Performance Evaluation of Intel EPT Hardware Assist

VMware ESX builds 140815 & 136362 (internal builds)

Introduction

For the majority of common workloads, performance in a virtualized environment is close to that in a native environment. Virtualization does create some overheads, however. These come from the virtualization of the CPU, the MMU (Memory Management Unit), and the I/O devices. In some of their recent x86 processors AMD and Intel have begun to provide hardware extensions to help bridge this performance gap. In 2006, both vendors introduced their first-generation hardware support for x86 virtualization with AMD-Virtualization™ (AMD-V™) and Intel® VT-x technologies. Recently Intel introduced its second generation of hardware support that incorporates MMU virtualization, called Extended Page Tables (EPT).

We evaluated EPT performance by comparing it to the performance of our software-only shadow page table technique on an EPT-enabled Intel system. From our studies we conclude that EPT-enabled systems can improve performance compared to using shadow paging for MMU virtualization. EPT provides performance gains of up to 48% for MMU-intensive benchmarks and up to 600% for MMU-intensive microbenchmarks. We have also observed that although EPT increases memory access latencies for a few workloads, this cost can be reduced by effectively using large pages in the guest and the hypervisor.

NOTE Many of the workloads presented in this paper are similar to those used in our recent paper about AMD RVI performance (*Performance Evaluation of AMD RVI Hardware Assist*). Because the papers used different ESX versions, however, the results are not directly comparable.

Background

Prior to the introduction of hardware support for virtualization, the VMware® virtual machine monitor (VMM) used software techniques for virtualizing x86 processors. We used binary translation (BT) for instruction set virtualization, shadow paging for MMU virtualization, and device emulation for device virtualization.

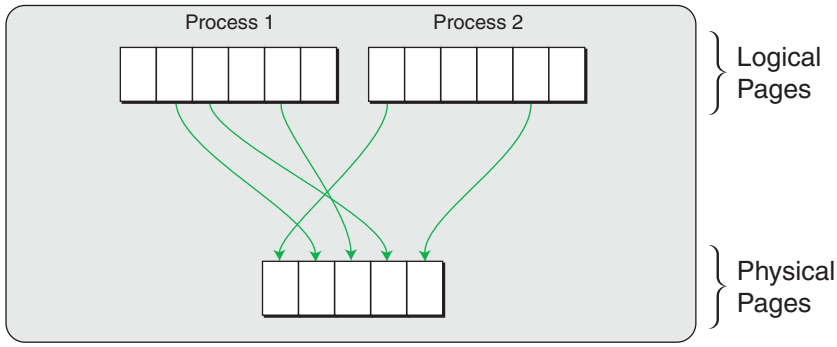
With the advent of Intel-VT in 2006 the VMM used VT for instruction-set virtualization on Intel processors that supported this feature. Due to the lack of hardware support for MMU virtualization in older CPUs, the VMM still used shadow paging for MMU virtualization. The shadow page tables stored information about the physical location of guest memory. Under shadow paging, in order to provide transparent MMU virtualization the VMM intercepted guest page table updates to keep the shadow page tables coherent with the guest page tables. This caused some overhead in the virtual execution of those applications for which the guest had to frequently update its page table structures.

With the introduction of EPT, the VMM can now rely on hardware to eliminate the need for shadow page tables. This removes much of the overhead otherwise incurred to keep the shadow page tables up-to-date. We describe these various paging methods in more detail in the next section and describe our experimental methodologies, benchmarks, and results in subsequent sections. Finally, we conclude by providing a summary of our performance experience with EPT.

MMU Architecture and Performance

In a native system the operating system maintains a mapping of logical page numbers (LPNs) to physical page numbers (PPNs) in page table structures (see Figure 1). When a logical address is accessed, the hardware walks these page tables to determine the corresponding physical address. For faster memory access the x86 hardware caches the most recently used LPN->PPN mappings in its translation lookaside buffer (TLB).

