ROUGH DRAFT—Needs revising: more technical language, less story telling, more diagrams, display results appropriately

An Investigation of Xen and PTLsim for Exploring Latency Constraints of Co-Processing Units

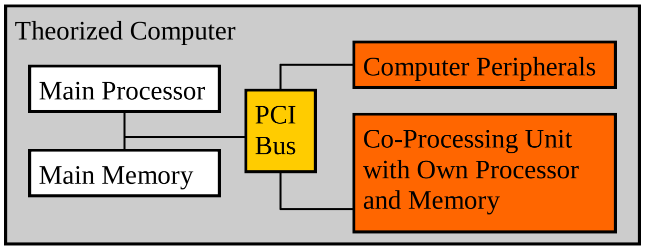
Grant Jenks - jenks@cs.ucla.edu
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095

ABSTRACT

What follows is a description that can hopefully act as a guide for researchers who are interested in using Xen with PTLsim to develop and analyze systems. This paper focuses on a specific set of systems that are intended to model a multiprocessing computer in which processing units, in addition to the main CPU and main memory, can assist in general computations. Particular depth is described in the methodology section in which a tremendous number of successes and failures are related to the reader in the hopes that they're research may be expedited through the process of setting up the initial environment. The environment used in this research involves an Intel processor with Virtualization Technology, Xen 3.0, Linux Kernel 2.6.20 and PTLsim rev 219. The results of three variations on systems are reported and some analysis is considered. The conclusion focuses on the future of Virtualization Technology and what contribution it may make in being able to accurately simulate hardware models of virtual machines.

INTRODUCTION

Most computers do not fully utilize all the PCI card slots that are available on the motherboard. This under-utilization provides an opportunity for improvement. The goal of this project will be to investigate the potential of a coprocessor which would reside on a PCI card. This coprocessor could handle some of the workload of the main CPU. Envisioned is a PCI card with appropriate features to offer the computer more parallel processing power and more memory. Some specific research has already been explored regarding the use of graphics cards as specialized processing units. The goal of this project is to understand what obstacles face a general processing unit.



The research will be carried out using cutting edge technology that has only recently become available in the consumer marketplace. Though machine virtualization has long been available for advanced servers and mainframe computers, recent advances in both hardware and software allow many options for the same virtualization on desktop computers. Used in this research will be a processor with Intel Virtualization Technology, a virtual machine platform from Xen, and a research platform from PTLsim. By utilizing the specialized hardware, Xen can run unmodified guests in virtual machine spaces. This virtual machine normally runs natively on the hardware in the desktop.

PTLsim allows the desktop machine hardware to be further virtualized so that cycle accurate measurements can be carried out. With the environment properly set up, the analysis is carried out using the PTLsim architecture for hardware simulation which has previously demonstrated accurate measurements.

The research is pursued in three steps. The first step involves demonstrating a need for offloading execution on a co-processing unit. Once this is fitfully demonstrated, the execution profiles of a variety of programs will be gathered when run natively on the unmodified machine. These profiles will be compared with subsequent profiles that are determined by running the programs on the simulated hardware. The comparison will yield results that can determine the overall usefulness of implementing the co-processing units that are modeled.

MOTIVATION

PCI card slots allow a plethora of peripherals to be added to machines without modification or upgrade of the motherboard. The number of PCI slots that are generally provided on a motherboard are often greater than the actual number of peripherals that are needed. The remaining peripheral spots may be used for various redundant or specialized applications. No known application however exists for general co-processing. Some designs currently devote these peripheral slots to graphics, networking, or even real-time physics processing but these applications are specific and the boards are tailored to their purpose. The real desire in this research is to have boards that can serve to offload the processing done by the main processor in any setting. A specific example of this involves offloading the cycles that run for programs in the systray of Windows clients so that the start up of programs may conceivably be accelerated with a processor that is less burdened.

