Porting Xen Paravirtualization to MIPS Architecture

Yonghong Song

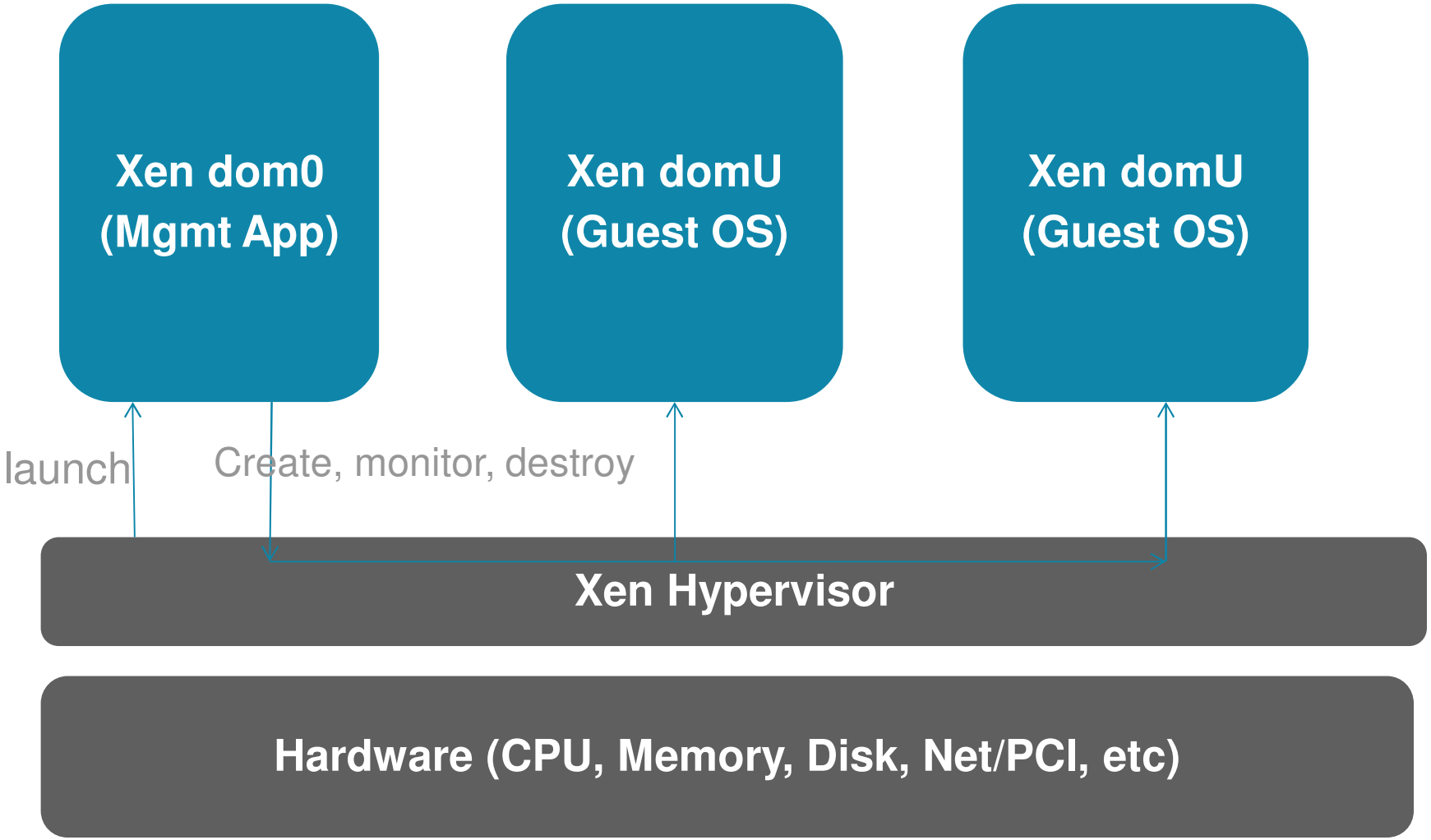
Broadcom

August 27-28, 2012
San Diego, CA, USA

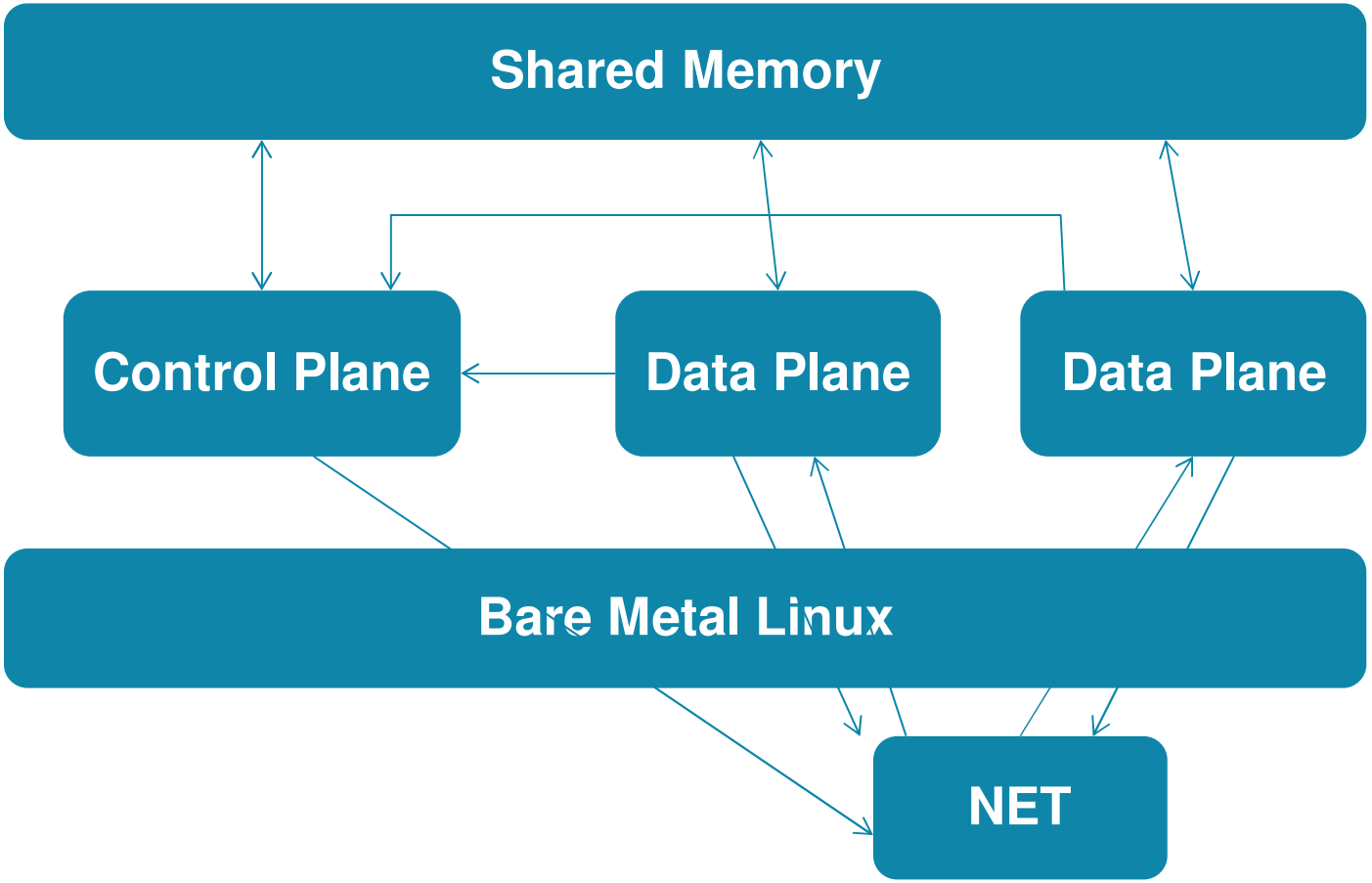
Motivation

- Broadcom XLP
  - 8 cores, 4 threads each core
  - Out-Of-Order
  - L1D, L1I, L2 each core, shared L3
  - Accelerators: NET, SEC, RAID, DMA, COMP, etc.
  - SOCs: USB, PCIE, FLASH, I2C, etc.
- Need for a software enabled virtualization solution
- Xen ported and provided as a solution