Figure 1. Native System Memory Management Unit Diagram

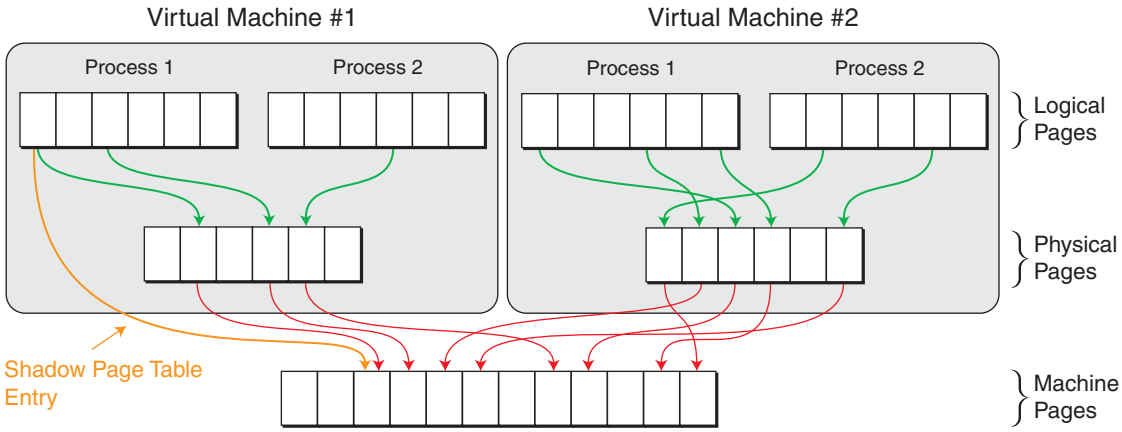


In a virtualized system the guest operating system maintains page tables just as the operating system in a native system does, but in addition the VMM maintains a mapping of PPNs to machine page numbers (MPNs), as described in the following two sections, "Software MMU" and "Hardware MMU."

Software MMU

In shadow paging the VMM maintains PPN->MPN mappings in its internal data structures and stores LPN->MPN mappings in shadow page tables that are exposed to the hardware (see Figure 2). The most recently used LPN->MPN translations are cached in the hardware TLB. The VMM keeps these shadow page tables synchronized to the guest page tables. This synchronization introduces virtualization overhead when the guest updates its page tables.

Figure 2. Shadow Page Tables Diagram

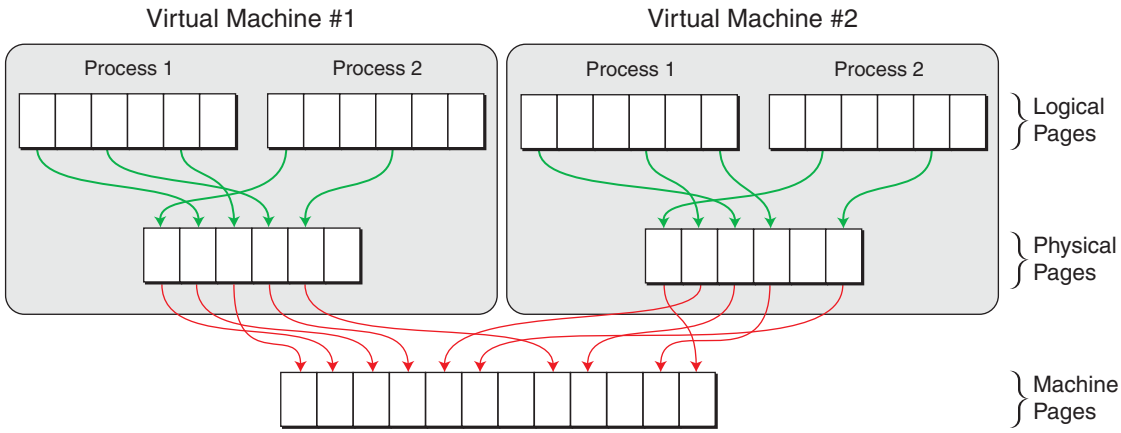


Hardware MMU

Using EPT, the guest operating system continues to maintain LPN->PPN mappings in the guest page tables, but the VMM maintains PPN->MPN mappings in an additional level of page tables, called nested page tables (see Figure 3). In this case both the guest page tables and the nested page tables are exposed to the hardware.

When a logical address is accessed, the hardware walks the guest page tables as in the case of native execution, but for every PPN accessed during the guest page table walk, the hardware also walks the nested page tables to determine the corresponding MPN. This composite translation eliminates the need to maintain shadow page tables and synchronize them with the guest page tables. However the extra operation also increases the cost of a page walk, thereby impacting the performance of applications that stress the TLB. This cost can be reduced by using large pages, thus reducing the stress on the TLB for applications with good spatial locality. For optimal performance the ESX VMM and VMkernel aggressively try to use large pages for their own memory when EPT is used.

Figure 3. Hardware Memory Management Unit Diagram



Experimental Methodology

This section describes the experimental configuration and the benchmarks used in this study.

Hardware

Configuration A

This configuration was used for all experiments except Order-Entry.

Intel OEM system (prerelease)

CPUs: Two Quad-Core Intel Xeon X5560 Processors (“Nehalem”)

BIOS: American Megatrends Inc. ver. 4.6.3.2

RAM: 36GB

Networking: Two Intel Corporation 82576 Gigabit network controllers

Storage Controller: Intel Corporation ICH10 6 port SATA AHCI controller

Disk: One 7200 RPM 500GB Seagate Barracuda SATA 3.0Gb/s Hard Drive

Configuration B

This configuration was used for the Order-Entry experiments.

