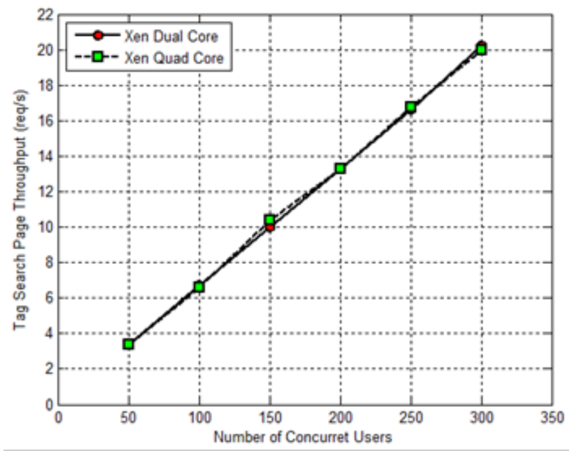

Fig. 12. Scenario II: Throughput for tag by search functionality

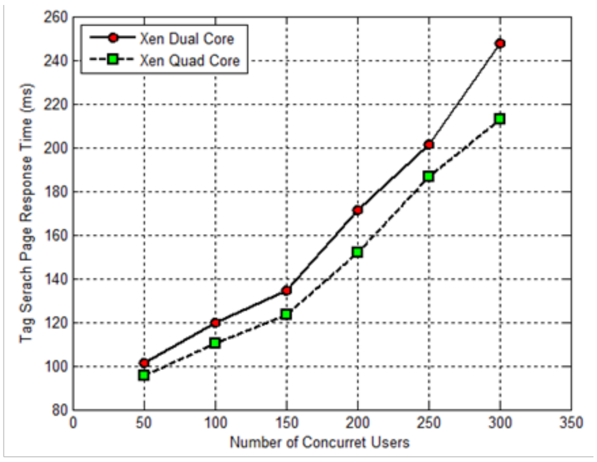

Fig. 13. Scenario II: Response time for tag by search functionality

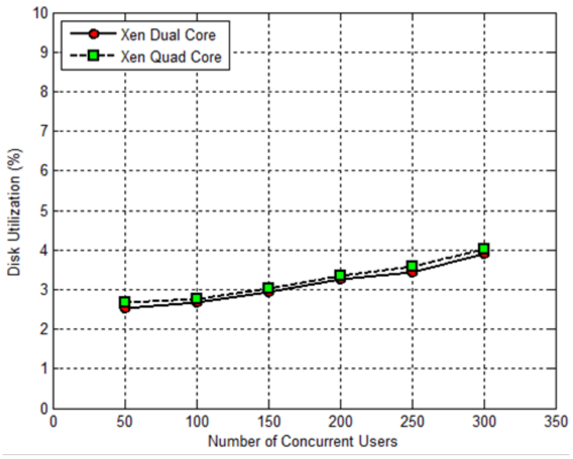

Fig. 14. Scenario II: Disk utilization

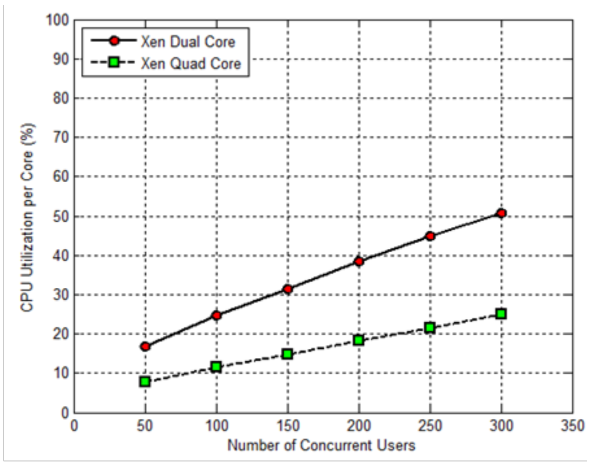

Fig. 15. Scenario II: CPU utilization

Now, we evaluate the possible causes of response time reduction in quad core configuration in more detail by comparing the performance event counter values of two configurations. Fig. 16 and Fig. 17 denote the aggregate CPU_CLK_UNHALTED and INST_RETIRED performance counter values for these configurations. As we can observe, the number of CPU clocks consumed and the number of instructions executed per CPU core is lower in quad core configuration. This is in agreement with the lower CPU utilization of this configuration from Fig. 15.

