EECS 252 Graduate Computer Architecture

Lec 17 – Advanced Memory Hierarchy 2

David Patterson
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~pattrsn
http://vlsi.cs.berkeley.edu/cs252-s06

Review [1]

- Memory wall inspires optimizations since so much performance lost there
  - Reducing hit time: Small and simple caches, Way prediction, Trace caches
  - Increasing cache bandwidth: Pipelined caches, Multibanked caches, Nonblocking caches
  - Reducing Miss Penalty: Critical word first, Merging write buffers
  - Reducing Miss Rate: Compiler optimizations
  - Reducing miss penalty or miss rate via parallelism: Hardware prefetching, Compiler prefetching
- "Auto-tuners" search replacing static compilation to explore optimization space?
- DRAM – Continuing Bandwidth innovations: Fast page mode, Synchronous, Double Data Rate

Review [2/2]

- VM Monitor presents a SW interface to guest software, isolates state of guests, and protects itself from guest software (including guest OSes)
- Virtual Machine Revival
  - Overcome security flaws of large OSes
  - Manage Software, Manage Hardware
  - Processor performance no longer highest priority

Outline

- Virtual Machines
- Xen VM: Design
- Administrivia
- Xen VM: Performance
- 80x86 VM Challenges
- AMD Opteron Memory Hierarchy
- Opteron Memory Performance vs. Pentium 4
- Fallacies and Pitfalls
- Discuss "Virtual Machines" paper
- Conclusion

Virtual Machine Monitors (VMMs)

- Virtual machine monitor (VMM) or hypervisor is software that supports VMs
- VMM determines how to map virtual resources to physical resources
- Physical resource may be time-shared, partitioned, or emulated in software
- VMM is much smaller than a traditional OS;
  - isolation portion of a VMM is ≈ 10,000 lines of code

VMM Overhead?

- Depends on the workload
- User-level processor-bound programs (e.g., SPEC) have zero-virtualization overhead
  - Runs at native speeds since OS rarely invoked
- I/O-intensive workloads ⇒ OS-intensive ⇒ execute many system calls and privileged instructions ⇒ can result in high virtualization overhead
  - For System VMs, goal of architecture and VMM is to run almost all instructions directly on native hardware
- If I/O-intensive workload is also I/O-bound ⇒ low processor utilization since waiting for I/O ⇒ processor virtualization can be hidden ⇒ low virtualization overhead

Requirements of a Virtual Machine Monitor

- A VM Monitor
  - Presents a SW interface to guest software,
  - Isolates state of guests from each other, and
  - Protects itself from guest software (including guest OSes)
- Guest software should behave on a VM exactly as if running on the native HW
  - Except for performance-related behavior or limitations of fixed resources shared by multiple VMs
- Guest software should not be able to change allocation of real system resources directly
- Hence, VMM must control ≈ everything even though guest VM and OS currently running is temporarily using them
  - Access to privileged state, Address translation, I/O, Exceptions and Interrupts, …

Requirements of a Virtual Machine Monitor

- VMM must be at higher privilege level than guest VM, which generally run in user mode
  - ⇒ Execution of privileged instructions handled by VMM
- E.g., Timer interrupt: VMM suspends currently running guest VM, saves its state, handles interrupt, determine which guest VM to run next, and then load its state
  - Guest VMs that rely on timer interrupt provided with virtual timer and an emulated timer interrupt by VMM
- Requirements of system virtual machines are ≈ same as paged-virtual memory:

1. At least 2 processor modes, system and user
2. Privileged subset of instructions available only in system mode, trap if executed in user mode
   - All system resources controllable only via these instructions

ISA Support for Virtual Machines

- If plan for VM during design of ISA, easy to reduce instructions executed by VMM, speed to emulate
  - ISA is virtualizable if can execute VM directly on real machine while letting VMM retain ultimate control of CPU: "direct execution"
  - Since VMs have been considered for desktop/PC server apps only recently, most ISAs were created ignoring virtualization, including 80x86 and most RISC architectures
- VMM must ensure that guest system only interacts with virtual resources ⇒ conventional guest OS runs as user mode program on top of VMM
  - If guest OS accesses or modifies information related to HW resources via a privileged instruction—e.g., reading or writing the page table pointer—it will trap to VMM
- If not, VMM must intercept instruction and support a virtual version of sensitive information as guest OS expects