Intel OEM system (prerelease)

CPUs: Two Quad-Core Intel Xeon X5570 Processors (“Nehalem”)

BIOS: American Megatrends Inc. version 4.6.3

RAM: 36GB

Networking: Intel Corporation 82571EB Gigabit Ethernet controllers (rev06)

Storage Controller: QLogic Corp. ISP2432-based 4Gb Fibre Channel to PCI Express HBA (rev 03)

External Storage:
4Gb/sec Fibre Channel switch
Two EMC CLARiiON CX3-80 arrays, each with 240 Seagate STT14685 CLAR146 146GB 15K RPM drives
One EMC CLARiiON CX3-40 array with 30 Hitachi HUS1511 CLAR146 146GB 15K RPM drives

Virtualization Software

All experiments except Order-Entry were performed with VMware ESX build 140815 (an internal build). The Order-Entry experiment was performed with VMware ESX build 136362 (an internal build). The tests in this study show performance differences between the VT VMM and the EPT VMM as a way of comparing shadow paging with EPT.

Benchmarks

In this study we used various benchmarks that to varying degrees stress MMU-related components in both software and hardware. These include:

- Kernel Microbenchmarks: A benchmark suite for system software performance analysis.
- Apache Compile: Compiling and building an Apache web server.
- Oracle Swingbench: An OLTP workload that uses Oracle Database Server on the backend.
- SPECjbb®2005: An industry standard server-side Java benchmark.
- Order-Entry benchmark: An OLTP benchmark with a large number of small transactions.
- SQL Server Database Hammer: A database workload that uses Microsoft SQL Server on the backend.

- Citrix XenApp: A workload that exports client sessions along with configured applications.

We ran these benchmarks in 32-bit and 64-bit virtual machines with a combination of Windows and Linux guest operating systems. Table 1 details the guest operating system used for each of these benchmarks.

Table 1. Guest Operating Systems Used for Benchmarks

Benchmark	Operating System
Kernel Microbenchmarks	32-bit Red Hat Enterprise Linux 5, Update 1 64-bit Red Hat Enterprise Linux 5, Update 1
Apache Compile	32-bit Red Hat Enterprise Linux 5, Update 3 64-bit Red Hat Enterprise Linux 5, Update 1
SPECjbb2005	32-bit Windows Server 2008 64-bit Windows Server 2008
Oracle Server Swingbench	64-bit Red Hat Enterprise Linux 5, Update 1
Order-Entry benchmark	64-bit Red Hat Enterprise Linux 5, Update 1
SQL Server Database Hammer	64-bit Windows Server 2008
Citrix XenApp	64-bit Windows Server 2003

It is important to understand the performance implications of EPT as we scale up the number of virtual processors in a virtual machine. We therefore ran some of the benchmarks in multiprocessor virtual machines.

Experiments

In this section, we describe the performance of EPT as compared with shadow paging for the workloads mentioned in the previous section.

MMU-Intensive Kernel Microbenchmarks

Kernel microbenchmarks comprise a suite of benchmarks that stress different subsystems of the operating system. These microbenchmarks are not representative of application performance; however they are useful for amplifying the performance impact of different subsystems so that they can be more easily studied.They can be broadly divided into system-call intensive benchmarks and MMU-intensive benchmarks. In these experiments we ran both 32-bit and 64-bit microbenchmarks. In our experiments we found that system-call intensive benchmarks performed equivalently with and without EPT enabled (for brevity these results are not included in this paper). However, with EPT enabled we observed gains of up to 600% on MMU-intensive microbenchmarks. Figure 4 and Figure 5 show the 32-bit and 64-bit results, respectively, for the following kernel microbenchmarks:

- segv: A C program that measures the processor fault-handling overhead in Linux by repeatedly generating and handling page faults.
- pcd: A C program that measures Linux process creation and destruction time under a pre-defined number of executing processes on the system.
- nps: A C program that measures the Linux process-switching overhead.
- fw: A C program that measures Linux process-creation overhead by forking processes and waiting for them to complete.

Figure 4. 32-bit MMU-Intensive Kernel Microbenchmark Results (Lower is Better)

