Xen and the Art of Virtualization

Ian Pratt

Keir Fraser, Steve Hand, Christian Limpach,
Dan Magenheimer (HP), Mike Wray (HP),
R Neugebauer (Intel), M Williamson (Intel)

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Outline

- Virtualization overview
- Xen 2.0 Features
- Architecture
- Performance
- Xen para-virtualized binary interface
  - Linux 2.6 on Xen/x86
- Work in Progress

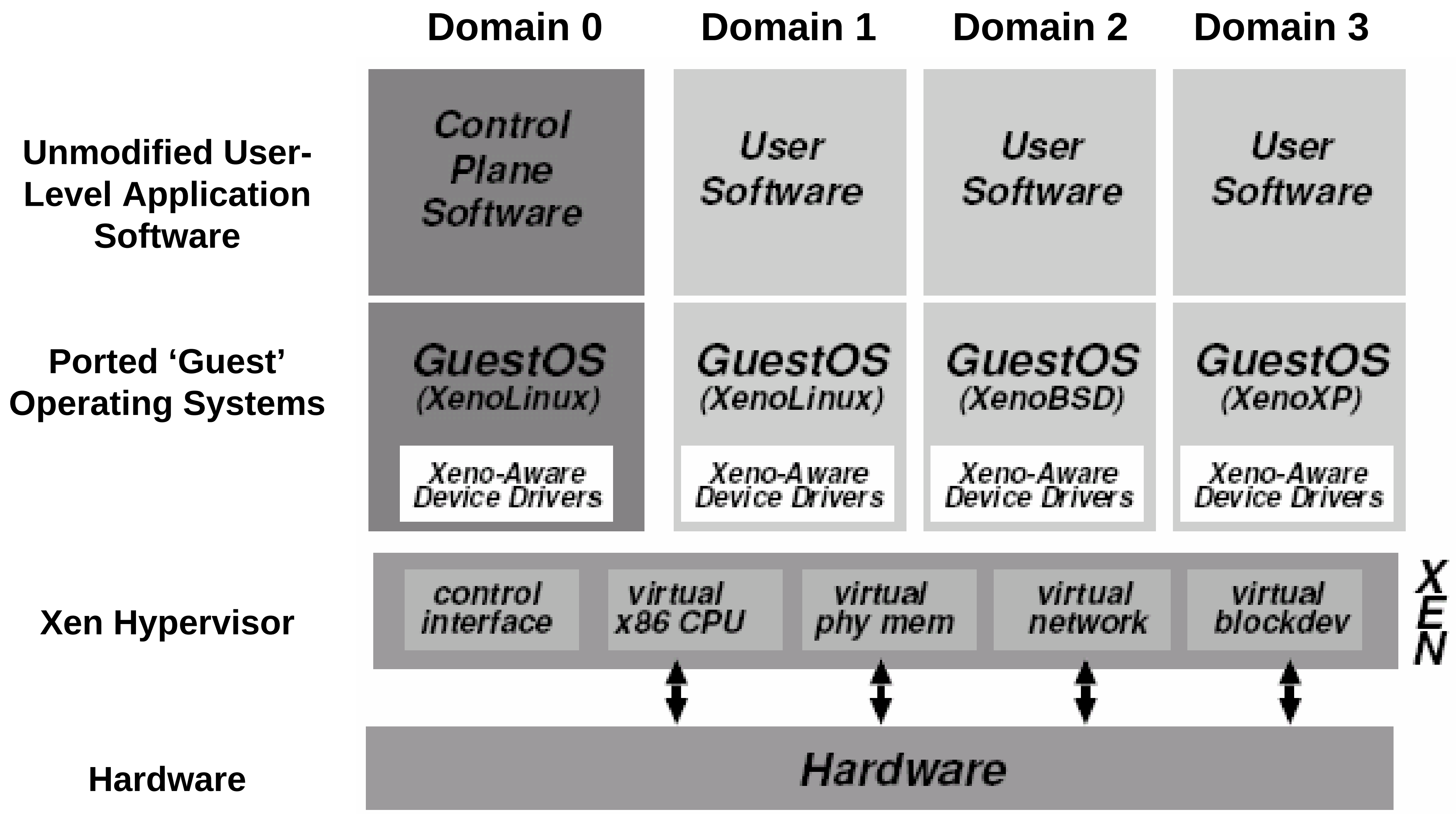
Virtualization Overview

- Single OS image: Ensim, Vservers, CKRM
  - Group user processes into resource containers
  - Hard to get strong isolation
- Full virtualization: VMware, VirtualPC
  - Run multiple unmodified guest OSes
  - Hard to efficiently virtualize x86
- Para-virtualization: UML, Xen
  - Run multiple guest OSes ported to special arch
  - Arch Xen/x86 is very close to normal x86