Fig. 18 shows the aggregate count of LLC_MISSES performance counter value for two configurations. As the figure illustrates, quad core configuration leads to less L3 cache misses compared to dual core configuration. This is due the fact that the number of domains is less than the number of physical cores in quad core configuration. Therefore, Xen can schedule and run any domain on physical cores without needing to suspend the other domains. This leads to less context switches, and as a result, less L3 cache misses. Unlike quad core configuration, in dual core configuration, the number of domains is more compared to the number of physical cores. Consequently, Xen can run only two domains concurrently and it must suspend one of them if it requires to run another domain. This causes much more context switches and L3 cache misses.

In summary, increasing the number of physical cores in Xen environment decreases the response time of the Web application because it allows Xen to run domains more concurrently, which leads to less CPU clocks consumed per core, and most importantly, less L3 cache misses.

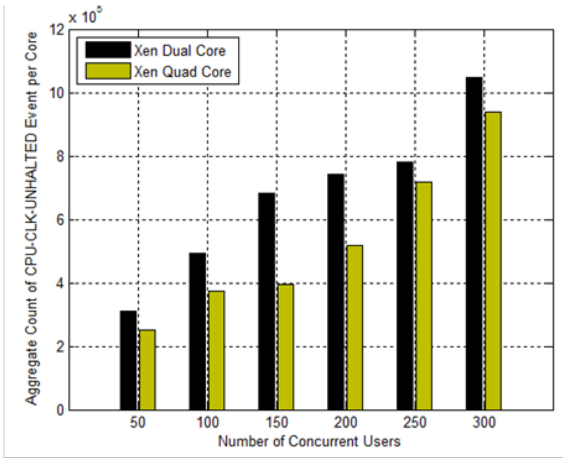

Fig. 16. Scenario II: Aggregate count of CPU_CLK_UNHALTED event

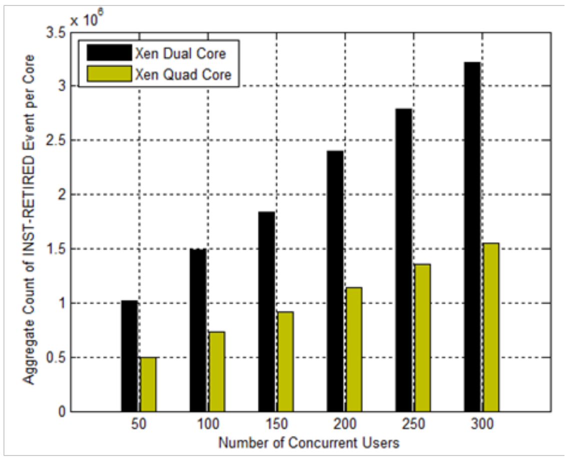

Fig. 17. Scenario II: Aggregate count of INST_RETIRED event

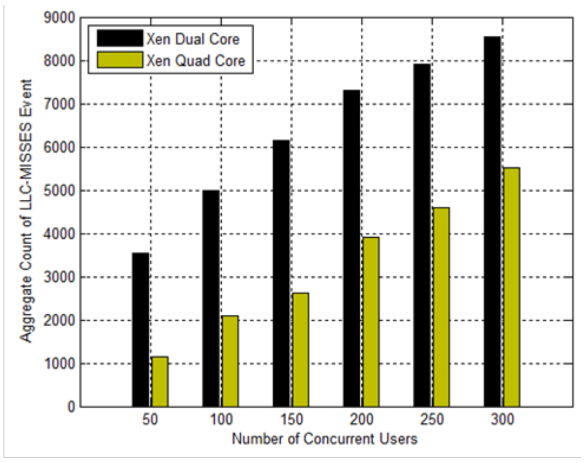

Fig. 18. Scenario II: Aggregate count of LLC_MISSES event

4.3 Scenario III: Performance profiling of multiple consolidated Web applications

In this section, we assess the performance of multiple consolidated Web applications in Xen environment. Here, not only Xen but also the functionality of the domains running other Web applications can have an impact on the performance of a particular Web application. Fig. 19 compares the throughput of the Web application in Xen unconsolidated environment with the throughput of the Web applications in Xen consolidated environment for tag by search functionality. As we can see, there is very little difference between the throughput of Web applications. Fig. 20 compares the response time of the Web applications in these two environments. As the figure shows, the response time of consolidated Web applications is much higher than the response time of unconsolidated Web application. For 250 concurrent users, the average response time of Web applications in consolidated environment is almost two and a half times of that in unconsolidated environment.

Fig. 21 illustrates the disk utilization of the system for two environments. We expected that the disk utilization in consolidated environment would be almost four times of that in unconsolidated one because we quadrupled the number of concurrent Web applications in consolidated environment. However, as the figure denotes, the disk utilization is much more higher in consolidated environment. For 250 concurrent users, it is approximately eight times higher compared to the unconsolidated environment. This indicates that one of the causes of higher response time in Xen consolidated environment is the higher disk utilization of this environment. Fig. 22 compares the CPU utilization of two environments. The CPU utilization in consolidated environment is approximately four times of that in unconsolidated environment as expected.