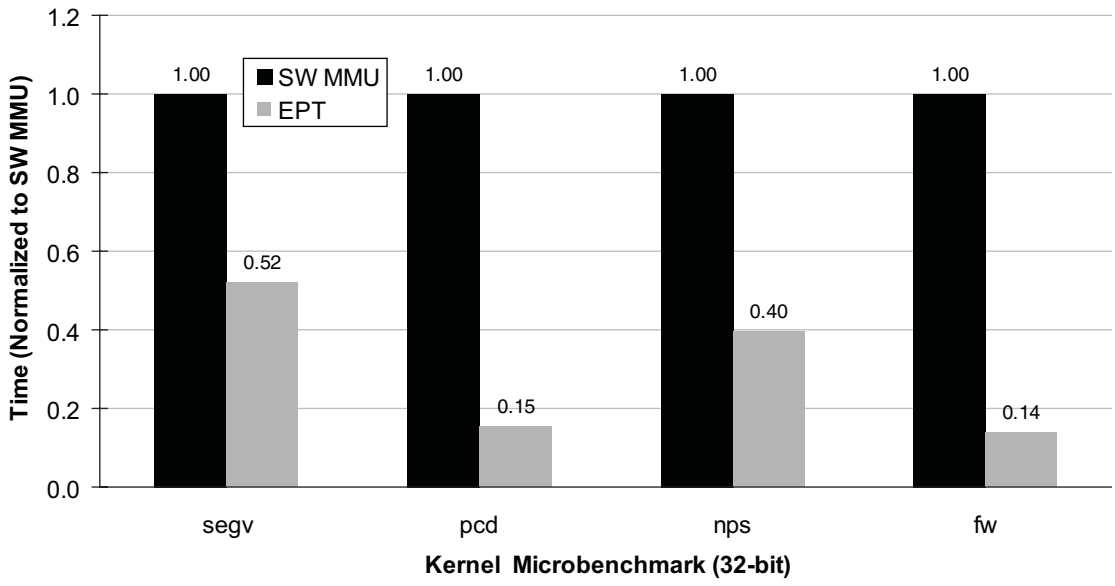
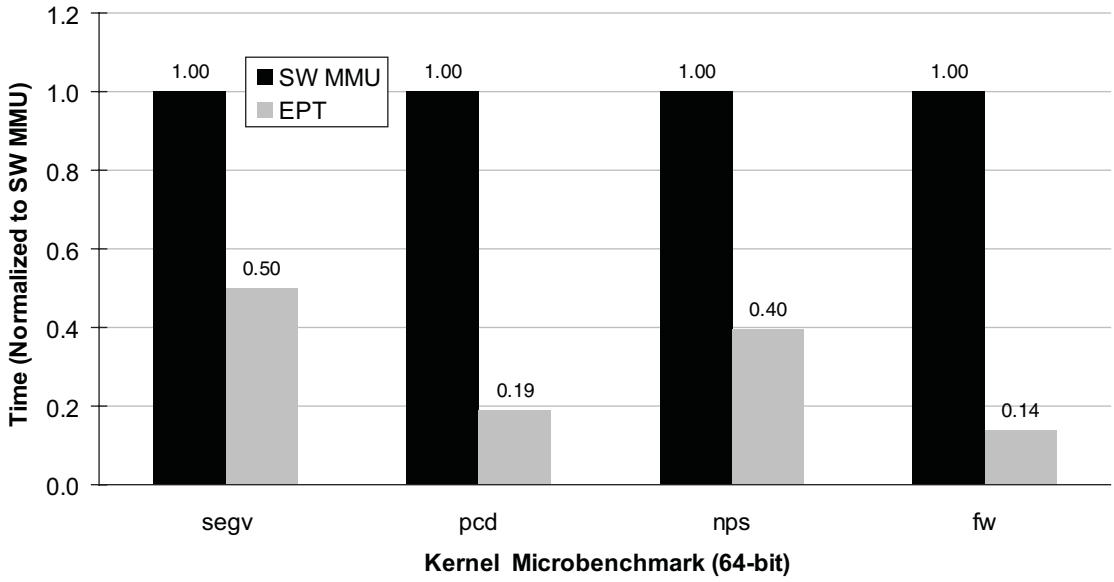


Figure 5. 64-bit MMU-Intensive Kernel Microbenchmark Results (Lower is Better)



Apache Compile

The Apache compile workload compiles and builds the Apache web server. This particular application is at an extreme of compilation workloads in that it is comprised of many small files. As a result many short-lived processes are created as each file is compiled. This behavior causes intensive MMU activity, similar to the MMU-intensive kernel microbenchmarks, and thus benefits greatly from EPT in both 32-bit and 64-bit guests, as shown in Figure 6 and Figure 7, respectively. The improvement provided by EPT increases with larger numbers of vCPUs; in the four vCPU case EPT performed 48% better than VT.

Figure 6. 32-bit Apache Compile Time (Lower is Better)