Xen 2.0 Features

- Secure isolation between VMs
- Resource control and QoS
- Only guest kernel needs to be ported
  - All user-level apps and libraries run unmodified
  - Linux 2.4/2.6, NetBSD, FreeBSD, WinXP
- Execution performance is close to native
- Live Migration of VMs between Xen nodes
- Xen hardware support:
  - SMP; x86 / x86_64 / ia64; all Linux drivers

Xen 1.2 Architecture



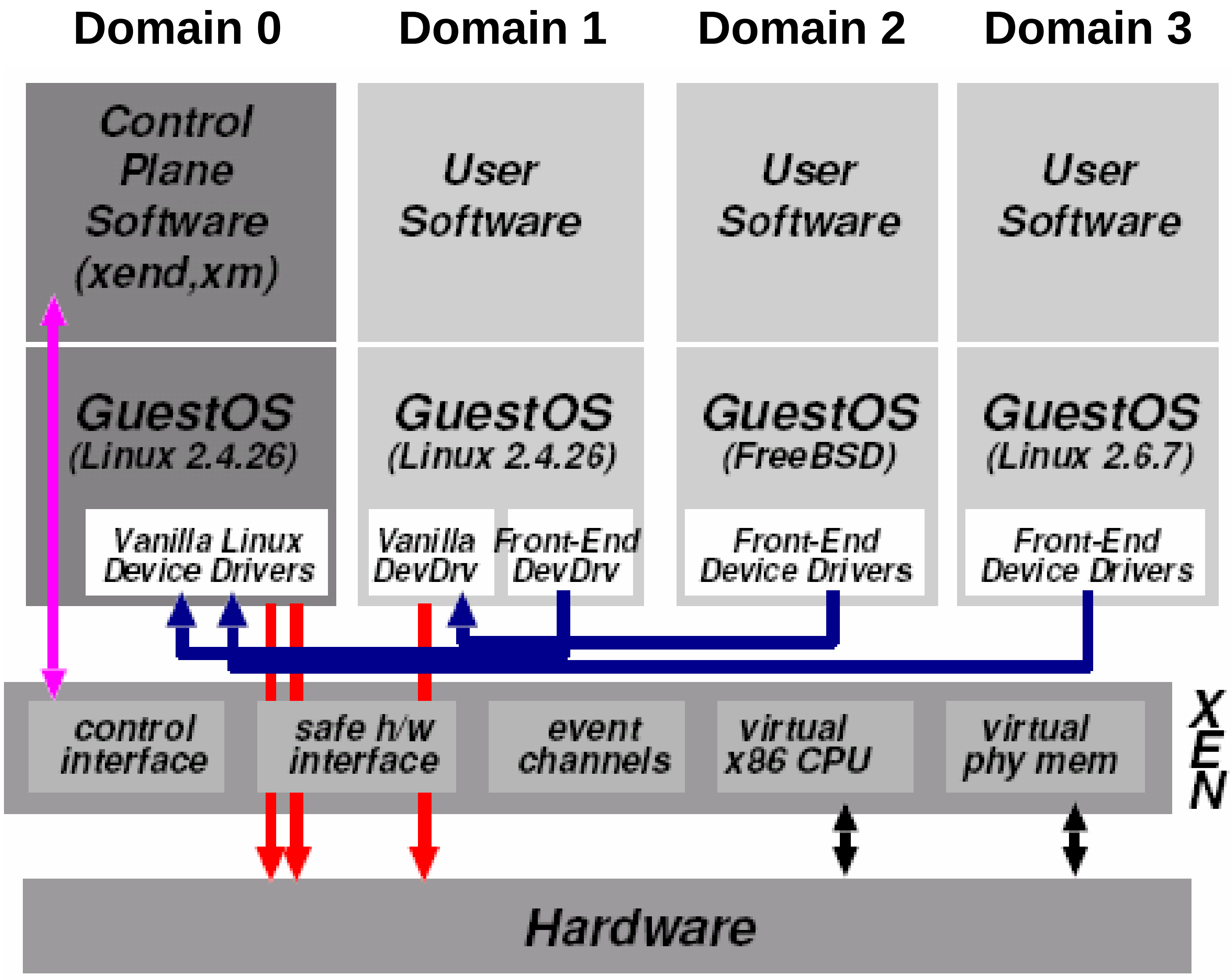
Xen 2.0 Architecture

Unmodified User-Level Application Software

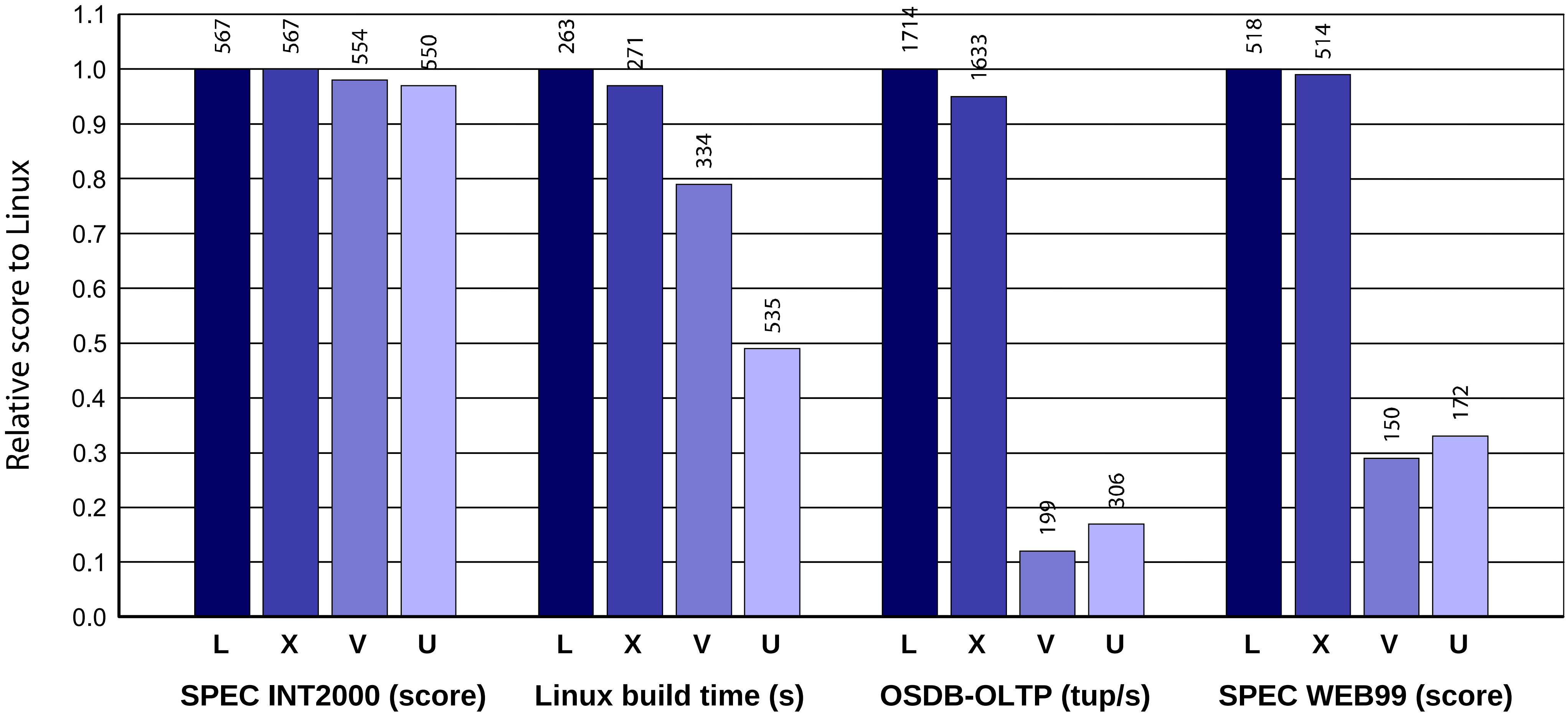
Ported 'Guest' Operating Systems

Xen Hypervisor

Hardware



System Performance



Benchmark suite running on Linux (L), Xen (X), VMware Workstation (V), and UML (U)