Impact of VMs on Virtual Memory

- Virtualization of virtual memory if each guest OS in every VM manages its own set of page tables?
- VMM separates real and physical memory
  - Makes real memory a separate, intermediate level between virtual memory and physical memory
  - Some use the terms virtual memory, physical memory, and machine memory to name the 3 levels
  - Guest OS maps virtual memory to real memory via its page tables, and VMM page tables map real memory to physical memory
- VMM maintains a shadow page table that maps directly from the guest virtual address space to the physical address space of HW
  - Rather than pay extra level of indirection on every memory access
  - VMM must trap any attempt by guest OS to change its page table or to access the page table pointer

ISA Support for VMs & Virtual Memory

- IBM 370 architecture added additional level of indirection that is managed by the VMM
  - Guest OS keeps its page tables as before, so the shadow pages are unnecessary
  - (AMD Pacifica proposes same improvement for 80x86)
- To virtualize software TLB, VMM manages the real TLB and has a copy of the contents of the TLB of each guest VM
  - Any instruction that accesses the TLB must trap
  - TLBs with Process ID tags support a mix of entries from different VMs and the VMM, thereby avoiding flushing of the TLB on a VM switch

Impact of I/O on Virtual Memory

- I/O most difficult part of virtualization
  - Increasing number of I/O devices attached to the computer
  - Increasing diversity of I/O device types
  - Sharing of a real device among multiple VMs
  - Supporting many device drivers that are required, especially if different guest OSes are supported on same VM system
- Give each VM generic versions of each type of I/O device driver, and let VMM to handle real I/O
- Method for mapping virtual to physical I/O device depends on the type of device:
  - Disks partitioned by VMM to create virtual disks for guest VMs
  - Network interfaces shared between VMs in short time slices, and VMM tracks messages for virtual network addresses to ensure that guest VMs only receive their messages

Example: Xen VM

- Xen: Open-source System VMM for 80x86 ISA
  - Project started at University of Cambridge, GNU license model
- Original vision of VM is running unmodified OS
  - Significant wasted effort just to keep guest OS happy
- "paravirtualization" - small modifications to guest OS to simplify virtualization

3 Examples of paravirtualization in Xen:

1. To avoid flushing TLB when invoke VMM, Xen mapped into upper 64 MB of address space of each VM
2. Guest OS allowed to allocate pages, just check that didn't violate protection restrictions
3. To protect the guest OS from user programs in VM, Xen takes advantage of 4 protection levels available in 80x86
   - Most OSes for 80x86 keep everything at privilege levels 0 or at 3.
   - Xen VMM runs at the highest privilege level (0)
   - Guest OS runs at the next level (1)
   - Applications run at the lowest privilege level (3)

Xen changes for paravirtualization

- Port of Linux to Xen changed ≈ 3000 lines, or ≈ 1% of 80x86-specific code
  - Does not affect application-binary interfaces of guest OS
- OSes supported in Xen 2.0

OS	Runs as host OS	Runs as guest OS
Linux 2.4	Yes	Yes
Linux 2.6	Yes	Yes
NetBSD 2.0	No	Yes
NetBSD 3.0	Yes	Yes
Plan 9	No	Yes
FreeBSD 5	No	Yes