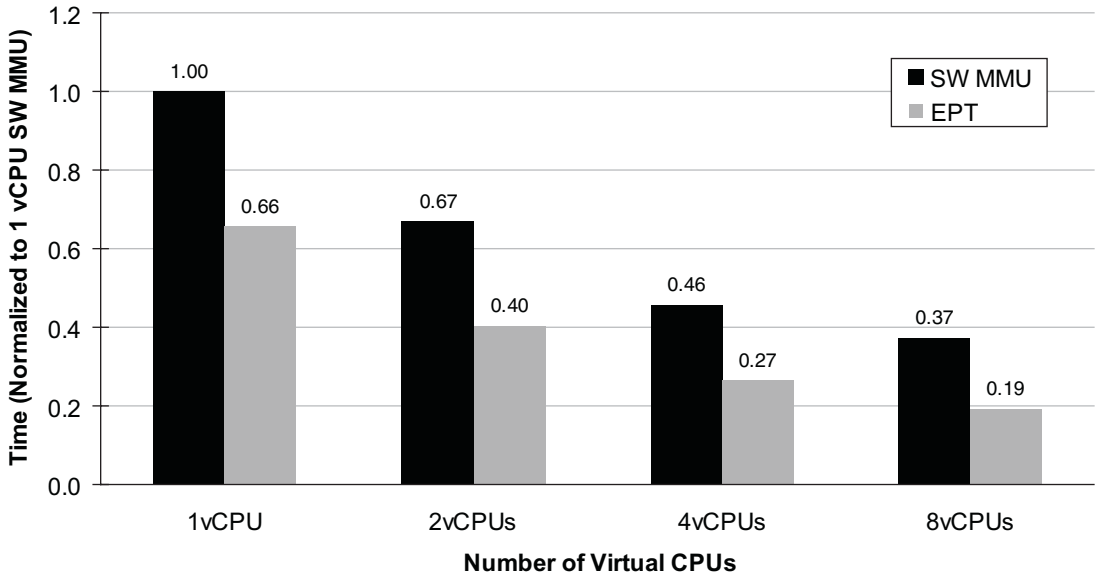
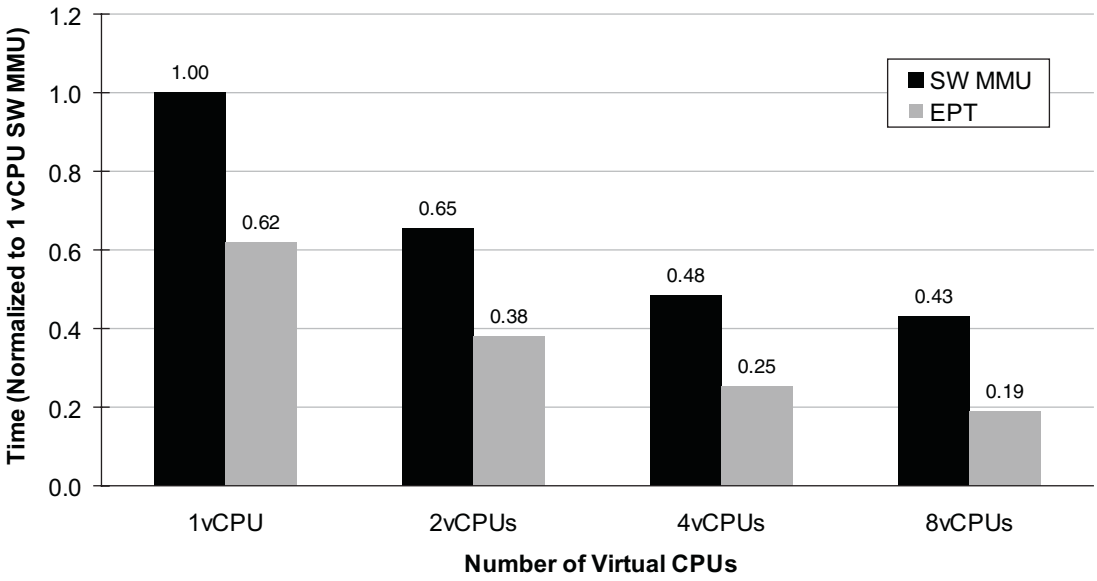


Figure 7. 64-bit Apache Compile Time (Lower is Better)



SPECjbb2005

SPECjbb2005 is an industry-standard server-side Java benchmark. It has little MMU activity but exhibits high TLB miss activity due to Java's usage of the heap and associated garbage collection. Modern x86 operating system vendors provide large page support to enhance the performance of such TLB-intensive workloads. Because EPT further increases the TLB miss latency (due to additional paging levels), large page usage in the guest operating system is imperative for high performance of such applications in an EPT-enabled virtual machine, as shown for 32-bit and 64-bit guests in Figure 8 and Figure 9, respectively.

Figure 8. 32-bit SPECjbb2005 Results (Higher is Better)