PRIOR WORK

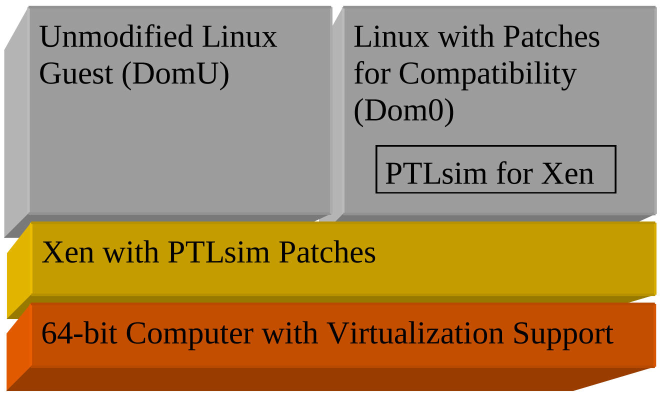
Much prior work has focused on using the peripherals attached to the motherboard for specialized processing techniques. A lot of papers have focused on ways to use the graphics cards in computers for computation intensive problems including many problems in linear algebra. A current manufacturer named Ageia has also produced a specialized card which is designed for physics processing of objects in games. These technologies are closely related to the exploration considered in this research. Many of the papers have been quite successful in demonstrating not only the usefulness of applying specialized hardware to various problems but also the efficiency. The purpose of this paper is different from other papers in this way. It is not intended for any specialized hardware to be included in the co-processing unit. While graphics cards have an enhanced capacity to do floating point calculations and other boards may specifically be optimized for power, the approach here is fundamentally different. To a certain degree, the idea presented here involves putting a computer within a computer. This statement holds true since the co-processing board will, just as any computer, be generally able to do any kind of processing. The drawback to this design is the same as that for any computer in which the ability to do calculations for a wide range of problems involves sacrificing the efficiency of solving specific, well understood problems. The reason for this design is, as stated previously, so that any program could be run on the co-processing system.

APPROACH

As the research is broken into three parts, each part has its own approach. The first part involves demonstrating some need for additional processing resources within a computer. This problem is solved using Intel’s Vtune program to gather runtime profiles of the computer under different loads. The computer which is used for this purpose in the research has a processor with Intel Hyperthreading technology. Therefore, runtime profiles with Intel Hyperthreading on and off are gathered and the results are described in a later section.

The second part of the research determines specific runtime profiles of programs run within the PTLsim simulation which can later be compared with profiles gathered from full simulation. This is carried out using PTLsim classic which has the limitation of only being able to monitor 32-bit programs in a cycle-accurate setting. This limitation will affect later work in which the programs run in full simulation must also be 32-bit as they are identical programs. This limitation is tolerable as many desktop machines are still running 32-bit operating systems.

The final and by far most complicated part of the research involves setting up PTLsim with Xen so that the hardware may be simulated in a cycle-accurate environment. This simulation will allow runtime profiles to be developed which can be compared to the previous runtime profiles gathered in PTLsim classic. This approach requires four key things. The first requirement is relatively new hardware. PTLsim requires a 64-bit processor and Xen requires a processor with virtualization technology support in order to run guest virtual machines unmodified. On top of the hardware, a modified version of Xen must be run which has the appropriate patches for PTLsim to interface with it. In order to start virtual machines, a Xen compatible Linux kernel machine must run on top of the Xen platform as domain 0. Once this is set up, PTLsim must be built appropriately so that advantage can be taken of the underlying Xen technology. The methodology section will speak at length about the many different configurations that were tried before a working environment could be set up. The final setup is shown in the figure below.


Unmodified Linux Guest (DomU)
Linux with Patches for Compatibility (Dom0)
PTLsim for Xen
Xen with PTLsim Patches
64-bit Computer with Virtualization Support

METHODOLOGY

The process of preparing the environments for each of the three stages of research has ranged from simply to very challenging. Each phase is described here with particular attention paid to the third stage was has the greatest associated difficulty and most useful capacity. Recall, the initial phase of the research sought to demonstrate a need for more processing power in today’s computers.

The runtime profiles of the test machine used in this research were initially collected using Intel’s Vtune software. Setup was easy and consisted of simply installing the program which can be acquired from Intel. Learning to use the program is aided with a detailed walk through guide that is provided with the software. The details of what programs were run and what the observed loads were on the machine are displayed in the results section of this paper.