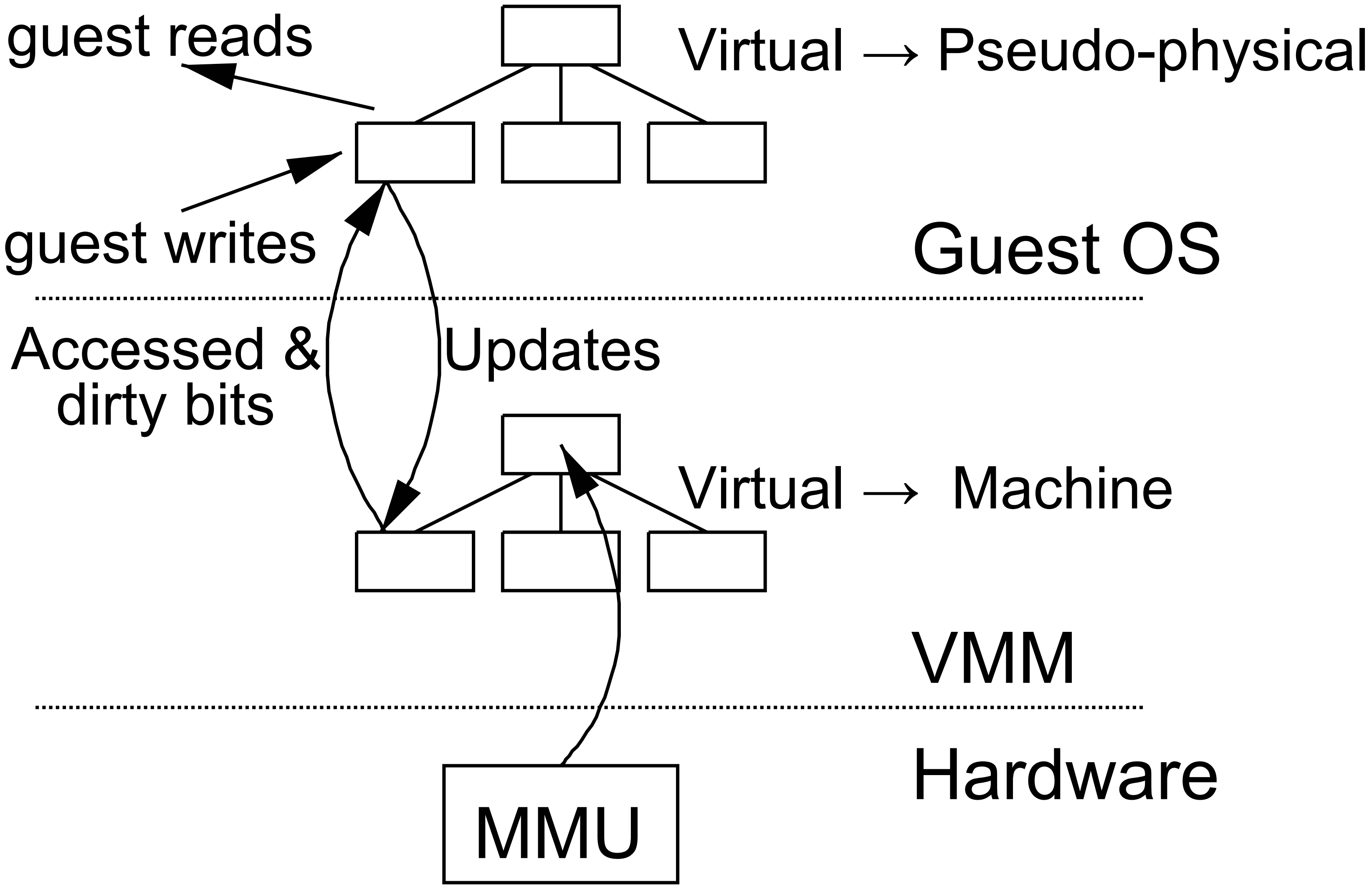
Xen Para-Virtualization

- Arch Xen/x86 – like x86, but replace privileged instructions with Xen hypercalls
  - Avoids binary rewriting and fault trapping
  - For Linux 2.6, only arch-dep files modified
- Modify OS to understand virtualised env.
  - Wall-clock time vs. virtual processor time
    - Xen provides both types of alarm timer
  - Expose real resource availability
    - Enables OS to optimise behaviour