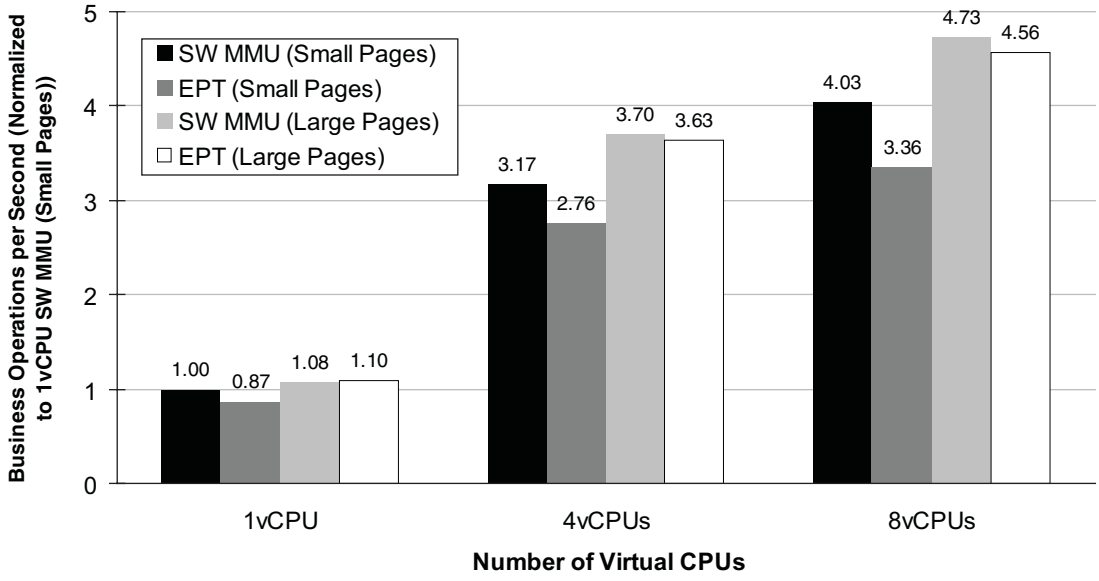
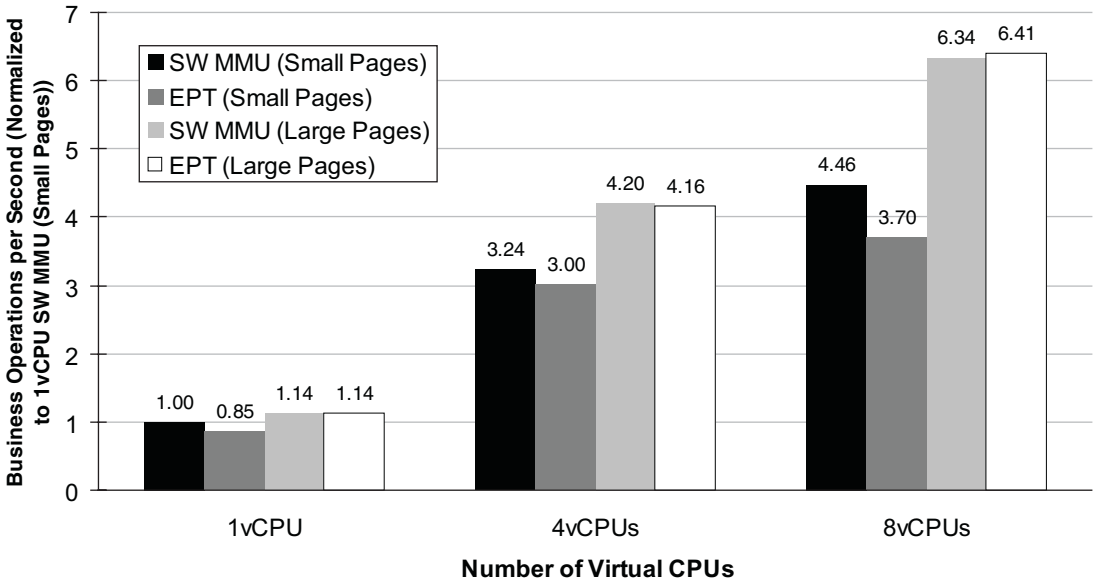


Figure 9. 64-bit SPECjbb2005 Results (Higher is Better)