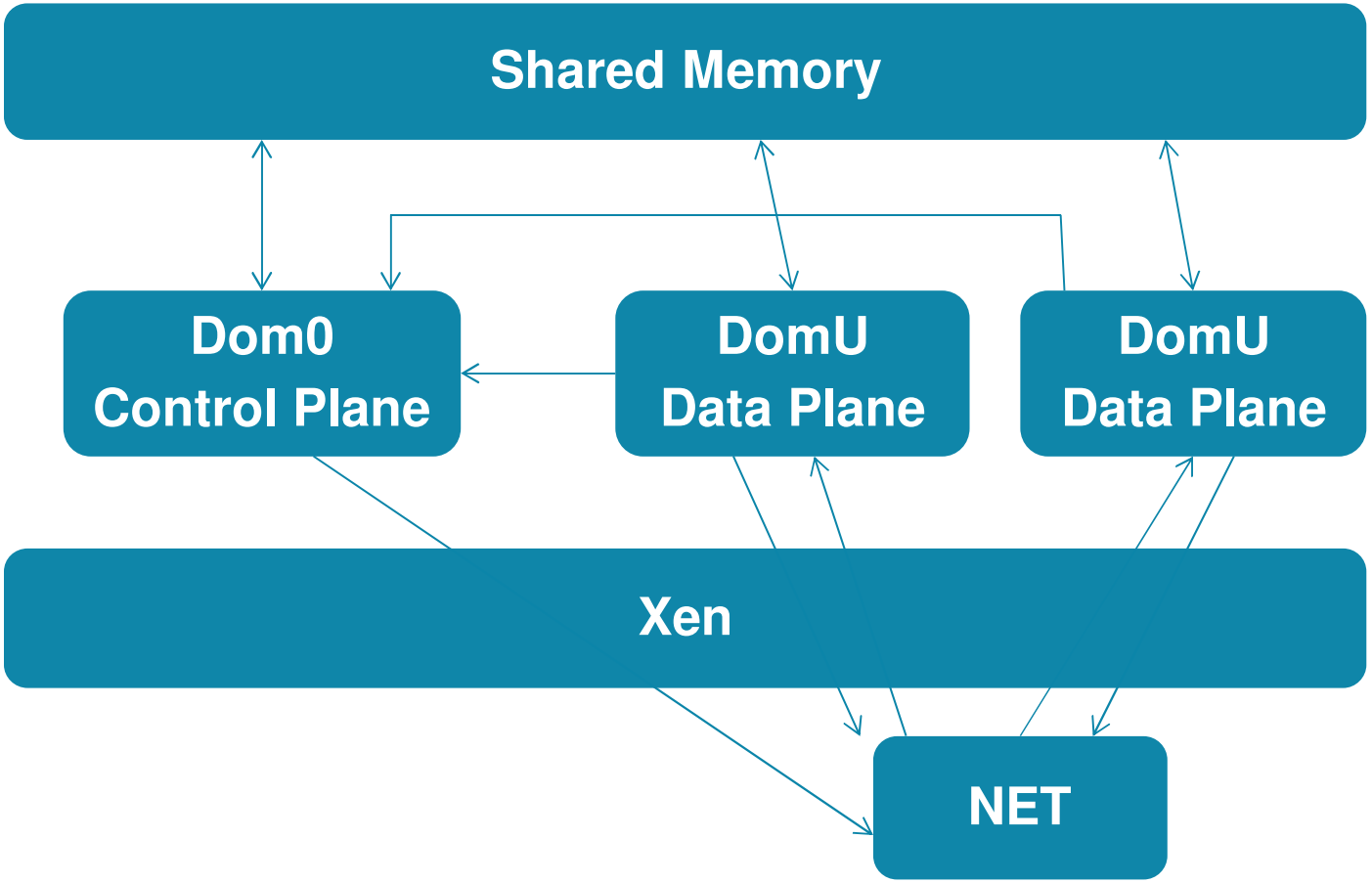
General Xen Usage Model



Hybrid Control/Data Plane Model



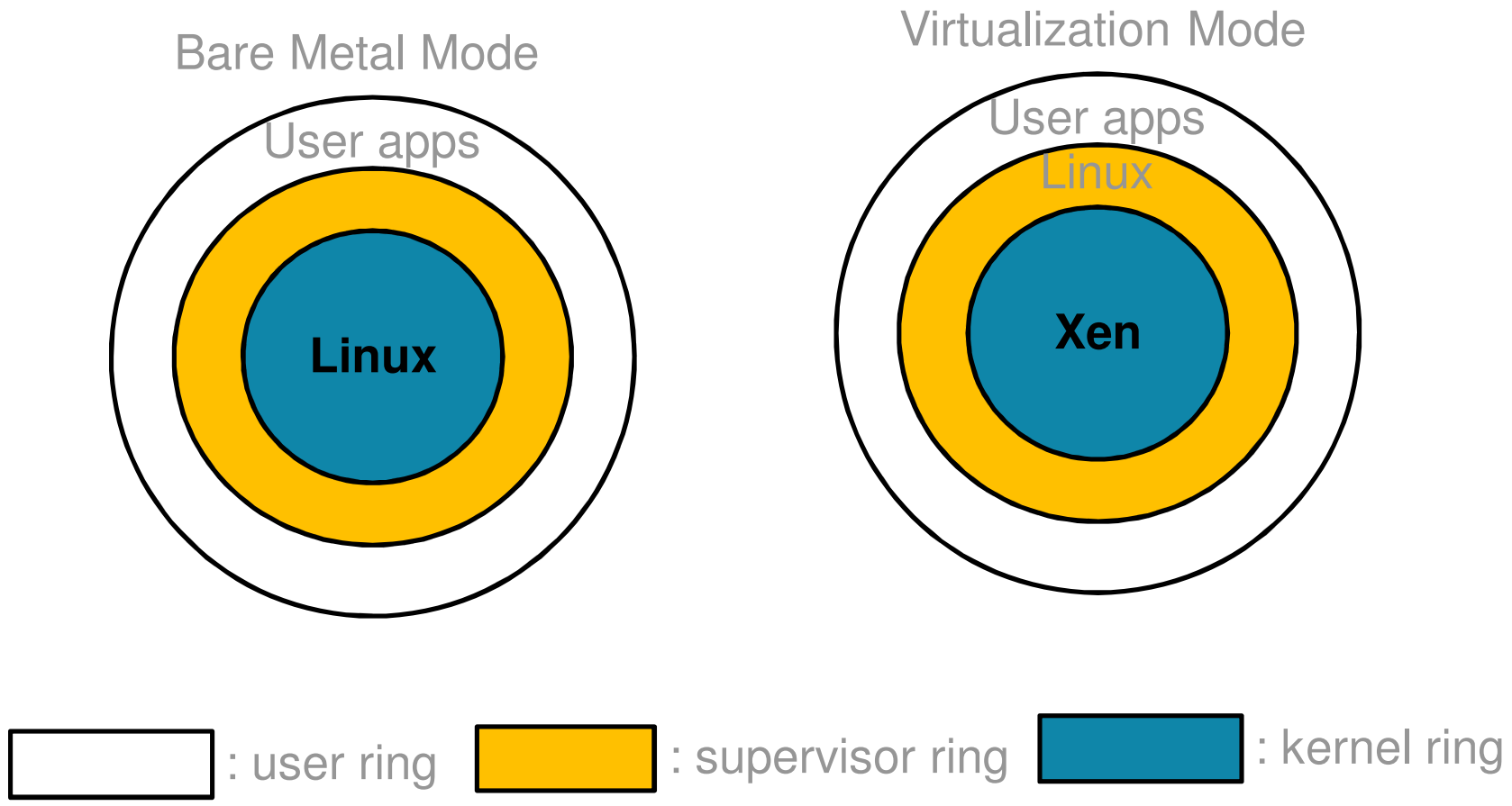
Proposed Model in Xen



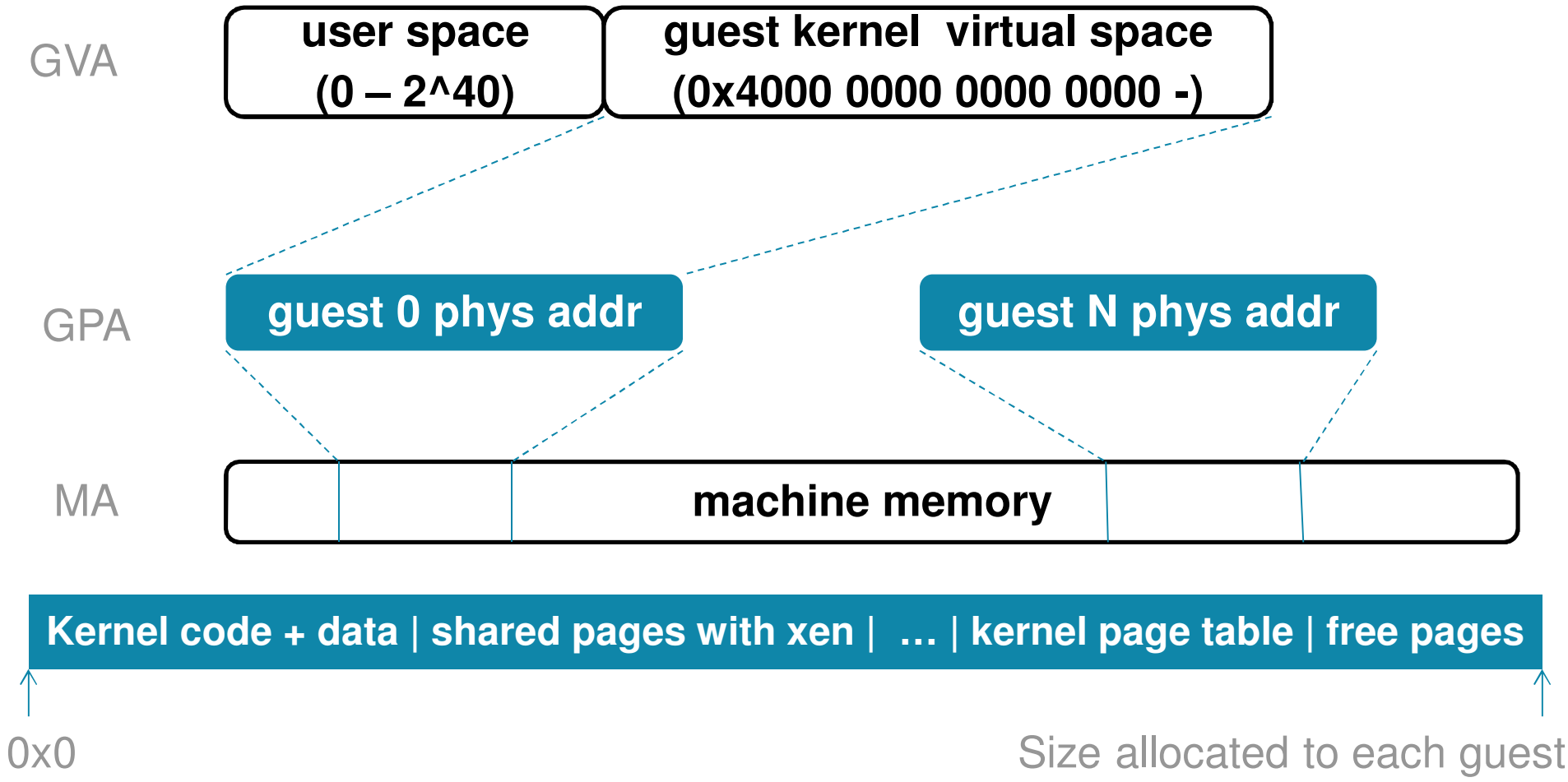
Outline

- CPU Virtualization (mips64r2 only)
  - Memory virtualization
  - Instruction emulation
  - Exception handling
  - Event Channel and Timer Interrupt
- Preliminary Benchmarking Results
- Summary and Future Work