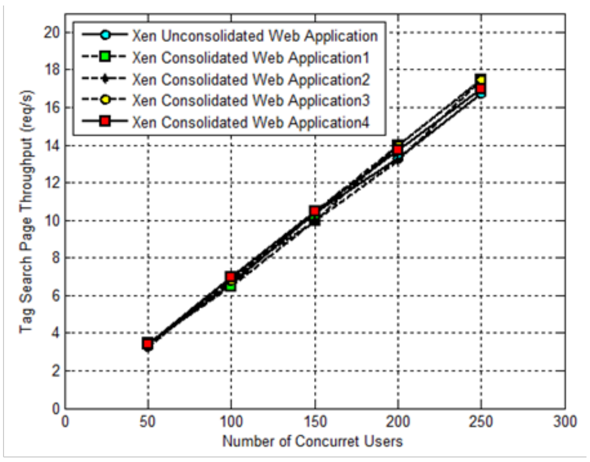

Fig. 19. Scenario III: Throughput for tag by search functionality

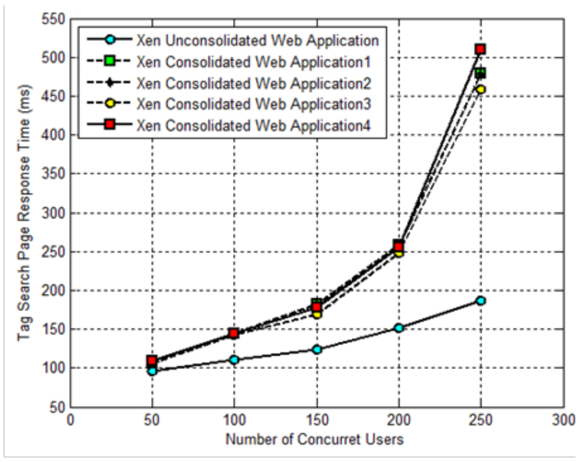

Fig. 20. Scenario III: Response time for tag by search functionality

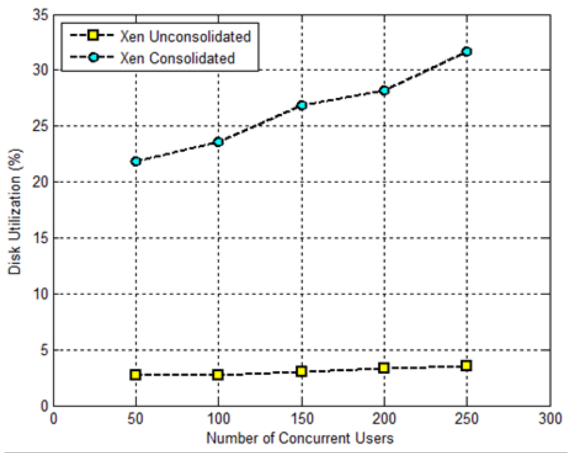

Fig. 21. Scenario III: Disk Utilization

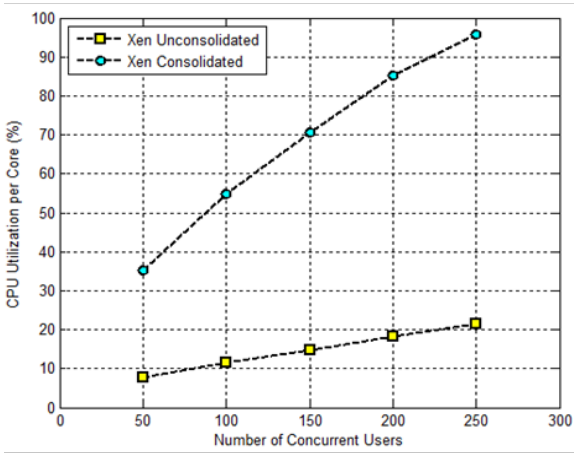

Fig. 22. Scenario III: CPU Utilization

Fig. 23 and Fig. 24 illustrate the aggregate *CPU_CLK_UNHALTED* and *INST_RETIRED* performance counter values for two environments, respectively. The value of both counters in consolidated environment is almost four times of that in unconsolidated one. This is in consistent with the results of CPU utilization from Fig. 22.

Fig. 25 shows the *LLC_MISSES* performance counter value for two environments. As the figure illustrates, the number of L3 cache misses is much more higher in consolidated environment. For 250 concurrent users, the L3 cache misses in consolidated environment is almost eighteen times of that in unconsolidated one. This denotes that another cause of higher response time in consolidated environment is very high L3 cache misses in this environment. The higher cache misses in consolidated environment comes back to the fact that the number of running domains is more than the number of physical cores (nine domains vs. four physical cores) in this environment. This makes Xen suspend some running domains in order to run the other domains and leads to more context switching, and consequently, more L3 cache misses. This results are consistent with the results of disk utilization from Fig. 21 because more disk utilization means more memory accesses and more L3 cache misses.

Table 15 lists the percentage of performance event counter values for Xen kernel. As the table shows, about five percent of counter values relates to Xen kernel. Now, we look at the functions that have the most proportions of the performance event counter values in Xen and Dom0 kernel. Tables 16 through 19 list the top five functions of Xen and Dom0 kernel that consumed most CPU clocks and caused the most cache misses. According to the Table 16, function *spin_lock* that implements the spinlock mutual exclusion mechanism to schedule virtual cores on physical cores has consumed the most CPU clocks in Xen kernel. In addition, as the Table 17 shows, functions *page_get_owner_and_reference*, *memcpy*, and *put_page_type* caused totally about thirty-five percent of L3 cache misses in Xen kernel. These functions are engaged to handle the I/O of frontend and backend domains. Moreover, as we can see in Table 19, function *skb_copy_bits* leads to almost twenty percent of L3 cache misses in Dom0 kernel. This function also is engaged to process the I/O operations of frontend and backend domains.

