Evaluating Methods for Improving Efficiency in Xen Virtual Machine CPU Scheduling

James Devine

December 6, 2009

Abstract

Operating system virtualization is a technique for running multiple operating systems on a single host machine. This is done though a lightweight operating system called a hypervisor that allocates resources to each guest operating system. Xen is one such hypervisor that is both open source and free. The Xen hypervisor has changed greatly since its inception in 2003 and it currently contains three different virtual machine scheduling algorithms. Yet, justification for the changes in scheduling algorithms is missing from the Xen documentation. This is quite interesting given the fact that virtual machine performance can be greatly impacted by the hypervisor's scheduling of CPU resources. The proposed research seeks to determine if particular schedulers will perform better for specific work loads and proposes three case study applications for determining and evaluating the performance of each of the three scheduling algorithm for each of the three workloads.

1 Introduction

Virtualization is a software technique that allows for multiple operating systems to be run concurrently on a single computer. The concept of virtualization began many years ago with the advent of the IBM 360 operating system. Within the last five years virtualization has become increasingly popular due to rapidly dropping hardware prices along with increasing speed. This has led to the fact that on average computers are operating with a very small workload that only uses 10%-20% of hardware resources.

The trend of having unused resources is a major concern of businesses running multiple servers. If eight servers are running at 10% average utilization then they could be consolidated to a single server with an 80% utilization. The motivation of server consolidation has caused a rapid increase in the use of virtualization by businesses. Aside from the cost savings, there are also several other benefits to virtualization that include easy

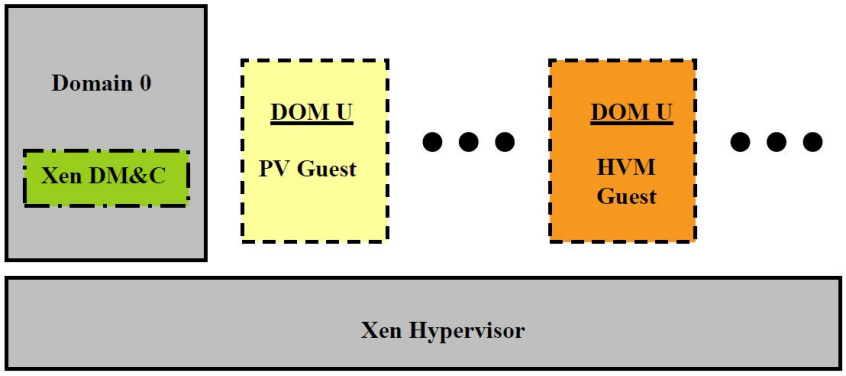

Figure 1: **An Abstract View of The Xen Architecture**. The Xen Hypervisor is the underlying software that directly accesses the CPU and memory. Domain 0 is a special virtual machine that has direct access to the physical resources of the host. Two guest virtual machines are shown to the right of Domain 0, one paravirtualized (PV) and one fully virtualized (HVM) [9].

reallocation of resources and decreased power consumption.

Xen is an open source hypervisor that was first released in 2003 [1]. A hypervisor, also referred to as a virtual machine monitor (VMM), is a lightweight kernel that provides an abstraction of physical resources to virtual machines. Initially, Xen only supported paravirtualization, which runs virtual machines with a modified version of the host Linux kernel. This has changed within the last few years and Xen now supports full virtualization which enables Windows and most all other operating systems to be virtualized [9].

The Xen architecture, depicted in Figure 1, is different from that of Microsoft and VMware in that I/O is not handled by the hypervisor. The Xen hypervisor is primarily responsible for allocating the CPU and memory and has direct access to the physical hardware. All disk and network I/O is handled though a special virtual machine referred to as domain 0. Domain 0 is the only virtual machine that has direct access to the physical hardware resources. All of the other virtual machines (referred to as domains) access disk, network, universal serial bus (USB) devices, and all other peripherals though Domain 0.

The Xen hypervisor can run one of three scheduling algorithms: borrowed virtual

time, atropos, and simple earliest deadline first. Each scheduling algorithm is real-time; real-time scheduling algorithms are used because each virtual machine must have a low response time for the guest operating systems to run efficiently. The **borrowed virtual time** algorithm monitors thread execution time in terms of virtual time and dispatches the runnable thread with the earliest effective virtual time. Latency-sensitive threads, which require a low turn-around time, are allowed to "warp" back in virtual time to make them appear earlier, allowing threads to gain dispatch preference [3]. The **atropos** algorithm provides guarantees about absolute shares of the CPU and has a facility for sharing slack CPU time on a best-effort basis [7]. The **simple earliest deadline first** algorithm places processes in a priority queue and executes the processes closest to their deadlines first [4].

2 Prior Work

This is one of the first studies that seeks to compare Xen virtual machine scheduling algorithm performance with case study applications. In fact, the case study workloads that will be created are an initial step towards developing platform independent virtual system benchmarks. Most of the prior work conducted deals with each individual scheduling algorithm and establishing that CPU scheduling can affect performance [2] [13].