Having acquired the runtime profiles of the system under different loads, the second part of the research involved collecting specific profiles of programs when run on the PTLsim virtual systems. This process involves installing PTLsim in classic mode and learning it well enough to collect data that can later be compared to data collected from the PTLsim/Xen environment. The steps involved in this are not much more challenging than before except that the sources for PTLsim must be downloaded from the online site and compiled and installed. With the necessary tools already installed, this requires only typing a few commands in the shell. Learning PTLsim is also not challenging as an extensive manual is provided with it that details not only its design and implementation of the simulated hardware but also its usage. The results of the software profiles that were gathered are detailed later in the results section.

Implementing the third stage of the research was considerably challenging. These challenges are described here in detail. The first challenge is acquiring hardware that will be compatible with the simulation environment. PTLsim specifies that it only needs a 64-bit computer to meet its hardware requirements. Xen is more demanding and requires a processor with virtualization technology in order to run unmodified guests under full virtualization. This is most popular for running Windows clients on virtual machines. PTLsim does not work with fully virtualized guests and so this requirement is not necessary for it. Xen will generally warn during an installation if the computer lacks virtualization technologies. Two machines were tried for the Xen environment. First an AMD 64-bit processor without virtualization technology computer was tested without much success. Though Xen would install and a Dom0 guest could be run on top of it, the system was buggy and limited in that Xen could not load unmodified systems and so could not be extensively tested. The second machine used had an Intel 5300 series 64-bit processor with Virtualization Technology. This machine worked very well with the newest version of Xen but warned when older versions of Xen were installed that the CPU microcode was not recognizable. This problem may be attributed simply to the hardware being ahead of the software. By using online updates to patch this problem, the issue was overcome.

With the proper hardware acquired, the next problem is to install Xen, modify it appropriately with PTLsim patches, and then modify the Dom0 operating system so that it is compatible with the setup. This challenge is not easily solved and is very time consuming. The experiences that I have gone through in this part are described individually and only in retrospect are they seen for the many dead ends that they are. Whatever parallels you can see with your own thinking, take heed that my experiences are presented so that yours are not the same.

It is the recommendation of PTLsim to use an operating system from one of the major distributors that has a considerable plethora of features. The one that they presently use is OpenSuse 10.2. They use the kernel from this operating system in both Dom0 and DomU. In my attempts I have used OpenSuse 10.2 and 10.1, Fedora 6, Xen 3.0, CentOS 5, and Ubuntu 6. The operating system that finally worked for the environment was OpenSuse 10.2 and I would recommend it just as much as PTLsim does.

When I began, I installed OpenSuse 10.2 on the machine without virtualization technology. OpenSuse has a wonderful operating system installer called Yast that makes it very easy to set up the operating system with Xen installed. After several installations it became clear to me that a simple partition table is much more easily managed in this situations than a normal one. Dividing each main folder onto its own partition is a standard Linux method for securing data but in this environment I've generally found it makes things needlessly complicated.

A word about partition setups before continuing. The latest and greatest file system will not make this process easy. Use ext2 or ext3 and remember to be sure that support for that file system is in whatever kernel is loaded at boot. This must be true of both the Dom0 virtual machine that is run and all guest virtual machines. Configuring guest properly can also be tricky with file systems. Xen tries hard to prevent the user from doing something foolish but is not always successful. A number of times, I have accidentally tried to boot Dom0 from its disk in a virtual machine as DomU while Dom0 is running. This is an almost guaranteed way to corrupt data on the disk. An all too frustrating thing occurs when after installing an operating system and all needed patches, the disk is corrupted in a few minutes and the process has to be started again.

Installing the patched version of the Xen kernel and tools in OpenSuse required only following the instructions provided on the web site. A very challenging error occurred however when trying to get the PTLsim demo machine to run. The error was a kernel panic error in which the booting virtual machine could not mount the disks to load the virtual file system. This error is difficult since the abstraction of the disks should be handled by Xen and the needed files for Xen were certainly available on the machine. While struggling with this error for a while, I learned the location of two important folders. The first location is /etc/xen. This folder contains Xen configuration files, most notably xensys.config and virtual machine configuration scripts. The second folder is /var/xen/log. This folder contains Xen and Qemu log files which can aid considerably in determining problems. Before I could resolve the kernel panic error, I got better hardware and decided to implement the environment on that hardware instead.