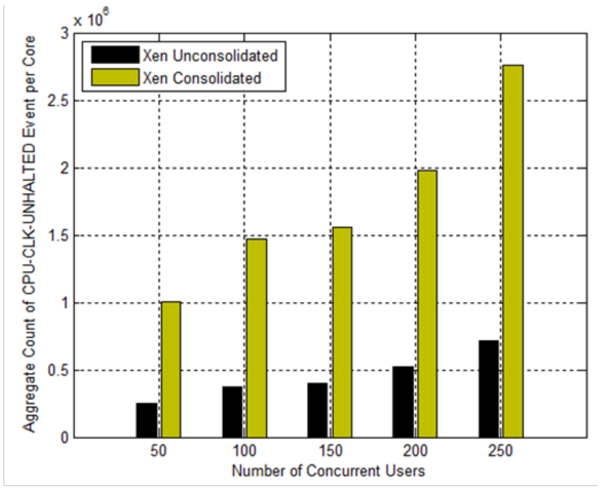

Fig. 23. Scenario III: Aggregate count of CPU_CLK_UNHALTED event

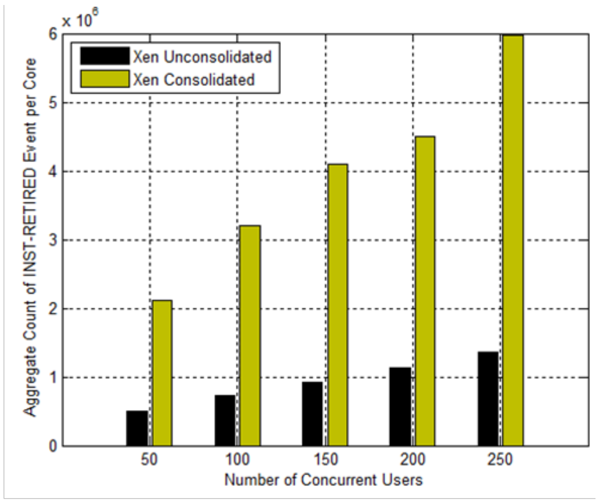

Fig. 24. Scenario III: Aggregate count of INST_RETIRED event

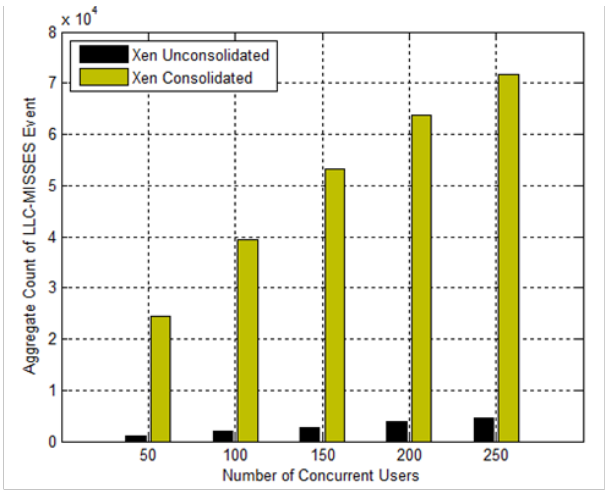

Fig. 25. Scenario III: Aggregate count of LLC_MISSES event

Performance Counter	Percentage for Xen Kernel
CPU_CLK_UNHALTED	4.8
INST_RETIRED	4.6
LLC_MISSES	5.8

Table 15. Scenatio III: Percentage of performance counter values for Xen kernel

Symbol Name	Percentage of CPU Clocks
spin_lock	7.4
page_get_owner_and_reference	6.3
gnttab_map_grant_ref	3.9
csched_schedule	3.3
flush_area_local	3.1

Table 16. Scenario III: Percentage of CPU_CLK_UNHALTED counter values for top five functions in Xen kernel

Symbol Name	Percentage of L3 cache misses
page_get_owner_and_reference	13.2
memcpy	11.4
put_page_type	10.7
set_status	8.9
evtchn_set_pending	5.5

