

Power Architecture™ '07  
DEVELOPER CONFERENCE



# Power Virtualization

24 Sep 2007

Hollis Blanchard, IBM

Power.ORG ™

# Topics



- » Processor Virtualization Overview
- » Server Power Virtualization
- » Embedded Power Virtualization

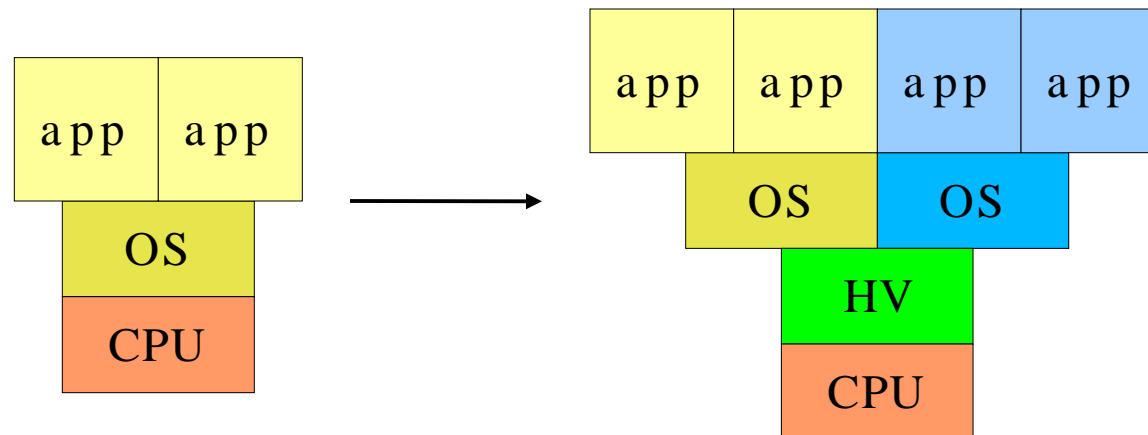
# Virtualization in General



# Processor Virtualization



- » Make each software stack think it owns the whole processor.
  - Rapidly switch between virtual machines without OS knowledge.
- » Isolation is a requirement.
- » The hypervisor is the software layer which multiplexes the VMs.



# The Good



- » Workload consolidation to increase server utilization.
- » Sandbox to run untrusted code.
- » Container to protect intrusion detection services.
- » Legacy software support.
- » Migrate running software stacks to other hardware.
- » Better visibility into low-level OS debugging.
- » Instant provisioning.
- » ...

# The Bad



- » There is a performance tradeoff.
  - Applications that used to own the whole processor must now share it.
  - Hypervisor adds a little runtime overhead too.
- » Increase in management complexity
  - Old scenario: two software stacks + two hardware systems
  - New scenario: two software stacks + one hardware system + one hypervisor
- » More abstraction, more software layers, more complexity...
  - More bugs
  - Increases size of Trusted Code Base
- » Increases impact of unpredicted hardware failure

# Server Virtualization



# Why Server Virtualization?



- » Workload consolidation
  - Increase server utilization
  - Reduce capital expenses
- » Legacy software
  - Continue to run legacy OS stacks
- » Migrate running software stacks to other systems
  - Workload balancing
  - Predicted hardware failure
- » Easier low-level debugging for developers
- » Instant provisioning
- » ...

# Hardware Support



- » New hypervisor privilege level
- » New real mode address translation
  - MMU inactive in real mode, so need another mechanism to enforce guest isolation
- » New hypervisor interrupts
  - hypercall
  - hypervisor decrementer
    - New hypervisor-privileged registers for interrupt handling
- » Most interrupts are delivered directly to guest
  - E.g. page fault, illegal instruction, etc
  - External interrupt is selectable
- » IO chipset support for DMA isolation
  - Allows direct IO access from guests