http://wiki.xensource.com/xenwiki/OSCompatibility

Xen and I/O

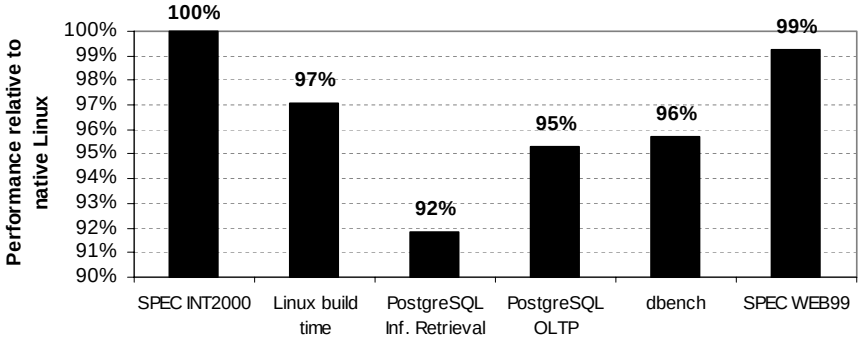
- To simplify I/O, privileged VMs assigned to each hardware I/O device: "driver domains"
  - Xen Jargon: "domains" = Virtual Machines
- Driver domains run physical device drivers, although interrupts still handled by VMM before being sent to appropriate driver domain
- Regular VMs ("guest domains") run simple virtual device drivers that communicate with physical devices drivers in driver domains over a channel to access physical I/O hardware
- Data sent between guest and driver domains by page remapping

CS252: Administrivia

- Advanced Memory Hierarchy (Ch 5) this week
- Monday 4/10: James Laudon on Sun T1/Niagara
  - Ph.D. Stanford '94 multiprocessor architect DASH / SGI Origin
- Wed 4/12 – Mon 4/17 Storage (Ch 6)
  - Handout Chapter 6
- Project Update Meeting Wednesday 4/19
- Monday 4/24 Quiz 2 5-8 PM (Mainly Ch 4 to 6)
- Wed 4/26 Bad Career Advice / Bad Talk Advice
- Project Presentations Monday 5/1 (all day)
- Project Posters 5/3 Wednesday (11-1 in Soda)
- Final Papers due Friday 5/5 (email Archana, who will post papers on class web site)

Xen Performance

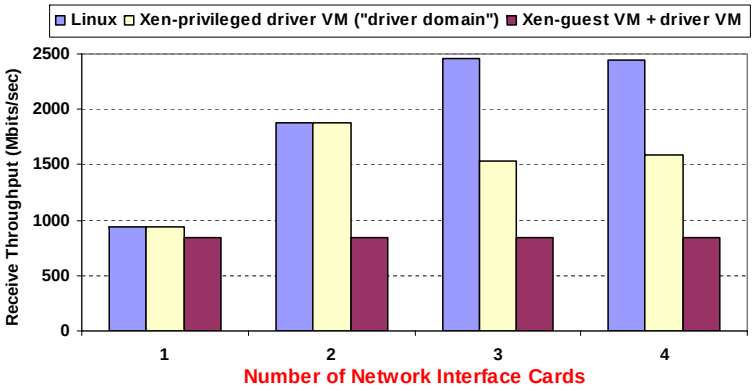
- Performance relative to native Linux for Xen for 6 benchmarks from Xen developers



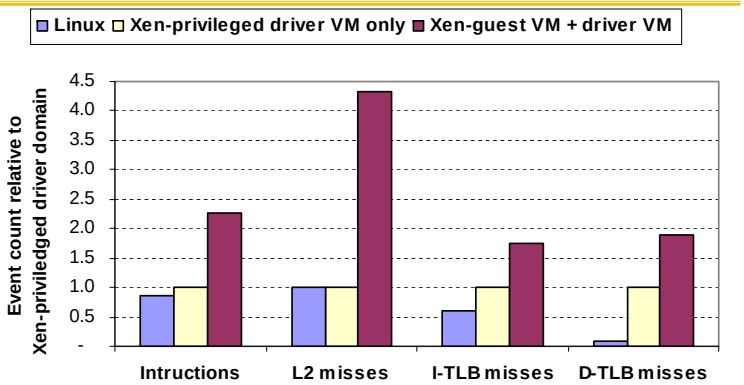
- Slide 6: User-level processor-bound programs? I/O-intensive workloads? I/O-Bound I/O-Intensive?