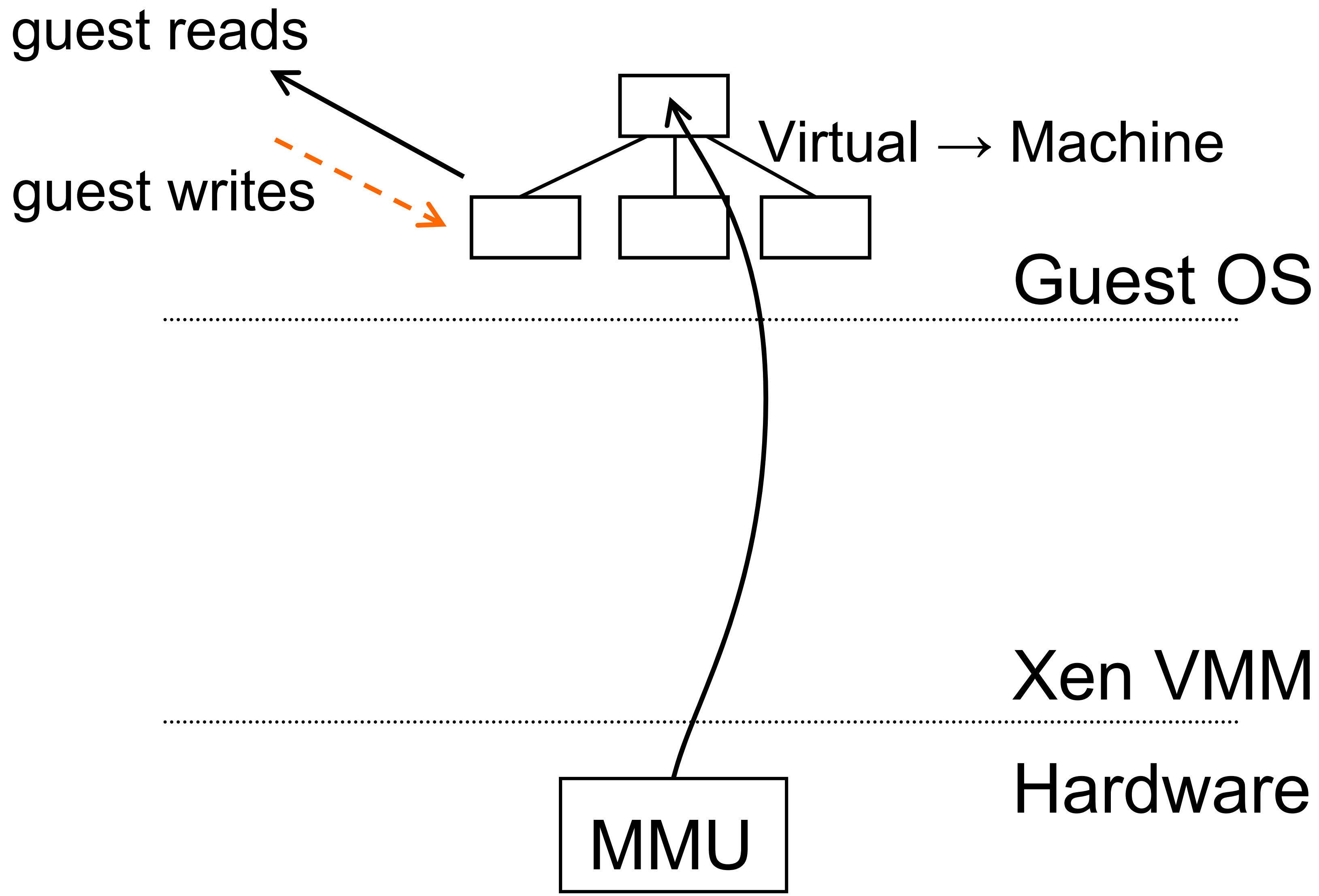
x86 CPU virtualization

- Xen runs in ring 0 (most privileged)
- Ring 1/2 for guest OS, 3 for user-space
  - GPF if guest attempts to use privileged instr
- Xen lives in top 64MB of linear addr space
  - Segmentation used to protect Xen as switching page tables too slow on standard x86
- Hypercalls jump to Xen in ring 0
- Guest OS may install ‘fast trap’ handler
  - Direct ring user-space to guest OS system calls
- MMU virtualisation: shadow vs. direct-mode