Change of Privilege Levels

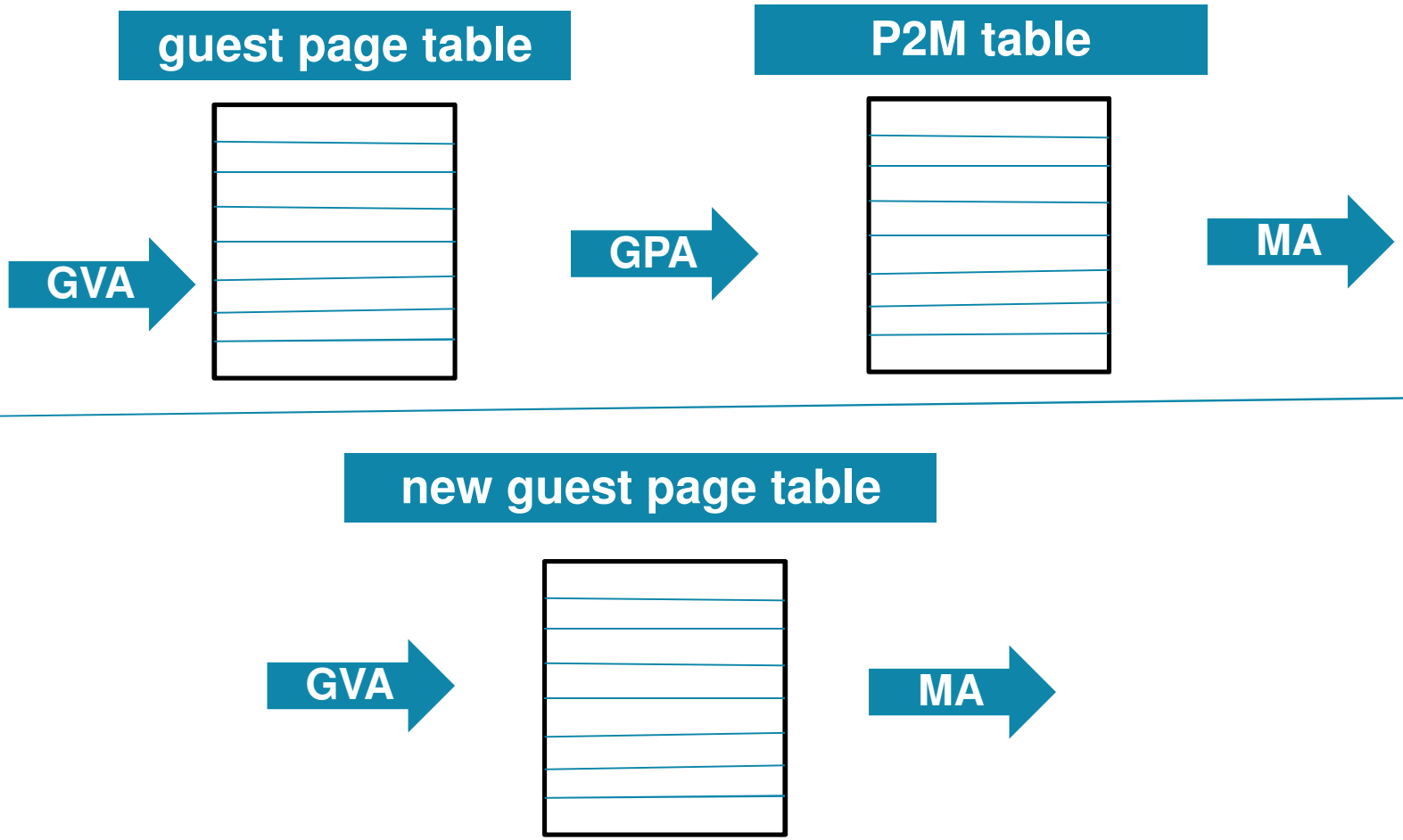


Address Spaces

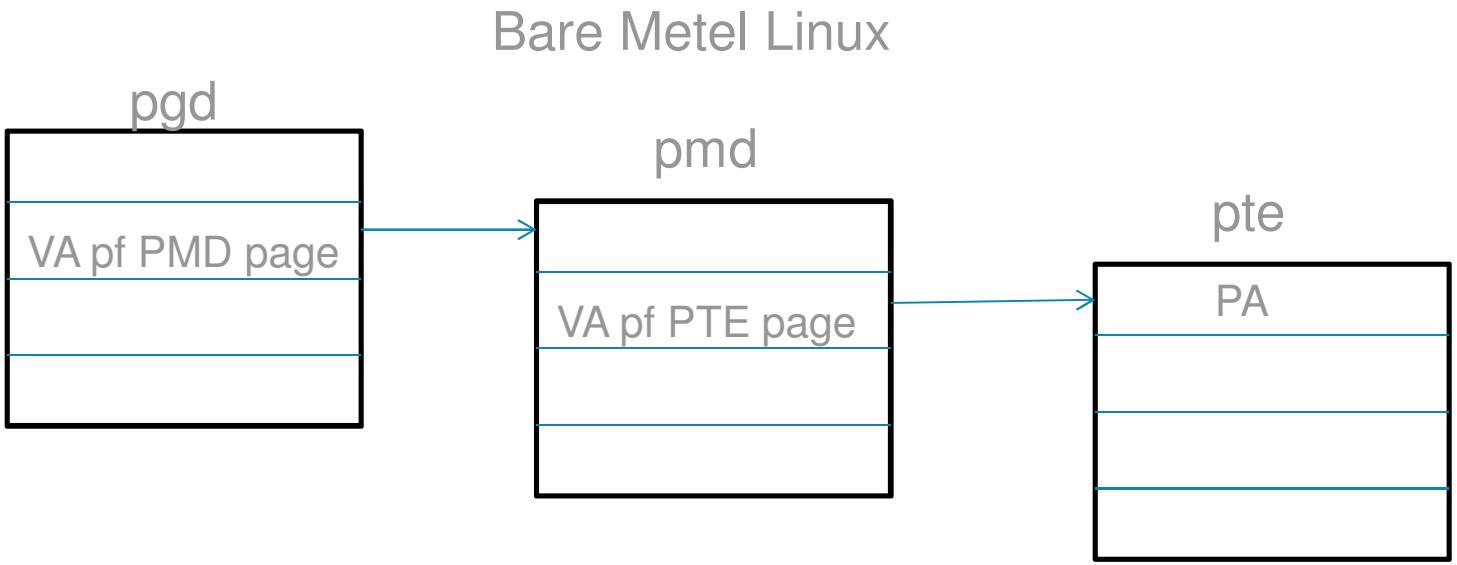


Xen in unmapped space

Page Table Management

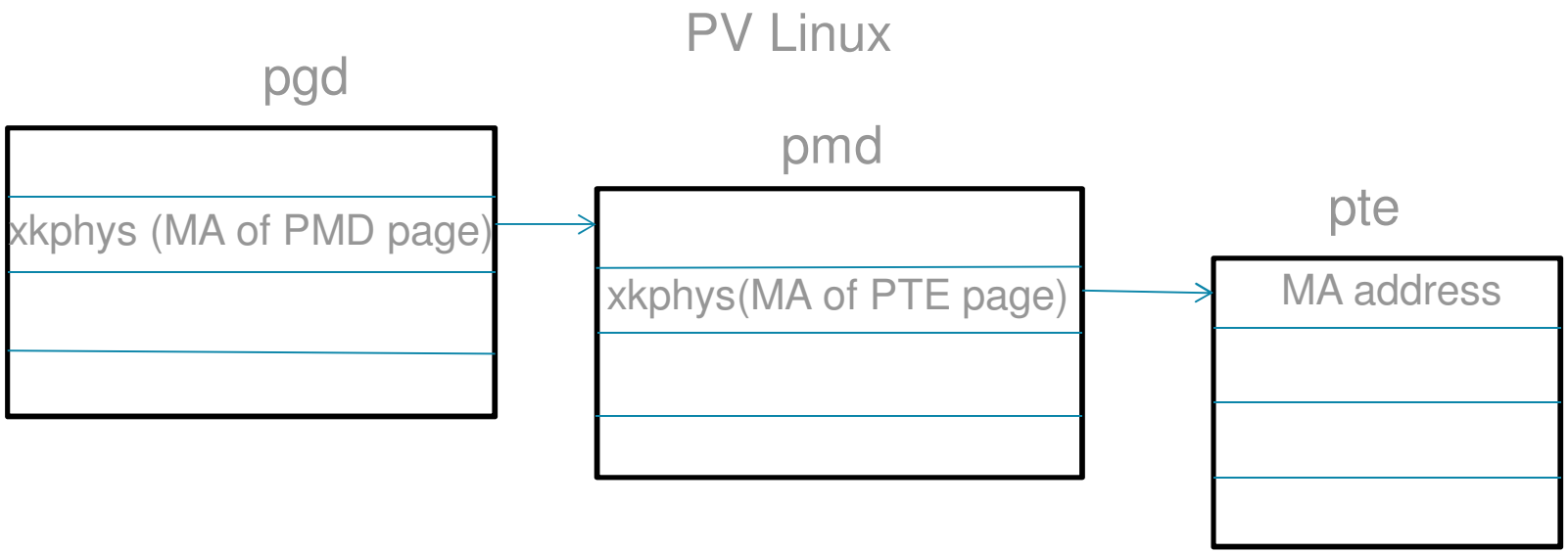


Page Table Layout



pgd: page global directory
pmd: page middle directory
pte: page table entry

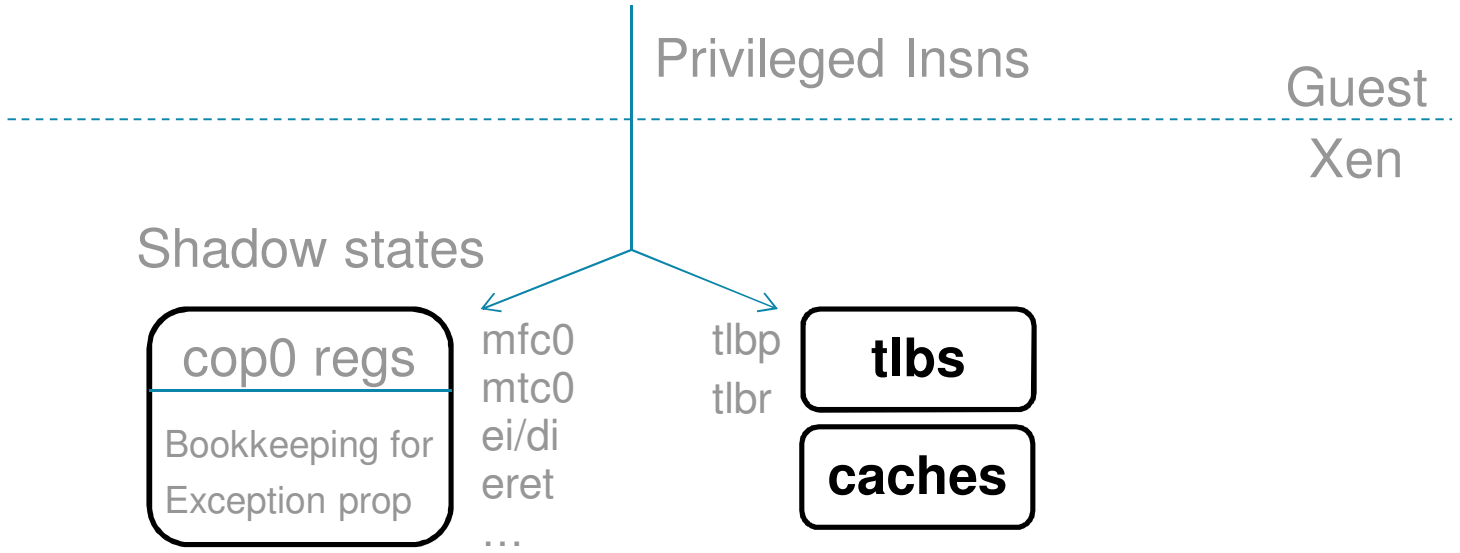
Page Table Layout



xkphys: 64-bit kernel physical space (unmapped)