Table 17. Scenario III: Percentage of LLC_MISSES counter values for top five functions in Xen kernel

Symbol Name	Percentage of CPU Clocks
hypercall_page	5.5
evt_chn_do_upcall	5.4
kmem_cache_zalloc	4.8
nf_iterate	4.4
spin_lock_irqsave	3.9

Table 18. Scenario III: Percentage of CPU_CLK_UNHALTED counter values for top five functions in Dom0 kernel

Symbol Name	Percentage of L3 cache misses
skb_copy_bits	19.1
copy_skb_header	7.4
put_page	6.5
spin_lock_irqsave	6.1
swiotlb_unmap_single	5.8

Table 19. Scenario III: Percentage of LLC_MISSES counter values for top five functions in Dom0 kernel

In summary, when multiple Web applications are consolidated in Xen environment, not only high CPU utilization but also high disk utilization of the system cause the performance degradation of the Web applications. In addition, detailed analysis of the performance counter values indicates that the number of L3 cache misses is much more higher in consolidated environment compared to unconsolidated one. It is another cause of performance degradation in Xen consolidated environment. Moreover, the functions which are responsible for handling the I/O operations of frontend and backend domains cause the most L3 cache misses in Xen and Dom0 kernels.

5 Conclusion

In this paper, we provided in-depth performance analysis and profiling of the Web application(s) in Xen environment to recognize the major sources of performance degradation in Xen. For different scenarios, we first measured the throughput and response time of the web application(s) to illustrate the performance degradation. Meanwhile, we monitored the CPU and disk utilizations of the system as well as the hardware performance events occurred at the system.

We illustrated that the main causes of performance degradation of a single Web application in Xen compared to native environment are the higher CPU clocks consumed by processes, higher instructions executed by CPU, and most importantly, higher L3 cache misses in Xen environment. Furthermore, the most proportion of the performance event counter values in Xen and Dom0 kernel relates to the functions that schedule the domains and process the I/O operations of the frontend and backend domains. In addition, we indicated that by increasing the physical CPU cores of the system, performance degradation decreases in Xen environment because it leads to higher concurrency degree and less L3 cache misses. We also showed that in Xen consolidated environment, the performance of the Web applications degrades dramatically because not only CPU utilization but also disk utilization considerably increases in this environment. Moreover, very high number of L3 cache misses is another cause of performance degradation in this environment. Most of these cache misses in Xen and Dom0 kernel are associated with the functions that handle the I/O operation of the backend and frontend domains.

As a next step, we intend to evaluate and profile the performance of Web application in other virtualized environments such as those based on VMWare, KVM, or OpenVZ and compare the results. Moreover, we can repeat our experiments for other configurations including pinning virtual cores to physical cores or assigning more virtual cores to the backend and frontend domains and assess the effect of them on the performance of Web application. Most importantly, we identified the functions of Xen and Dom0 kernels which have the most contribution to the performance degradation of Web application. We plan to optimize some of these functions and make Xen environment more appropriate for running Web applications.

References

1. Barham P, Dragovic B, Fraser K, et al. (2003) Xen and the art of virtualization. ACM SIGOPS Oper Syst Rev 37:164–177. DOI: 10.1145/1165389.945462.
2. VMware. Available online at: http://www.vmware.com/, (accessed on 08.20.2017).
3. Kivity A, Kamay Y, Laor D, et al. kvm: the Linux Virtual Machine Monitor, 2007.
4. Kim S, Eom H, Yeom H (2013) Virtual machine consolidation based on interference modeling. J Supercomput 66:1489–1506. DOI: 10.1007/s11227-013-0939-2.