MMU Virtualizion : Shadow-Mode



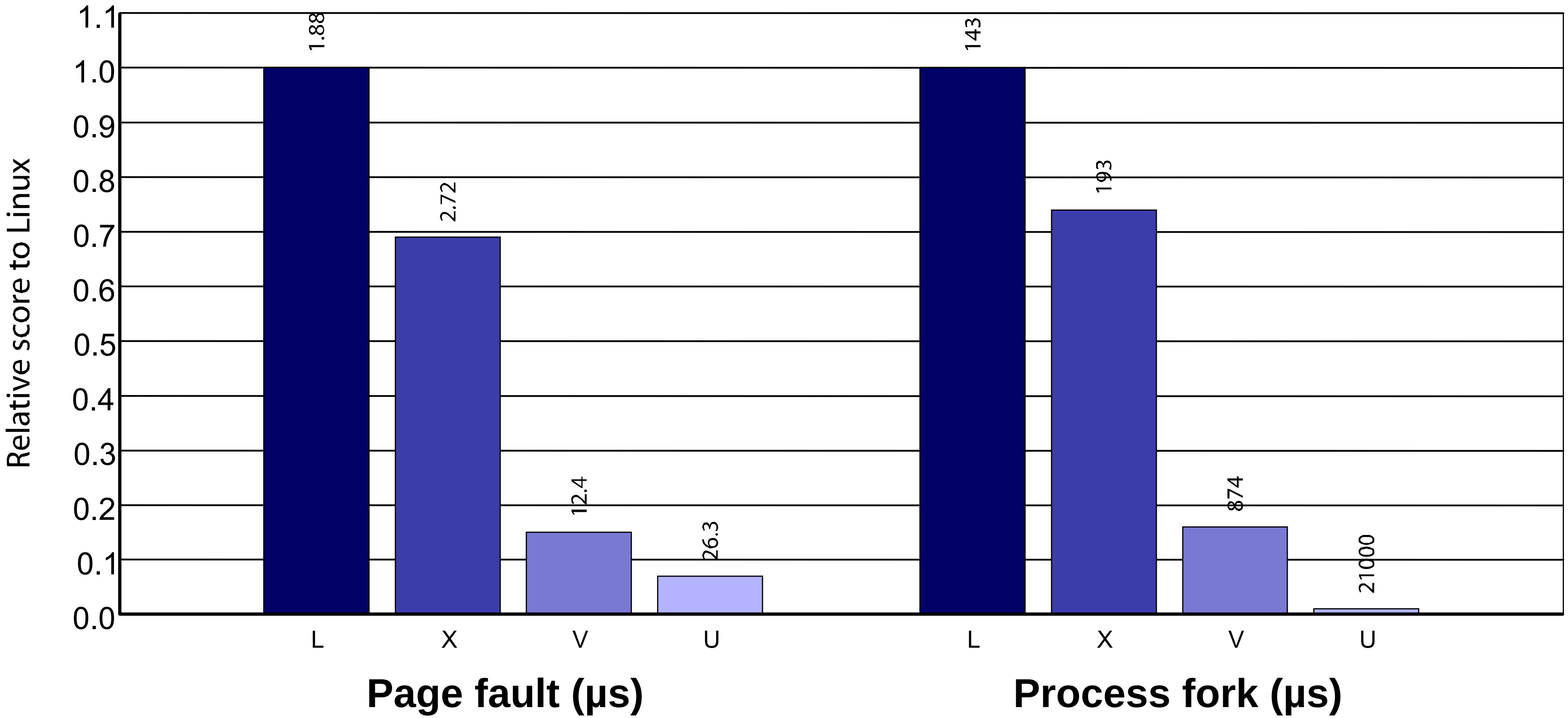
MMU Virtualization : Direct-Mode



Para-Virtualizing the MMU

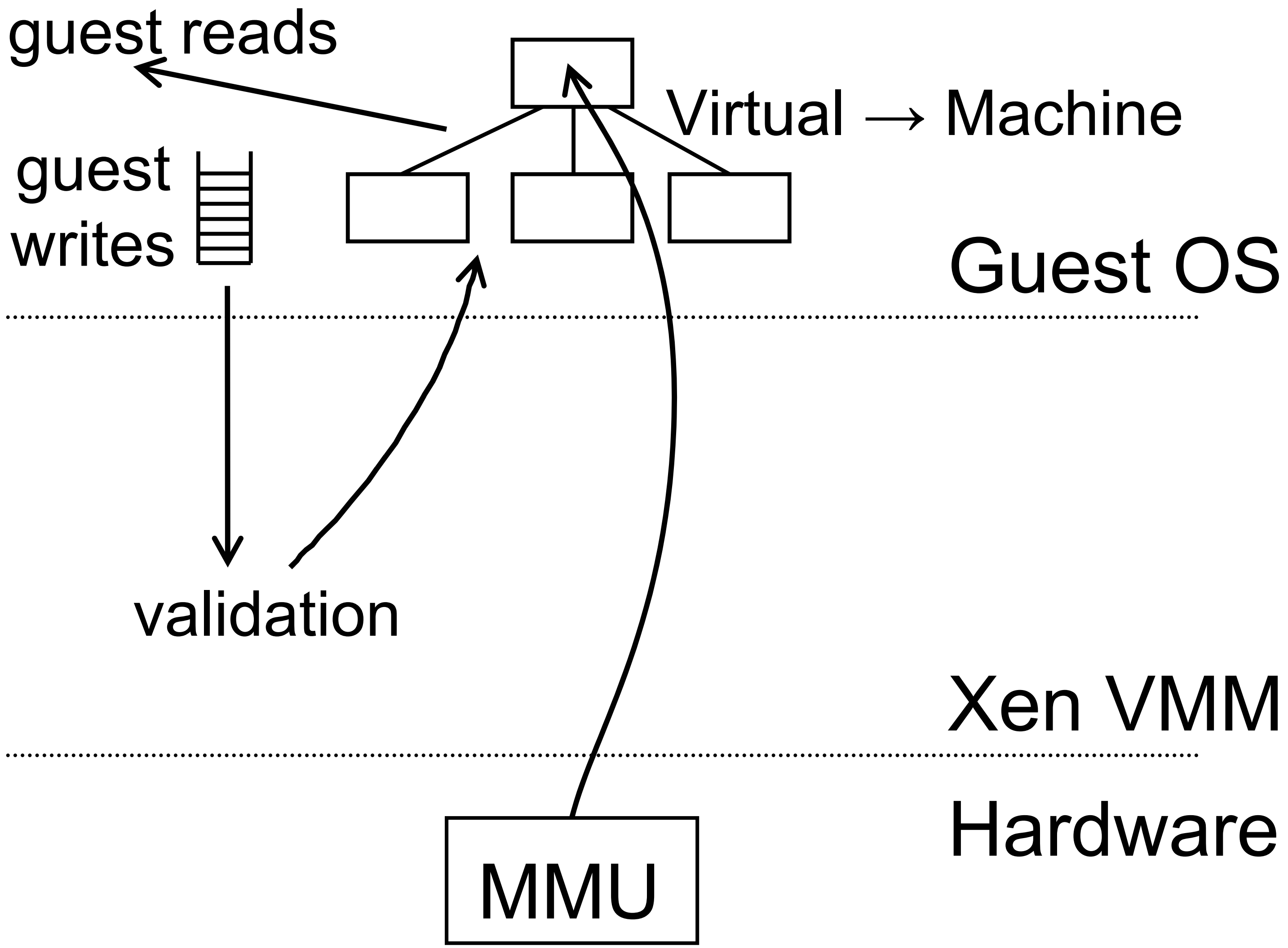
- Guest OSes allocate and manage own PTs
  - Hypercall to change PT base
- Xen must validate PT updates before use
  - Updates may be queued and batch processed
- Validation rules applied to each PTE:
  1. Guest may only map pages it owns*
  2. Pagetable pages may only be mapped RO
- Xen tracks page ownership and current use
  - L4/L3/L2/L1/Normal (plus ref count)