Xen Performance, Part II

- Subsequent study noticed Xen experiments based on 1 Ethernet network interfaces card (NIC), and single NIC was a performance bottleneck



Xen Performance, Part III



1. > 2X instructions for guest VM + driver VM
2. > 4X L2 cache misses
3. 12X – 24X Data TLB misses

Xen Performance, Part IV

1. > 2X instructions: page remapping and page transfer between driver and guest VMs and due to communication between the 2 VMs over a channel
2. 4X L2 cache misses: Linux uses zero-copy network interface that depends on ability of NIC to do DMA from different locations in memory
   - Since Xen does not support "gather DMA" in its virtual network interface, it can't do true zero-copy in the guest VM
3. 12X – 24X Data TLB misses: 2 Linux optimizations
   - Superpages for part of Linux kernel space, and 4MB pages lowers TLB misses versus using 1024 4 KB pages. Not in Xen
   - PTEs marked global are not flushed on a context switch, and Linux uses them for its kernel space. Not in Xen

- Future Xen may address 2. and 3., but 1. inherent?

Protection and Instruction Set Architecture

- Example Problem: 80x86 POPF instruction loads flag registers from top of stack in memory
  - One such flag is Interrupt Enable (IE)
  - In system mode, POPF changes IE
  - In user mode, POPF simply changes all flags *except* IE
  - Problem: guest OS runs in user mode inside a VM, so it expects to see changed a IE, but it won't
- Historically, IBM mainframe HW and VMM took 3 steps:

1. Reduce cost of processor virtualization
   - Intel/AMD proposed ISA changes to reduce this cost
2. Reduce interrupt overhead cost due to virtualization
3. Reduce interrupt cost by steering interrupts to proper VM directly without invoking VMM

- 2. and 3. not yet addressed by Intel/AMD; in the future?

80x86 VM Challenges

- 18 instructions cause problems for virtualization:

1. Read control registers in user model that reveal that the guest operating system in running in a virtual machine (such as POPF), and
2. Check protection as required by the segmented architecture but assume that the operating system is running at the highest privilege level

- Virtual memory: 80x86 TLBs do not support process ID tags ⇒ more expensive for VMM and guest OSes to share the TLB
  - each address space change typically requires a TLB flush

Intel/AMD address 80x86 VM Challenges

- Goal is direct execution of VMs on 80x86
- Intel's VT-x
  - A new execution mode for running VMs
  - An architected definition of the VM state
  - Instructions to swap VMs rapidly
  - Large set of parameters to select the circumstances where a VMM must be invoked
  - VT-x adds 11 new instructions to 80x86
- Xen 3.0 plan proposes to use VT-x to run Windows on Xen
- AMD's Pacifica makes similar proposals
  - Plus indirection level in page table like IBM VM 370
- Ironic adding a new mode
  - If OS start using mode in kernel, new mode would cause performance problems for VMM since ≈ 100 times too slow

Outline

- Virtual Machines
- Xen VM: Design
- Administrivia
- Xen VM: Performance
- 80x86 VM Challenges
- AMD Opteron Memory Hierarchy
- Opteron Memory Performance vs. Pentium 4
- Fallacies and Pitfalls
- Discuss "Virtual Machines" paper
- Conclusion