xkphys: avoid TLB refill during page table walk
Hardware page walker is used

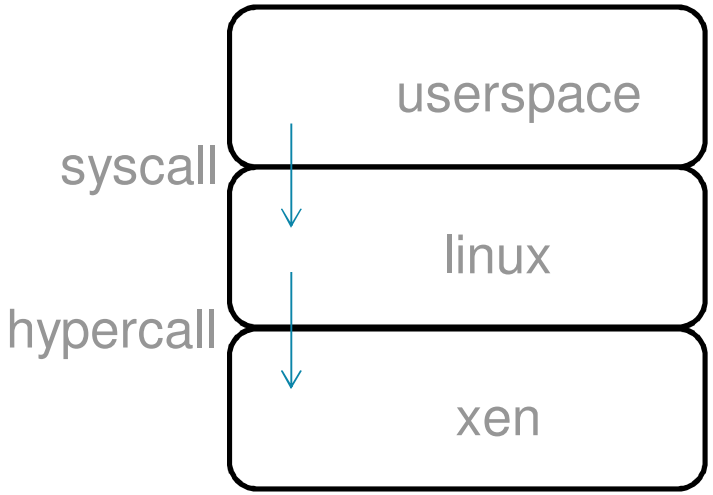
Instruction Emulation

- Privileged instructions in guests get trapped and emulated
- XEN trap handlers decipher the instruction and emulate appropriately
- A few instructions cause hardware state to change, while others change the shadow state
- Shadow state is maintained per virtual cpu of domains