The second machine used in the research was a server that had two processors with four cores each, two gigabit Ethernet ports, and a raid controller. In describing the process of getting the environment to work, we will see how each of these aspects complicated things. I must also admit that I had never worked with a server before and so have learned a number of system administration tasks. Now, a few notes about the server before continuing. Make sure the BIOS and other firmware in the machine is up to date. A couple of errors that I have seen involved invalid microcode being read by the kernel. I don’t exactly know how this affects an operating system but having everything up to date is especially important to Xen. Another mistake would be to implement a raid. It’s not that it’s not possible. Rather, it is best to become fluent in installing the environment before trying to speed everything up. It is noteworthy however to say that a Raid may be particularly useful in some environments since Xen can become overly burdened and even crash in very I/O intensive tasks. This will effect the results in your environment since Xen/PTLsim will not abstract properties of the disk that may be changed.

After updating and tuning the server as necessary, I tried installing OpenSuse. Installation failed repeatedly for me. While investigating this I determined the problem was similar to another person’s who had recognized the problem to be a segmentation fault that occurred when reading the disk meta-data that was written by the Raid controller. Unfortunately, no one determined a solution for several weeks and so I meanwhile decided to try other operating systems. The first operating system that I tried was OpenSuse 10.1. I do not suggest using the older operating system for virtualization. Kernel improvements have been extremely important and so recent changes must be included in the distribution that is used. OpenSuse 10.1 had the same problems as 10.2 and never got past the installation phase.

The second distribution I tried was Fedora Core 6. This distribution is fundamentally different in that it derives from the Red Hat code base. Installation of Fedora is relatively simple and facilitated by its Anaconda installer. Installing Fedora with Xen automatically installs all the Xen libraries which include modified glib libraries. PTLsim has made further modifications to these libraries for its purposes which is why the patched Xen kernel and tools must be installed. The Fedora installation worked fine and was capable of creating virtual machine guests. It’s always a good test to be able to create virtual machines on the unmodified platform before making changes and the virtual machines under hardware simulation. By testing this process first, later issues can be nailed down more quickly.

In addition to trying Fedora, I installed CentOS which also derives significantly from the Fedora code base. Installation for CentOs was the same as that for Fedora. As far as I can tell from installation, the distributions are very similar. After following the instructions in the PTLsim manual with the exception of a few name changes, the environment was set up for simulation. It's important to note that the kernel that is built and patched by the operating system installer and update manager was the only compatible kernel that worked with Xen. When trying to run the sample domain that is provided, the same error as before was generated. This error was the kernel panic error which results from the VFS failing at boot up. This error also often occurred only seconds before the machine would freeze and need to be forcibly rebooted. Since hard reboots were often required, I recommend using a file system with journaling so as to avoid damaging blocks in these systems.

Given the similarity between Fedora and CentOS, it is not entirely surprising that they both failed. I decided after trying these systems to implement Ubuntu which has gained considerable favor with many Linux users recently. The other benefit of Ubuntu is that its code base is different from that of Suse or Redhat. Working with both the desktop version and the server version of the operating system resulted in the same outcome. Ubuntu will install and configure Xen during installation of the operating system. This makes it, like the others, very easy to install Xen. Making the changes to Xen however, proved to be too challenging for me in my narrowing time window. The Ubuntu installation with Xen includes the Xen modified glib libraries. These libraries are not compatible with the package manager which can install gcc. The GNU C compiler is absolutely necessary for both building the revised Xen kernel and the PTLsim executable. Furthermore, installing both sets of glib libraries was not acceptable to Ubuntu and led me to give up and try OpenSuse again, this time with success.