3 Purpose

The purpose of this work is to explore the effect of different scheduling algorithms on particular types of workloads. To this end, the hypothesis for the proposed research is:

Hypothesis

An efficient virtual machine scheduling algorithm is important for increased throughput and decreased response time. Given varying workloads, there will be some schedulers that are more efficient at scheduling virtual machines for particular types of workloads. Taking this into account, it is possible to fine tune the Xen hypervisor to maximize throughput and minimize response time with specific types of workloads.

Method of Approach

This project will develop and explore virtual machine workloads in order to evaluate the hypothesis by conducting the following steps:

- Create three case-study workloads that can be used in future research.
- Determine the performance of each scheduling algorithm for each case-study workload using throughput and response time as metrics.
- Analyze the results to determine whether scheduling algorithm performance is workload specific and if so which scheduling algorithm is best suited for each of the experimental workloads.

Specifically, this work will develop a set of three case study workloads. The workloads will be platform independent and able to run on any number of other hypervisors such as those offered by Microsoft and VMware. Two of the case study workloads will be based on open source software allowing the entire collection of virtual machines to be released for use in future work. This is significant due to the fact that no such workloads currently exist.

The results of running a performance study with each case study workload and each scheduling algorithm will provide insight into the impact of the virtual machine scheduler on the overall performance of a virtual machine system. Further, the work can yield some best practices that can be generalized into customization of the Xen scheduler for differing workloads. Such best practices could lead to install time optimization of the Xen hypervisor based on the intended workload.

4 Implementation and methodology

The project will be broken up into four parts: background research, designing the case study workloads, conducting an experimental performance study, and analyzing the results.

Research

An in-depth investigation into the Xen architecture and scheduling algorithms is the first step. The research will provide a solid ground for the rest of the project to run smoothly and efficiently. A hardware proposal and acquisition will be included in the research phase.

Designing Case Study Workloads

Three case study workloads will be created to test the efficiency of each Xen scheduling algorithm for each of the workloads. Efficiency will be measured in terms of throughput (higher is better) and response time (lower is better). The proposed workloads are described below.

Workload I: Web Application Case Study

The web application workload will replicate a very simple web application that uses MySQL, Apache, and NFS (Network File System) servers. Each server will be separate so there is transfer of data between multiple virtual machines as they work together to serve web content. The actual web application will be written in PHP (PHP: Hypertext Preprocessor) and perform a number of representative SQL (Structured Query Language) queries. The number of clients will be simulated using httperf [8] so different workloads can be tested. The proposed virtual machine architecture is depicted in Figure 2.

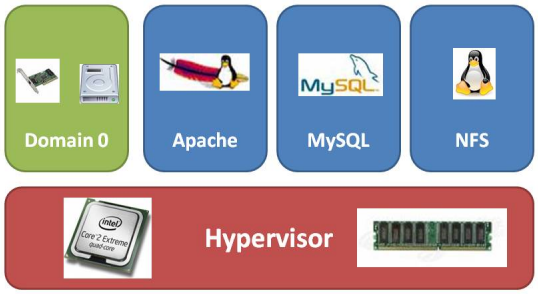

Figure 2: Web Application Case Study Workload Architecture

Workload II: Virtual Private Server (VPS) Hosting Case Study

The VPS hosting case study will simulate a workload that is present in virtual shared hosting environments. In such environments, each user is given root access to a virtual machine in which any number of installed applications can be run. For this workload case study, each VPS will be generalized as a single instance of Apache and MySQL. Each VPS will run the same web application from Workload I. The proposed virtual machine architecture is depicted in Figure 3.

Figure 3: VPS Hosting Case Study Workload Architecture

Workload III: Windows Domain Case Study

The Windows Domain case study workload will attempt to replicate a simple corporate Windows domain consisting of an Active Directory server, an MSSQL Server, an Exchange Server, and a Sharepoint Portal Server. This workload will use all closed source tools. It is included because it represents the set of servers that are run in many businesses. Further, Citrix and Sun have adopted the Xen hypervisor in their virtualization products which are both aimed at datacenter usage to consolidate servers, many of which are running Windows server operating systems. The proposed virtual machine architecture of the workload is depicted in Figure 4.

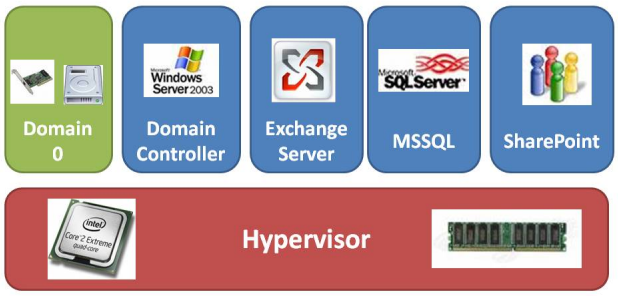

Figure 4: Windows Domain Case Study Workload Architecture

Conducting a Performance Study

Each of the case study workloads will be run with each of the three scheduling algorithms. Response time and throughput will be the principal metrics used to measure performance. The case study workloads will be run with varying numbers of simulated users. After all the varying user loads are tested, a VM with 100% CPU utilization will be run using cpuburn [10]. This will examine how each scheduler handles a VM that is over-utilizing CPU resources such as the case with a frozen VM which is using a large amount of the CPU time and not completing any meaningful work.