# Issues



- » **Requires** guest OS modification
  - Limited real-mode address space
  - Cannot emulate hypervisor-privileged instructions
    - All privileged instruction traps handled by guest OS
  - Guest cannot insert hash table mappings
- » Poor real-time characteristics
  - Guest controls hardware “interrupts enabled” bit
    - Doesn't affect hypervisor decrementer interrupt
  - Guest can block device interrupts for an entire timeslice
- » Guest debugging almost impossible
  - All debug trap instructions handled by guest OS
- » Inconsistent interrupt delivery models
  - Some hypervisor interrupts use SRR0/1, some use HSRR0/1

# Issues



- » Technologies precluded by direct-to-guest page faults:
  - Live migration
    - Copy guest pages to remote host, marking each read-only
    - When guest modifies page, add it back to the dirty list
    - Repeat until a small number of dirty pages remain
  - Memory compaction
    - Hypervisor coalesces identical guest pages to share a single page frame, using copy-on-write model
  - Memory overcommit
    - Hypervisor swaps guest memory to disk
- » Subsequent architectural hack makes these possible
  - POWER5+ and later

# Implementations



## » IBM: Advanced POWER Virtualization

- System p and BladeCenter JS21: POWER4 – POWER6, PowerPC 970

## » Sony: Cell OS LV1

- PlayStation 3: Cell BE

## » Toshiba: BEAT

- Celleb: Cell BE

## » Xen

- JS21: PowerPC 970
- <http://wiki.xensource.com/xenwiki/XenPPC>

# Embedded Virtualization



# Why Embedded Virtualization?



- » Reliable remote system software upgrades
- » RTOS + Linux functionality
  - Including GPL barrier
- » Failover without additional hardware
- » Contain untrusted code (e.g. console games)
- » Consolidate telecom services without sacrificing reliability
- » Legacy software
  - Exploit multicore processors with uniprocessor software stacks
- » Intrusion detection on network-attached devices
- » ...

# Hardware Support



» None.

- User and supervisor privilege levels
- MMU to enforce memory translation

# Issues



- » Must run guest in user level to enforce isolation
  - **All privileged instructions trap, not just the interesting ones**
    - Boring: mtspr SPRG0
    - Interesting: wrteei
  - **Removing all privileged instructions would be very invasive**
- » **Guests think they own the MMU**
  - **Removing MMU control would be very invasive**
- » **No chipset support for IO virtualization**
  - **Direct IO access to DMA-capable devices would break isolation guarantees**
  - **All IO is completely emulated**
    - **Drastic impact on guest interrupt handler latency**

# Direct IO



- » Guest can directly access physical IO without host involvement
  - Native speed
- » IOMMU provides isolation and physical address translation
  - Translation could be done with guest modifications
- » Issues:
  - IOMMU **required** for DMA isolation
  - Limited by number of physical IO devices
  - Guests must have device drivers
    - What about legacy guests on new hardware?
  - Breaks migration
  - IRQ delivery and routing

# Emulated IO



- » Host software emulates guest IO accesses
- » Issues:
  - Must write software to (perfectly?) emulate hardware
  - Dramatic increase in IO latency
  - Host OS must have physical device drivers
    - Device driver availability, licensing concerns

# Virtual IO



- » No hardware at all, just inter-guest data transfer
- » New guest device drivers co-operate with host
- » Issues:
  - Requires guest modification (at least new device drivers)
  - Host OS still needs physical IO drivers

# Embedded Virtualization: real-time



## » Real-time with dedicated cores

- Hypervisor isolates guests, but otherwise stays out of the way.
- More cores increases cost.

## » Real-time with a multiplexed core

- Hypervisor has its own CPU scheduler.
  - Must be RT-aware.
- Interrupt handler latency
  - Host introduces some interrupt handler latency
    - Maybe a lot of interrupt handler latency
  - What happens when the target guest isn't even running?
    - Context switch it in. How long does that take?

# Implementations



- » Green Hills Software: INTEGRITY Padded Cell
- » SYSGO: PikeOS
- » KVM (in development)
  - <http://kvm.qumranet.com/kvmwiki/CategoryPowerPC>

# Conclusion



# Conclusion



- » Server virtualization on Power Architecture is already thriving.
- » Embedded virtualization is heating up, and there are too many benefits to ignore!

**Power Architecture™**  
DEVELOPER CONFERENCE

**'07**

Power.ORG ™