When I came back around to OpenSuse, considerable work had been done on the bug that I had earlier discovered. The solution to the bug was to zero out all the disks on the drive which would effectively erase the meta-data that caused the segmentation fault. After several attempts at this, I discovered that disk meta-data is stored at the very end of a disk and so skipping to the end and then writing zeros is a very efficient way of preparing the disk for installation. With this change in effect, the installation of OpenSuse 10.2 succeeded exactly as it should.

Following the installation, the steps in the online manual were followed as closely as possible. The kernel that is provided with PTLsim was installed to the necessary directory but failed to boot. Constructing an ram disk for boot up and providing that as a parameter for the kernel in Grub, also did not help. Both of these components are used later however and so should be held on to. The kernel that is installed with Yast is not compatible with the changes that are made to Xen. The problems observed were that the Yast configuration tool for virtual machines constantly reported no memory available in the computer and virtual machines that were started would silently fail in a process that degraded the stability of the machine. The kernel that finally did work was custom built from the patched sources that are provided on the PTLsim web site. The configuration file for building the kernel was based off the previous configuration file that is created during the initial installation. Additionally, conservative options were marked when configuring modules that were not already accounted for in the previous kernel configuration. The main goal of the final environment is stability and so including nothing experimental and everything recommended worked as it should. An initial ram disk for this kernel also had to be created using the initrd command.

Booting the PTLsim patched Xen kernel with the modified Xen tools and custom built Linux kernel with ram disk worked successfully. The resulting machine had all the features of OpenSuse 10.2 including the ability to manage software repositories with Yast. The only failed module at boot up was the AppArmor module which is not included in the kernel that is provided on the PTLsim web site. This failure did not affect the system. The process of starting virtual machines requires a lot of command line work. I tried repeatedly to port virtual machines that I had developed in other environments to this test machine unsuccessfully. For some reason, both fully virtualized and paravirtualized machines would fail silently. It is important to note that at the time of this writing, the PTLsim patches do not support fully virtualized machines and so trying to do so will result in a machine that freezes during boot. This also implies that PTLsim cannot be used to simulate Windows environments which would require full virtualization. The only effective way that was found to create paravirtualized machines was to do it the basic way. This involves creating an image file in the Dom0 domain and then copying over the disk from a working bootable system. Oddly, the kernel that was used to boot Dom0 was not able to boot the paravirtualized machine. Instead, the kernel that was provided on the site along with the initial ram disk that was made for it were able to.

With the ability to boot a working environment and start virtual machines under simulation, the three hardware profiles could be tested.

[This part is not finished in the research.]

Having created the three necessary environments for the research the tests were carried out successfully and have produced the results described next.

RESULTS

Figures 1, 2, and 3 display the profiles of their respective systems.

ANALYSIS

[Still preliminary]

The data gained from the first part of the research shows that the processor is over burdened by the work that needs to be done. Furthermore, the results show that the problem is not necessarily with speed. Rather, the issue is that there are more processes ready to run than there are processors on which to run them. Even when Hyperthreading is turned on in the processor, the report states that the machine would benefit from more processors. This provides evidence in favor of the idea that computers would benefit from co-processing units.

The second set of data gathered is displayed in the graph as

[This part is not finished in the research.]

In the third part of the research, runtime profiles were gathered with the system under different loads.

[This part is not finished in the research.]