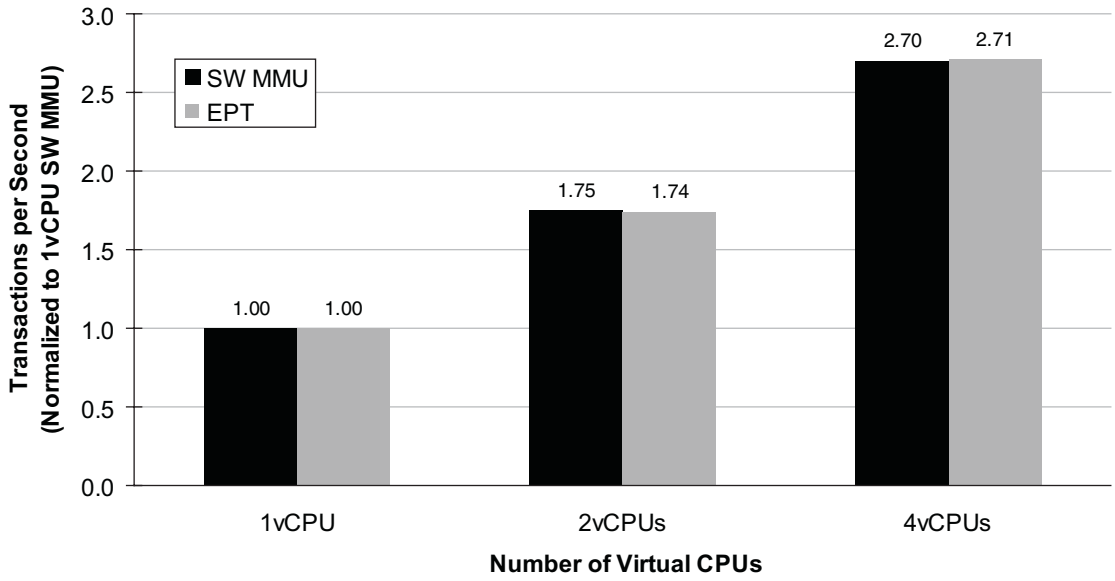


Oracle Server Swingbench

Swingbench is a database workload for evaluating Oracle database performance. We configured this experiment so that our database was fully cached in memory. As a result we used a small number of users connecting to the database and saturating the CPU. For this configuration Swingbench does not show significant MMU activity thereby gaining no performance due to EPT, as shown in Figure 10. We use large pages in the guest for all our runs.

Because the database is fully cached, a small number of users with no think time are sufficient to saturate the CPU. If the number of users is increased further, each user steps on the progress of another user and the benchmark scales negatively. In such a misconfiguration of this benchmark heavy context switching occurs between the user processes which can result in EPT showing significant performance boost as compared to VT.

Figure 10. Oracle Server Swingbench Results (Higher is Better)



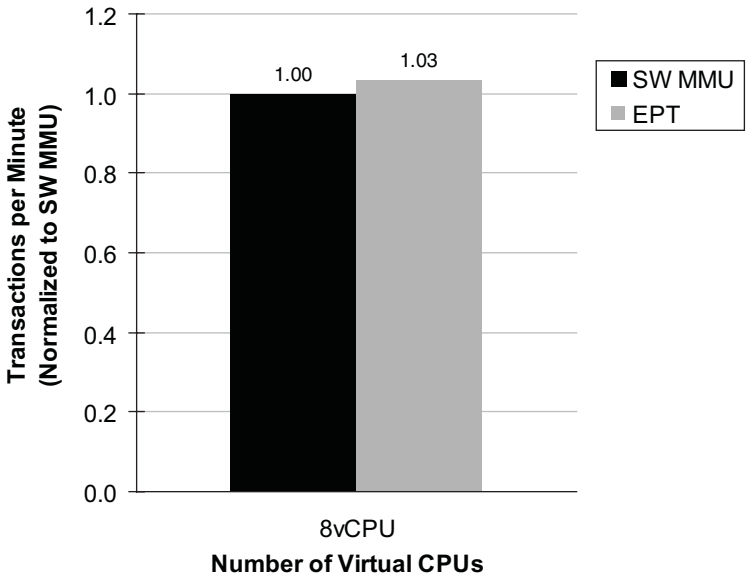
Order-Entry Benchmark

The Order-Entry benchmark is an OLTP benchmark with a large number of small transactions. Of the five transaction types, three update the database and two (of relatively low frequency) are read-only. The I/O load is very heavy and consists of small access sizes (2K-16K). The SAN accesses consist of random reads and writes with a 2:1 ratio in favor of reads. This experiment was performed using a trial version of Oracle 11g R1, and guest large pages were enabled.

NOTE The Order-Entry benchmark is a non-comparable implementation of the TPC-C business model. Our results are not TPC-C compliant, and not comparable to official TPC-C results. Deviations from the TPC-C specification: batch implementation; an undersized database for the observed throughput.

Figure 11 shows the results of the experiments.

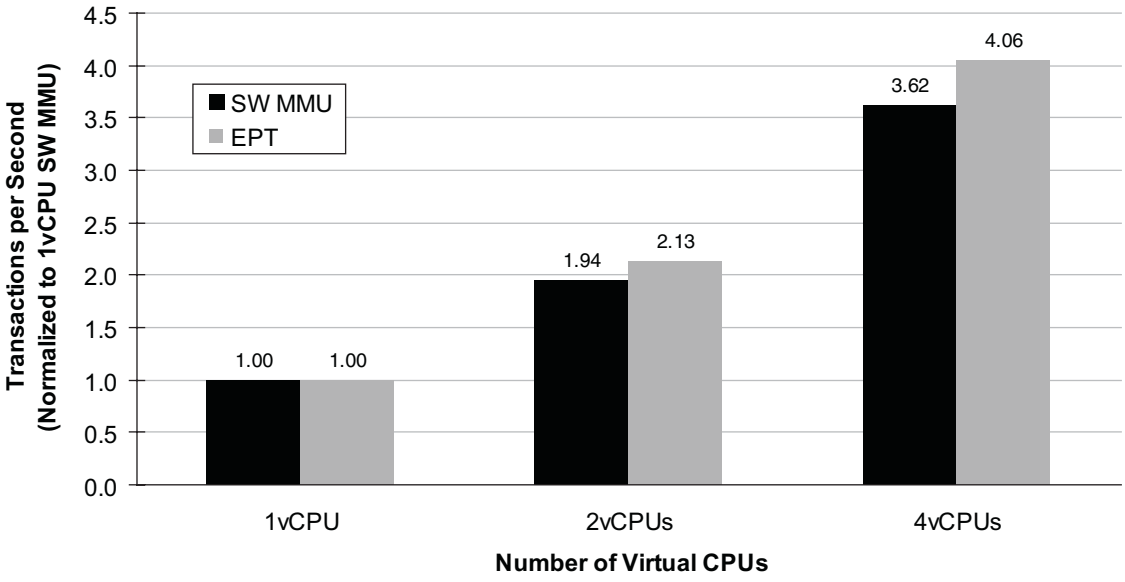
Figure 11. Order-Entry Benchmark Results (Higher is Better)



