Hot Topics
(CSE 422S)

Ken Wong
Washington University
kenw@wustl.edu
www.arl.wustl.edu/~kenw

Topics

- Chip Multiprocessors
  » Increasing clock speed \(\rightarrow\) Increasing power/heat
  » Want higher speed but at about same power as 1 CPU
  » Multiple CPU/caches and memory attached to an interconnect

- Software Isolated Processes (MS Singularity)
  » Use SW verification instead of HW protection
    • Verify safe behavior (cannot construct or corrupt a memory ref.)
    • Type safe and memory safe operations

- OS Virtualization
  » Run multiple commodity OSes on the same hardware instance
    • e.g., XP and Linux on the same x86 processor
  » Want resource isolation and performance guarantees
  » OSes sit on top of a Virtual Machine Monitor

Multiple Processors on One Chip

- Why?
  » Transistor densities allow more processing units
  » Simpler circuits \(\rightarrow\) Lower power
  » Hope: Exploit parallelism to get higher speed

- Examples
  » Sun Niagara (8 processors)
  » IBM Power4 (2 processors)
  » Intel 28xx (17 processors)
  » Intel Pentium D and Core Duo (2 processors each)

- Hurdles
  » Programming
  » Memory bottleneck

Intel IXP 2800 Network Processor

- 16 MEs (Micro Engines)
  » 8 local thread contexts (hw)
    • Non-preemptive
  » Local I and D memory

- XScale control processor
  » Multiple memory interfaces
    • Local: 3 clocks
    • Scratch: 60 clocks
    • SRAM: 150 clocks
    • DRAM: 300 clocks

- Multiple memories
  » input MP ME ME ME ME output
  » PCI, GigE
  » SRAM
  » DRAM
  » no cache memory
### Application: Routing Packets

- **Table-driven architecture**
  - Fields of pkt header used in key search
  - Search TCAM to decide on action

### Thread Execution

- Non-preemptive threads
- Hardware-assisted context switching
- Signals indicate asynchronous operation done (e.g., memory)
- SRAM has:
  - atomic operations (e.g., add)
  - atomic enqueue, dequeue, get and put to/from 64-item queue array

### OS Virtualization

- **VMs (Guest OSes)**
  - User Software
    - Linux
    - BSD
    - Win XP

- **VMM**
  - virtual x86 CPU
  - virtual phy mem
  - virtual network
  - virtual blockdev

- **Virtual Machine Monitor (VMM)**
  - Provides the illusion of many virtual machines
  - Enables server consolidation, application mobility, new distributed (Internet) services

### Paravirtualization

- Def. Provide a VM abstraction that is similar but not identical to the underlying hardware
- Want
  - No modifications to application binaries
  - Support for full multi-application OSes
  - High performance and strong resource isolation
- Examples
  - Xen, Denali, VMware
Xen x86 Interface (1)

- **Memory Management**
  - Most difficult part of paravirtualization
  - x86 doesn't have software-managed TLB
    - TLB misses serviced by processor walking the page table
  - x86 TLB doesn't have identifier tags
    - Address space switches require complete TLB flush
  - Top 64 MB is reserved for Xen and is not accessible to guest OSes
  - All page table and segment table updates are validated by Xen

Xen x86 Interface (2)

- **CPU**
  - Xen runs in privilege ring 0 (highest)
    - Guest OS runs in privilege ring 1
    - Applications run in privilege ring 3
  - Privileged instructions (e.g., install new page table) are validated and executed by Xen instead of Guest OS
  - Exception handling (e.g., memory faults, system traps)
    - Registered with Xen by each Guest OS
    - System calls handled by fast handler which doesn't go thru Xen
  - Interrupts
    - Replaced by lightweight event system
  - Time
    - Each Guest OS has a timer interface (real and virtual time)

- **Device I/O**
  - Data transferred using asynchronous I/O rings

Current Approaches - Intel VT-x (1)

- **New modes of CPU operation**
  - VMX root for VMMs
  - VMX non-root for guest OSes

- **Virtual Machine Control Structure (VMCS)**
  - Contains guest state and host state
  - VM entry and VM exit triggered by certain instructions
    - VM entry
      - store host processor state and load guest processor state
    - VM exit reverses VM entry
      - many instructions can trigger VM exit from guest processor state
  - VM entry takes 2409 cycles on Intel P4 architecture
    - Expect 937 cycles on next generation Intel Core architecture

Current Approaches - Intel VT-x (2)

- **Extended Page Tables (EPT)**
  - Eliminate need for VMM maintaining shadow page tables
  - avoids VM entries and exits
  - EPT adds separate set of hardware-walked page tables
  - maps guest physical addresses (really virtual addresses) to host physical addresses

- **Tagged TLBs**
  - Assign virtual-processor identifier (VPI) to each virtual machine
    - tag TLB entries with VPID
    - avoids flushing TLB on every VM entry and VM exit
Intel VT-d (I/O extensions)

- DMA remapping
  » Software-specified protection domains for pages
  » restrict page access only to devices assigned to a domain
  » e.g., permit DMA only between a device and guest address space

- Interrupt Virtualization
  » IOTLB caches address translation lookups for I/O
  » DMA identifier is PCI Bus/device/function
  » Guest OS is notified after DMA completion
    » done without going through VMM
    » uses Message Signaled Interrupt (MSI) containing id that is used
      to index into interrupt-remapping table

References

- Barham, et. al., "Xen and the Art of Virtualization", Symp.
- Whitaker, Shaw and Gribble, "Denali: Lightweight Virtual
  Machines for Distributed and Networked Applications
- Fisher-Ogden, "Hardware Support for Efficient Virtualization"
- Robin and Irvine, "Analysis of the Intel Pentium’s Ability to
  Support a Secure Virtual Machine Monitor"
- Osisek, Jackson and Gum, "Intel Virtualization Technology:
  Hardware Support for Efficient Processor Virtualization"
- AMD, "AMD64 Virtualization Codenamed Pacifica
  Technology: Secure Virtual Machine Architecture
  Reference Manual"
- Asanovic, et. al., "The Landscape of Parallel Computing
  Research: A View from Berkeley" (Many core chips)