CONCLUSION

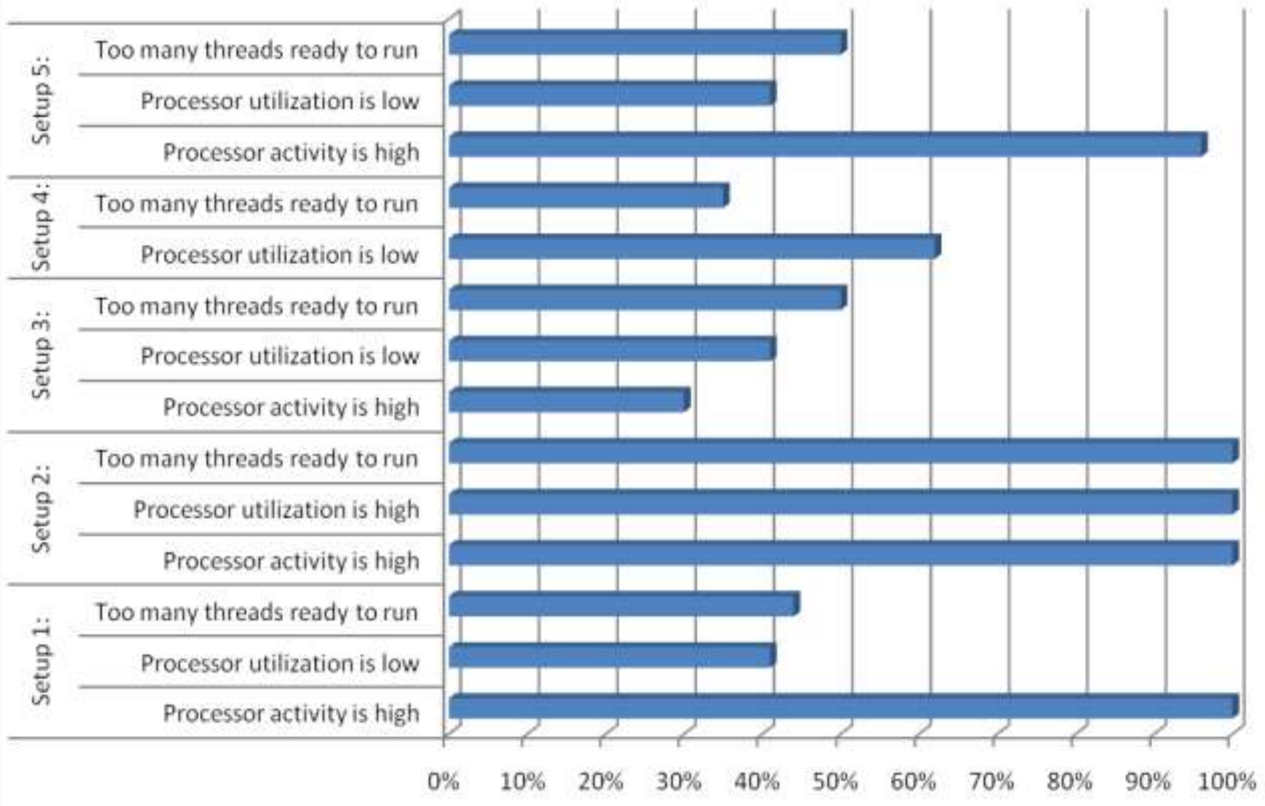
[Data conclusion not complete yet.]

The virtualization technologies used in this research are very promising in their capac-

Operating System	Successes	Failures	Notes
OpenSuse 10.2	Boots to working environment.	Raid does not work.	This is the best choice.
OpenSuse 10.1	Xen started properly.	Could not start virtual machines.	Use the newer version.
Fedora Core 6	Could start virtual machines.	Froze when changes were applied.	Good back up.
Ubuntu 6	Xen started properly.	Could not get necessary tools.	Good for learning Xen.
CentOS 5	Could start virtual machines.	Silently failed after changes.	Resilient to failure.

Results

Figure 1



Part 2 - Still pending.

Part 3 - Still pending.

ity to model hardware and gauge the effectiveness of changes. The environment in which this is done however, is overly complicated. Getting the three different kernels to work together with PTLsim is a challenging task. The recent advances in kernels is extremely important and differences between changes made even within a few weeks can make big differences in virtualization software now. Future techniques will probably use the newer kernel based virtual machine in which the Linux kernel developers are building in support. This should make things significantly easier for the user as less issues are bound to occur between fewer kernels. This technology, known as KVM, is actually what PTLsim will use by the end of the year. Hopefully, this will make hardware simulation with virtualization easier and more popular.

REFERENCES

- Yourst, Matt. "x86-64 Cycle Accurate Processor Simulation Design Infrastructure" 2006. 8 Oct. 1997 <http://www.ptlsim.org>.
- http://www.ageia.com/
- http://www.xensource.com/
- http://www.cl.cam.ac.uk/research/srg/netos/xen/

[Not exhaustive list. Cleanup required.]