Hypercalls

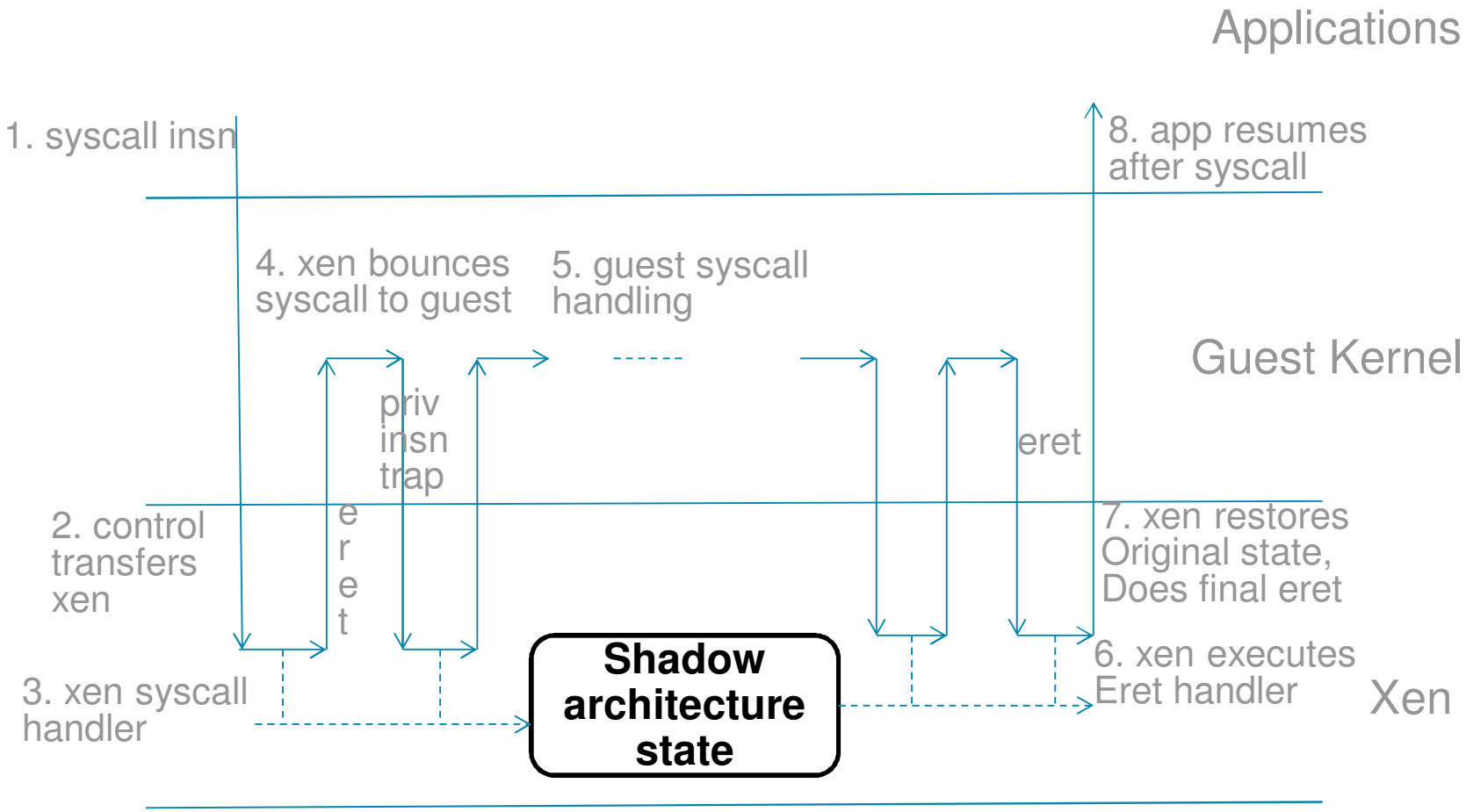
- The service API between guests and xen
- Analogous to system calls between userspace and linux
- Used when a particular service is requested or the overhead of trap and emulate is high
- Implemented using the “syscall” instruction
- Sample uses: vcpu creation, request cache flush, etc



Exception Handling

- Exceptions triggered by guests handled by xen
  - Hypercalls
  - Address error exception
  - Privileged instruction traps
- Exceptions triggered by userspace bounced into guests
  - Guests register callbacks for exception entry points such as general exception vector etc
  - Xen maintains shadow state to return to userspace after the propagated exception is handled
  - Interrupts injected into guests while the bounced exception is handled, retaining regular linux semantics