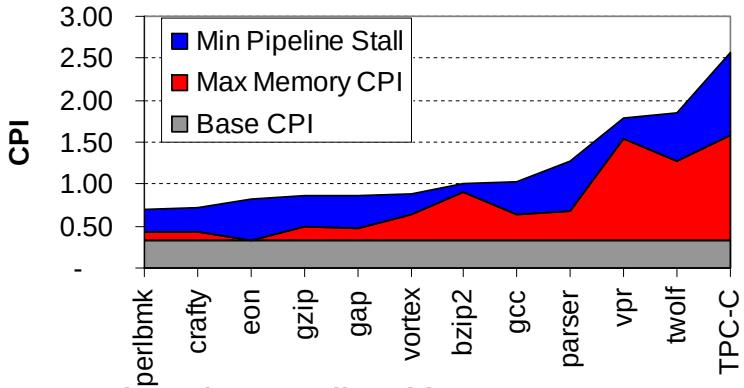
AMD Opteron Memory Hierarchy

- 12-stage integer pipeline yields a maximum clock rate of 2.8 GHz and fastest memory PC3200 DDR SDRAM
- 48-bit virtual and 40-bit physical addresses
- I and D cache: 64 KB, 2-way set associative, 64-B block, LRU
- L2 cache: 1 MB, 16-way, 64-B block, pseudo LRU
- Data and L2 caches use write back, write allocate
- L1 caches are virtually indexed and physically tagged
- L1 I TLB and L1 D TLB: fully associative, 40 entries
  - 32 entries for 4 KB pages and 8 for 2 MB or 4 MB pages
- L2 I TLB and L1 D TLB: 4-way, 512 entities of 4 KB pages
- Memory controller allows up to 10 cache misses
  - 8 from D cache and 2 from I cache

Opteron Memory Hierarchy Performance

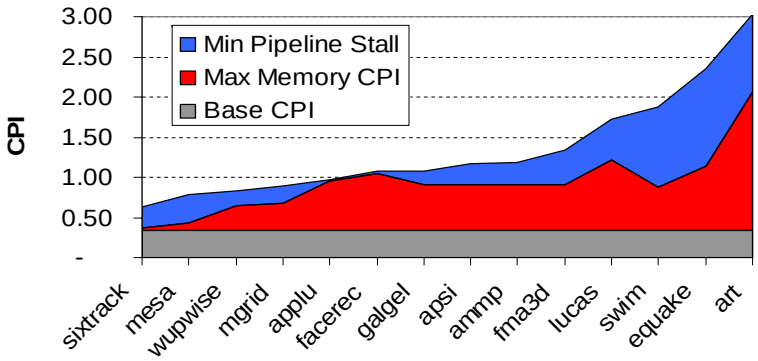
- For SPEC2000
  - I cache misses per instruction is 0.01% to 0.09%
  - D cache misses per instruction are 1.34% to 1.43%
  - L2 cache misses per instruction are 0.23% to 0.36%
- Commercial benchmark ("TPC-C-like")
  - I cache misses per instruction is 1.83% (100X!)
  - D cache misses per instruction are 1.39% (≈ same)
  - L2 cache misses per instruction are 0.62% (2X to 3X)
- How compare to ideal CPI of 0.33?

CPI breakdown for Integer Programs



- CPI above base attributable to memory ≈ 50%
- L2 cache misses ≈ 25% overall (50% memory CPI)
  - Assumes misses are *not* overlapped with the execution pipeline or with each other, so the pipeline stall portion is a lower bound

CPI breakdown for Floating Pt. Programs



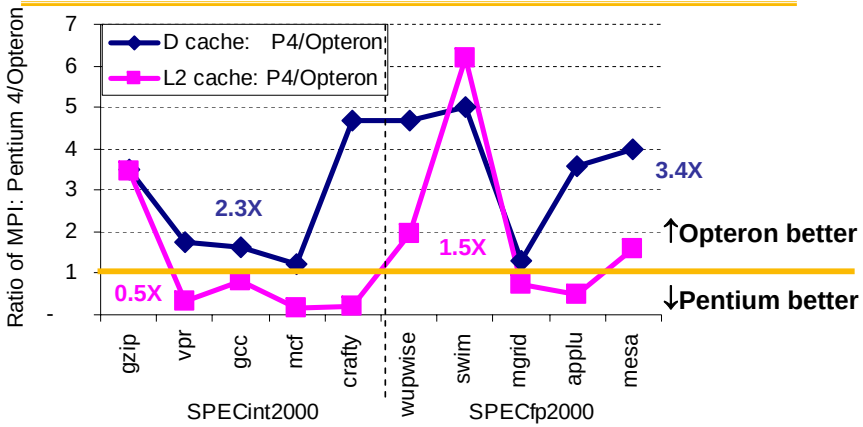
- CPI above base attributable to memory ≈ 60%
- L2 cache misses ≈ 40% overall (70% memory CPI)
  - Assumes misses are *not* overlapped with the execution pipeline or with each other, so the pipeline stall portion is a lower bound