5. Benevenuto F, Fernandes C, Santos M, et al. (2006) Performance models for virtualized applications. In: Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). pp 427–439
6. Chiang RC, Huang HH (2011) TRACON: Interference-aware scheduling for data-intensive applications in virtualized environments. Int Conf of High Perform Comput Networking, Storage Anal (SC), 2011, pp 1–12.
7. Koh Y, Knauerhase R, Brett P, et al. (2007) An Analysis of Performance Interference Effects in Virtual Environments. Perform Anal Syst Software, 2007 ISPASS 2007 IEEE Int Symp 200–209. DOI: 10.1109/ISPASS.2007.363750.
8. Kundu S, Rangaswami R, Dutta K, Zhao M (2010) Application performance modeling in a virtualized environment. High Perform Comput Archit (HPCA), 2010 IEEE 16th Int Symp 1–10. DOI: 10.1109/HPCA.2010.5463058.
9. Galán F, Fernández D, Fuertes W, et al. (2009) Scenario-based virtual network infrastructure management in research and educational testbeds with VNUML. Ann Telecommun - Ann des télécommunications 64:305–323. DOI: 10.1007/s12243-009-0104-3.
10. Che J, Shi C, Yu Y, Lin W (2010) A Synthetical Performance Evaluation of OpenVZ, Xen and KVM. In: Proc. 2010 IEEE Asia-Pacific Serv. Comput. Conf. IEEE Computer Society, Washington, DC, USA, pp 587–594. DOI: 10.1109/APSCC.2010.83.
11. Tikotekar A, Vallée G, Naughton T, et al. (2009) An Analysis of HPC Benchmarks in Virtual Machine Environments. In: César E, Alexander M, Streit A, et al. (eds) Euro-Par 2008 Work. - Parallel Process. SE - 8. Springer Berlin Heidelberg, pp 63–71. DOI: 10.1007/978-3-642-00955-6_8.
12. NasiriGerdeh, R., Hosseini, N., RahimiZadeh, K., & AnaLoui, M. (2015, October). Performance analysis of Web application in Xen-based virtualized environment. In Computer and Knowledge Engineering (ICCKE), 2015 5th International Conference on (pp. 256-261). IEEE. DOI: 10.1109/ICCKE.2015.7365837
13. Cherkasova L, Gardner R (2005) Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor. In: Proc. Annu. Conf. USENIX Annu. Tech. Conf. USENIX Association, Berkeley, CA, USA, p 24.
14. Chung, H., & Nah, Y. (2017, March). Performance Comparison of Distributed Processing of Large Volume of Data on Top of Xen and Docker-Based Virtual Clusters. In International Conference on Database Systems for Advanced Applications (pp. 103-113). Springer, Cham. DOI: 10.1007/978-3-319-55753-3_7
15. Voron, G., Thomas, G., Quéma, V., & Sens, P. (2017, April). An interface to implement NUMA policies in the Xen hypervisor. In EuroSys (pp. 453-467)
16. Patel, M., Chaudhary, S., & Garg, S. (2016, April). Performance modeling of skip models for VM migration using Xen. In Computing, Communication and Automation (ICCCA), 2016 International Conference on (pp. 1256-1261). IEEE. DOI: 10.1109/CCAA.2016.7813909
17. Kaneko, Y., Ito, T., & Hara, T. (2016, September). A measurement study on virtualization overhead for applications of industrial automation systems. In Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on (pp. 1-8). IEEE. DOI: 10.1109/ETFA.2016.7733507
18. Menon A, Santos JR, Turner Y, et al. (2005) Diagnosing Performance Overheads in the Xen Virtual Machine Environment. In: Proc. 1st ACM/USENIX Int. Conf. Virtual Exec. Environ. ACM, New York, NY, USA, pp 13–23. DOI: 10.1145/1064979.1064984.
19. Oprofile: A system profiler for Linux, http://oprofile.sourceforge.net/, (accessed on 08.20.2017).
20. Menon A, Cox AL, Zwaenepoel W (2006) Optimizing Network Virtualization in Xen. In: Proc. Annu. Conf. USENIX '06 Annu. Tech. Conf. USENIX Association, Berkeley, CA, USA, p 2.
21. Du J, Sehrawat N, Zwaenepoel W (2011) Performance Profiling of Virtual Machines. SIGPLAN Not 46:3–14. DOI: 10.1145/2007477.1952686.
22. Bellard F (2005) QEMU, a Fast and Portable Dynamic Translator. In: Proc. Annu. Conf. USENIX Annu. Tech. Conf. USENIX Association, Berkeley, CA, USA, p 41.
23. Padala P, Zhu X, Wang Z, et al. (2007) Performance evaluation of virtualization technologies for server consolidation, HP Labs Tec. Report.
24. OpenVZ: http://openvz.org/, (accessed on 08.20.2017).
25. Jang J-W, Seo E, Jo H, Kim J-S (2012) A low-overhead networking mechanism for virtualized high-performance computing systems. J Supercomput 59:443–468. DOI: 10.1007/s11227-010-0444-9.
26. Pu X, Liu L, Mei Y, et al. (2013) Who is your neighbor: Net I/O performance interference in virtualized clouds. IEEE Trans Serv Comput 6:314–329. DOI: 10.1109/TSC.2012.2.