SQL Server Database Hammer

Database Hammer is a database workload for evaluating Microsoft SQL Server database performance. As shown in Figure 12, we observed that Database Hammer with lower vCPU counts is not MMU intensive, resulting in similar performance with and without EPT. However as we scale up the number of vCPUs we do see some MMU activity, thereby favoring EPT. We configured the guest to use large pages for all our Database Hammer runs.

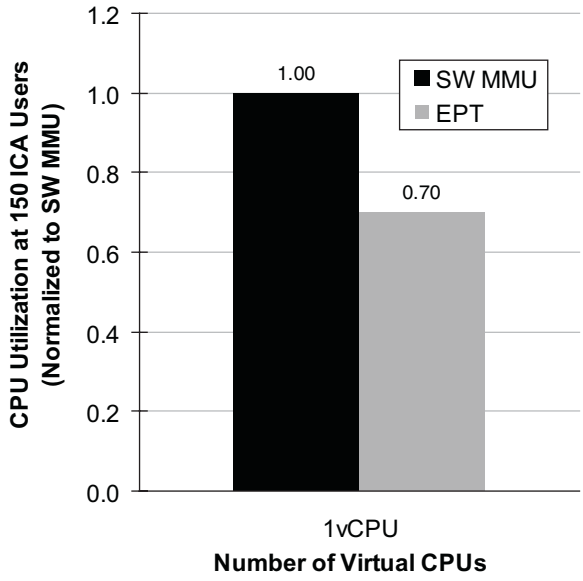
Figure 12. SQL Server Database Hammer Results (Higher is Better)



Citrix XenApp

Citrix XenApp is a presentation server or application session provider that enables its clients to connect and run their favorite personal desktop applications. To run Citrix, we used the Citrix Server Test Kit (CSTK) 2.1 workload generator for simulating users. Each user was configured as a normal user running the Microsoft Word workload from Microsoft Office 2000. This workload requires about 70MB of physical RAM per user. Due to heavy process creation and inter-process switching, Citrix is an MMU-intensive workload. As shown in Figure 13, we observed that EPT provided a boost of approximately 30%.

Figure 13. Citrix XenApp Results (Lower is Better)



Conclusion

Intel EPT-enabled CPUs offload a significant part of the VMM's MMU virtualization responsibilities to the hardware, resulting in higher performance. Results of experiments done on this platform indicate that the current VMware VMM leverages these features quite well, resulting in performance gains of up to 48% for MMU-intensive benchmarks and up to 600% for MMU-intensive microbenchmarks.

We recommend that TLB-intensive workloads make extensive use of large pages to mitigate the higher cost of a TLB miss.

About the Author

Nikhil Bhatia is a Performance Engineer at VMware. In this role, his primary focus is to evaluate and help improve the performance of the VMware Virtual Machine Monitor (VMM). Prior to VMware, Nikhil was a researcher at Oak Ridge National Laboratory (ORNL) in the Computer Science and Mathematics Division. Nikhil received a Master of Science in Computer Science from the University of Tennessee, where he specialized in tools for performance analysis of High Performance Computing (HPC) applications.

Acknowledgements

The author would like to thank the VMM developers for their tireless efforts in developing and optimizing the EPT VMM. The author would also like to thank Reza Taheri and Priti Mishra from the VMware Performance group for the Order-Entry Benchmark results. Finally he would like to thank members of the VMware VMM and Performance groups who have helped shape the document to its current state: Ole Agesen, Jennifer Anderson, Richard Brunner, Jim Mattson, and Aravind Pavuluri.

NOTE All information in this paper regarding future directions and intent are subject to change or withdrawal without notice and should not be relied on in making a purchasing decision concerning VMware products. The information in this paper is not a legal obligation for VMware to deliver any material, code, or functionality. The release and timing of VMware products remains at VMware's sole discretion.

If you have comments about this documentation, submit your feedback to: docfeedback@vmware.com

VMware, Inc. 3401 Hillview Ave., Palo Alto, CA 94304 www.vmware.com

Revision: 20090330; Item: EN-000194-00