Pentium 4 vs. Opteron Memory Hierarchy

CPU	Pentium 4 (3.2 GHz*)	Opteron (2.8 GHz*)
Instruction Cache	Trace Cache (8K micro-ops)	2-way associative, 64 KB, 64B block
Data Cache	8-way associative, 16 KB, 64B block, inclusive in L2	2-way associative, 64 KB, 64B block, exclusive to L2
L2 cache	8-way associative, 2 MB, 128B block	16-way associative, 1 MB, 64B block
Prefetch	8 streams to L2	1 stream to L2
Memory	200 MHz x 64 bits	200 MHz x 128 bits

***Clock rate for this comparison in 2005; faster versions existed**

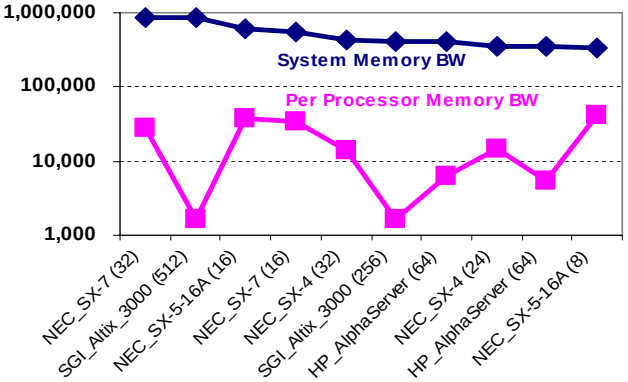
Misses Per Instruction: Pentium 4 vs. Opteron



- **D cache miss: P4 is 2.3X to 3.4X vs. Opteron**
- **L2 cache miss: P4 is 0.5X to 1.5X vs. Opteron**
- **Note: Same ISA, but not same instruction count**

Fallacies and Pitfalls

- **Not delivering high memory bandwidth in a cache-based system**
  - **10 Fastest computers at Stream benchmark [McCalpin 2005]**
  - **Only 4/10 computers rely on data caches, and their memory BW per processor is 7X to 25X slower than NEC SX7**



"Virtual Machine Monitors: Current Technology and Future Trends" [1/2]

- **Mendel Rosenblum and Tal Garfinkel, *IEEE Computer*, May 2005**
- **How old are VMs? Why did it lie fallow so long?**
- **Why do authors say this technology got hot again?**
  - **What was the tie in to massively parallel processors?**
- **Why would VM re-invigorate technology transfer from OS researchers?**
- **Why is paravirtualization popular with academics? What is drawback to commercial vendors?**
- **How does VMware handle privileged mode instructions that are not virtualizable?**
- **How does VMware ESX Server take memory pages away from Guest OS?**

"Virtual Machine Monitors: Current Technology and Future Trends" [2/2]

- How does VMware avoid having lots of redundant copies of same code and data?
- How did IBM mainframes virtualize I/O?
- How did VMware Workstation handle the many I/O devices of a PC? Why did they do it? What were drawbacks?
- What does VMware ESX Server do for I/O? Why does it work well?
- What important role do you think VMs will play in the future Information Technology? Why?
- What is implication of multiple processors/chip and VM?

And in Conclusion

- Virtual Machine Revival
  - Overcome security flaws of modern OSes
  - Processor performance no longer highest priority
  - Manage Software, Manage Hardware
- "… VMMs give OS developers another opportunity to develop functionality no longer practical in today's complex and ossified operating systems, where innovation moves at geologic pace ."

  [Rosenblum and Garfinkel, 2005]
- Virtualization challenges for processor, virtual memory, I/O
  - Paravirtualization, ISA upgrades to cope with those difficulties
- Xen as example VMM using paravirtualization
  - 2005 performance on non-I/O bound, I/O intensive apps: 80% of native Linux without driver VM, 34% with driver VM
- Opteron memory hierarchy still critical to performance