5 Hardware Requirements

In order to conduct this research a new machine will be purchased or built so that the newer features of Xen can be evaluated. The most significant specifications are CPU, RAM, and disk space. An AMD or Intel CPU that supports each manufacturer's respective virtualization technology is required. In addition the CPU will need to contain multiple cores so the ability of the scheduling algorithms to work across multiple cores can be evaluated. The system will also require at least 8 GB of RAM. If there is not enough physical memory in the system there will be side effects such as paging that will skew the results. Additionally the system will require at least 500 GB of storage with a fast serial SATA 2 bus. This re-

quirement will ensure that there is enough space to store all of the virtual machines that will be used and the SATA 2 bus will provide a theoretical maximum of 3 Gbps of throughput. The required system specifications are outlined in Table 1.

Table 1: Minimum specifications for the server that will run the Xen hypervisor.

Hardware	Minimum Requirement
CPU	2-4 core AMD or Intel CPU that supports virtualization extensions
Memory	8GB or more
Storage	500GB SATA 2 to allow for the storage of all the virtual machines

6 Analysis of Results

The results from conducting the performance study will be analyzed using statistical graphing tools like R. Once all of the results have been analyzed and represented graphically, conclusions will be drawn and trends found. This will be aided by tools like XenMon [5] that allow for reporting of performance statistics from running VMs. The proposed timetable can be seen in Table 2.

Table 2: A time line for the completion of the proposal.

Task	Timeline
Acquire parts for Xen machine and perform research (first semester).	2-4 weeks
Create a web application case study workload and adapt to a VPS case study work load (over winter break).	2-3 weeks
Create a domain case study workload (over winter break).	1-2 weeks
Perform experimentation (second semester).	3-6 weeks
Analyze results (second semester).	1-2 weeks
Finishing up the final paper (second semester).	2-3 weeks

7 Conclusion

An extensive study into the scheduling of virtual machines in the Xen hypervisor is currently an area that remains unexplored. The rising usage of virtualization provides a compelling case for the determination as to whether there exist certain situations where one scheduler

performs better than another. If so, it will likely be possible to customize the Xen hypervisor CPU scheduler for certain workload characteristic to increase performance.

References

[1] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[2] Ludmila Cherkasova, Diwaker Gupta, and Amin Vahdat. Comparison of the three CPU schedulers in Xen. *SIGMETRICS Perform. Eval. Rev.*, 35(2):42–51, 2007.

[3] Kenneth J. Duda and David R. Cheriton. Borrowed-virtual-time (bvt) scheduling: supporting latency-sensitive threads in a general-purpose scheduler. *Special Interest Group on Operating Systems Operating Systems Review*, 33(5):261–276, 1999.

[4] Gerhard Fohler. Scheduling of real-time systems fixed priority and earliest deadline first. Online at `http://www.artist-embedded.org/docs/Events/2005/Barcelona/ARTS2.pdf`, 2005.

[5] Diwaker Gupta, Rob Gardner, and Ludmila Cherkasova. XenMon: QoS monitoring and performance profiling tool. 2005. Online at `http://www.hpl.hp.com/techreports/2005/HPL-2005-187.html`.

[6] Ibrahim Hur and Calvin Lin. Memory scheduling for modern microprocessors. *ACM Trans. Comput. Syst.*, 25(4):10, 2007.

[7] I. Leslie, D. McAuley, R. Black, T. Roscoe, P. Barham, D. Evers, R. Fairbairns, and E. Hyden. The atropos scheduler. Online at `http://www.cl.cam.ac.uk/research/srg/netos/old-projects/pegasus/papers/jsac-jun97/node14.html`, 1999.

[8] David Mosberger and Tai Jin. httperf- a tool for measuring web server performance. 1998. Online at `http://www.hpl.hp.com/research/linux/httperf/docs.php`.

[9] Xen Project. Xen architecture overview. Online at `http://wiki.xensource.com/xenwiki/XenArchitecture?action=AttachFile&do=get&target=Xen+Architecture_Q1+2008.pdf`, 2008.

[10] Robert Redelmeier. Welcome to the cpuburn homepage. Online at `http://pages.sbcglobal.net/redelm/`, 2001.

[11] Seetharami R. Seelam and Patricia J. Teller. Virtual I/O scheduler: a scheduler of schedulers for performance virtualization. In *Proceedings of the 3rd International Conference on Virtual Execution Environments*, pages 105–115, New York, NY, USA, 2007. ACM.

[12] Chee Siang Wong, Ian Tan, Rosalind Deena Kumari, and Fun Wey. Towards achieving fairness in the Linux scheduler. *Special Interest Group on Operating Systems Operating Systems Review*, 42(5):34–43, 2008.

[13] Xianghua Xu, Peipei Shan, Jian Wan, and Yucheng Jiang. Performance evaluation of the CPU Scheduler in Xen. In *Information Science and Engieering, 2008.*, volume 2, pages 68–72, Dec. 2008.