27. Mei Y, Liu L, Pu X, et al. (2013) Performance Analysis of Network I/O Workloads in Virtualized Data Centers. Serv Comput IEEE Trans 6:48–63. DOI: 10.1109/TSC.2011.36.
28. Zhang, W., Shi, Y., Zheng, Y., Liu, L., & Cui, L. (2017, January). Resource and performance prediction at high utilization for N-Tier cloud-based service systems. In Proceedings of the Australasian Computer Science Week Multiconference (p. 43). ACM. DIO: 10.1145/3014812.3014857
29. Li, Z., Zhang, B., Ren, S., Liu, Y., Qin, Z., Goh, R. S. M., & Gurusamy, M. (2017, May). Performance Modelling and Cost Effective Execution for Distributed Graph Processing on Configurable VMs. In Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (pp. 74-83). IEEE Press. DIO: 10.1109/CCGRID.2017.85
30. Patel, M., Chaudhary, S., & Garg, S. (2016, April). Performance modeling of skip models for VM migration using Xen. In Computing, Communication and Automation (ICCCA), 2016 International Conference on (pp. 1256-1261). IEEE. DIO: 10.1109/CCAA.2016.7813909
31. Dehsangi, M., Asyabi, E., Sharifi, M., & Azhari, S. V. (2015, August). cCluster: a core clustering mechanism for workload-aware virtual machine scheduling. In Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on (pp. 248-255). IEEE. DOI: 10.1109/FiCloud.2015.56
32. Menascé DA (2005) Virtualization: Concepts, Applications, and Performance Modeling. Proc Int Conf Comput Meas Group's. DOI: 10.1.1.61.6680.
33. Hong C-H, Kim B-J, Kim Y-P, et al. (2014) Performance Prediction and Evaluation of Parallel Applications in KVM, Xen, and VMware. In: Silva F, Dutra I, Santos Costa V (eds) Euro-Par 2014 Parallel Process. SE-9. Springer International Publishing, pp 99–110. DOI: 10.1007/978-3-319-09873-9_9.
34. RahimiZadeh K, Nasiri Gerde R, AnaLoui M, Kabiri P (2015) Performance evaluation of Web server workloads in Xen-based virtualized computer system: analytical modeling and experimental validation. Concurr Comput Pract Exp. DOI: 10.1002/cpe.3454.
35. Sun Microsystems: Project Faban, http://faban.sunsource.net, (accessed on 08.20.2017).
36. Sobel W, Subramanyam S, Sucharitakul A, et al. (2008) Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0. Proc of the 1st Workshop on Cloud Computing.
37. Apache Software Foundation: Olio, http://incubator.apache.org/olio, (accessed on 08.20.2017).