A syscall example



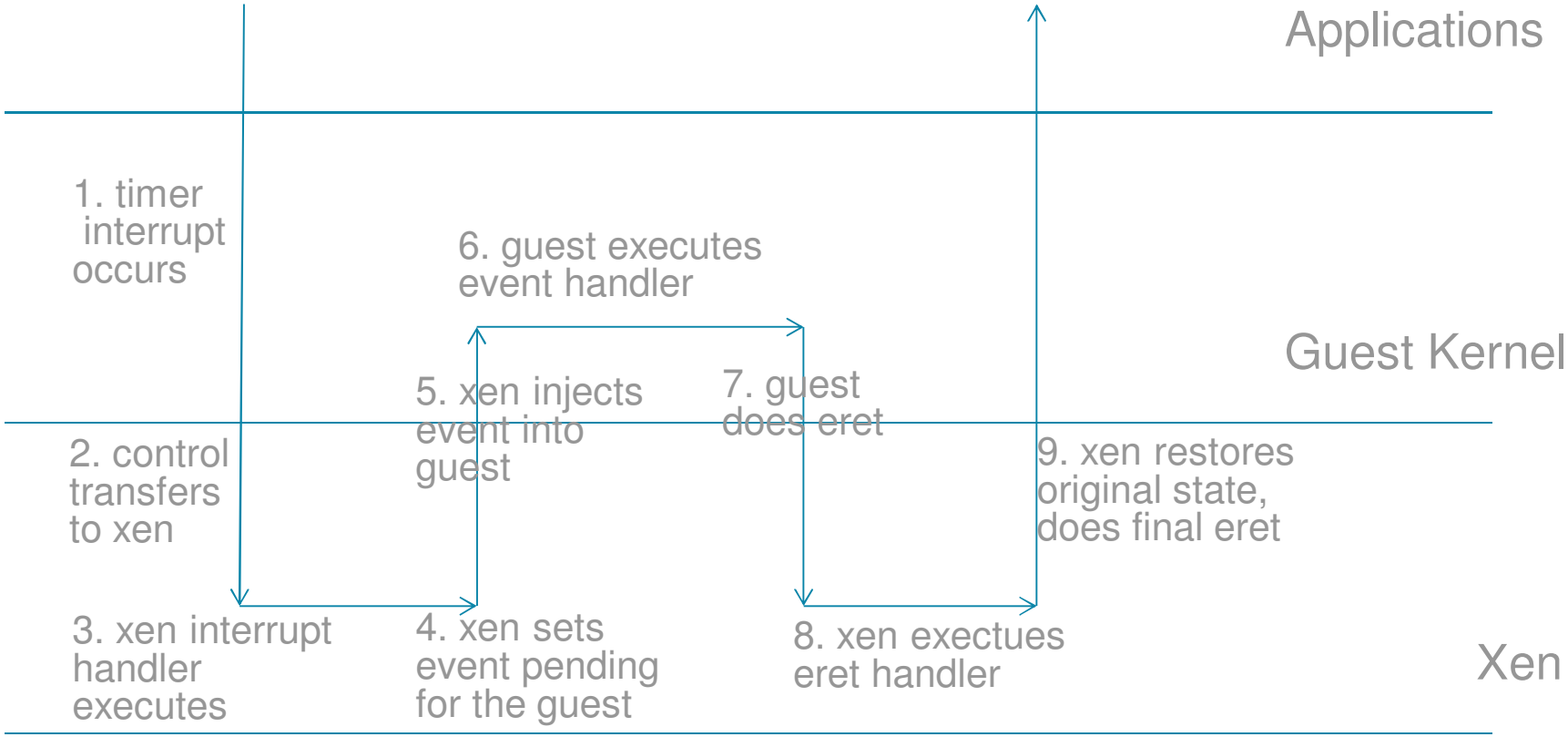
Event Channels

- Events: asynchronous notifications to domains (akin to signals in Unix)
- Event channels: abstract duplex communication channels (akin to sockets): <dom1, port1; dom2, port2>
- Interrupts are mapped to events
  - Intradomain & interdomain events (e.g., domU console)
  - Virtual IRQ (e.g., timer interrupts)
  - Physical IRQ (e.g., passthrough device interrupts)
- Delivered through a callback function

Time Management

- Time keeping in xen
  - Maintaining system time – Using XLP-specific internal global 64bit free running counter
  - Requesting timer interrupts: done by maintaining per-cpu timer list and programming the count/compare registers
- Guest OS
  - Xen clocksource: a hardware abstraction for a free running counter to maintain system time
    - Maintained through timestamps written by xen on a shared page
  - Xen clockevent: an interface to request timer interrupts
    - Done using the hypercall to program a single shot timer in xen

Timer Interrupt Illustration



Performance Optimization

- Expose certain shadow states for guest OS to avoid excessive exception start/end cost
- When guest executes “wait” insn, xen tries to “wait” also to avoid burning cpu resources

Preliminary Benchmarking Result

- XLP832: 8 cores, 4 threads each core, 1.0GHZ
  - Only 1 core, 4 threads used for measuring time
- Intel Core 2: 2 cores, 1 thread per core, 2.4GHZ
  - Not using hardware virtualization extensions
- CPU/Memory intensive benchmarks like dhrystone, eembc, coremark, etc.
  - 0 – 5% slowdown for dom0 compared to bare metal linux, for both x86 and XLP
- Hackbench (a lot of system calls)
  - 2X slowdown for dom0 compared to bare metal linux, for both x86 and XLP
- No noticeable performance difference between dom0 and domU on XLP

Summary and Future Work

- A MIPS port of xen paravirtualization has implemented
  - MMU, exception/interrupt handling, etc.
  - Comparable performance to x86 for bare metal vs. xen
- Currently, our implementation uses xen 3.4.0 for xen hypervisor, 4.0.0 for xen tools, linux 2.6.32 for PV linux, so we need to
  - Update to latest versions of Xen
  - Submit patches upstream
- More work on I/O paravirtualization
- Ongoing collaboration with MIPS Technologies

Thank You