MMU Micro-Benchmarks

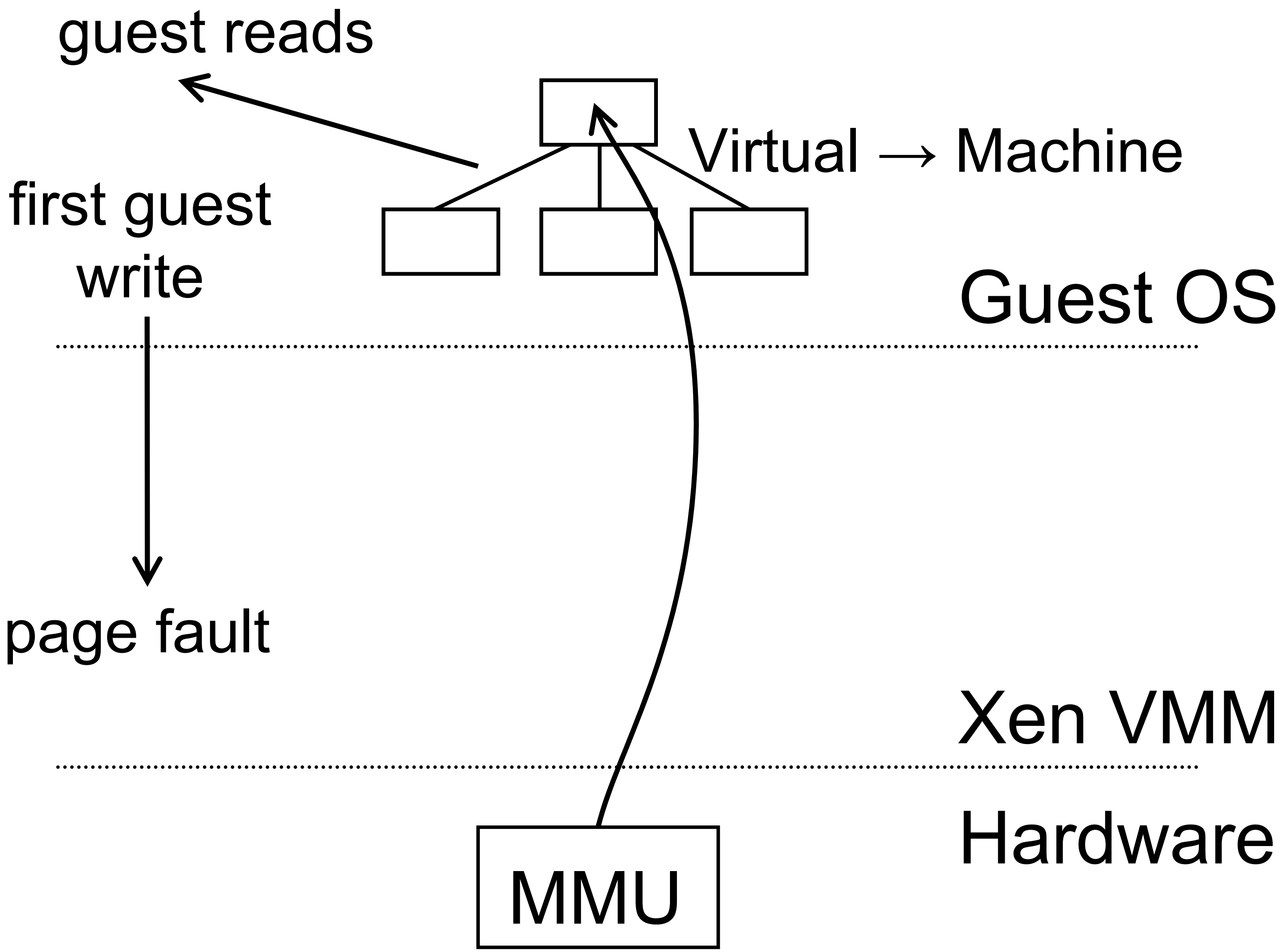


lmbench results on Linux (L), Xen (X), VMWare Workstation (V), and UML (U)

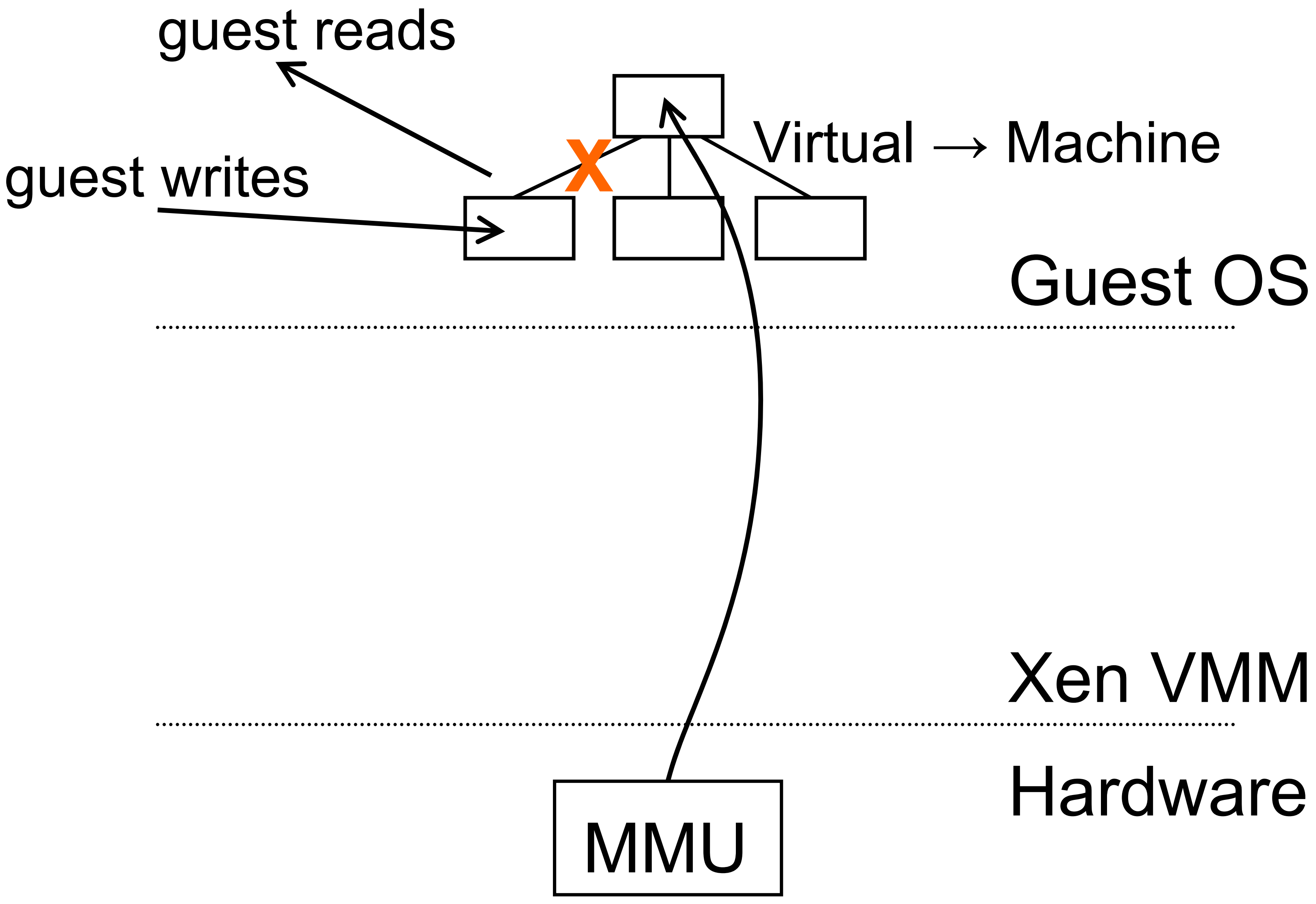
Queued Update Interface (Xen 1.2)



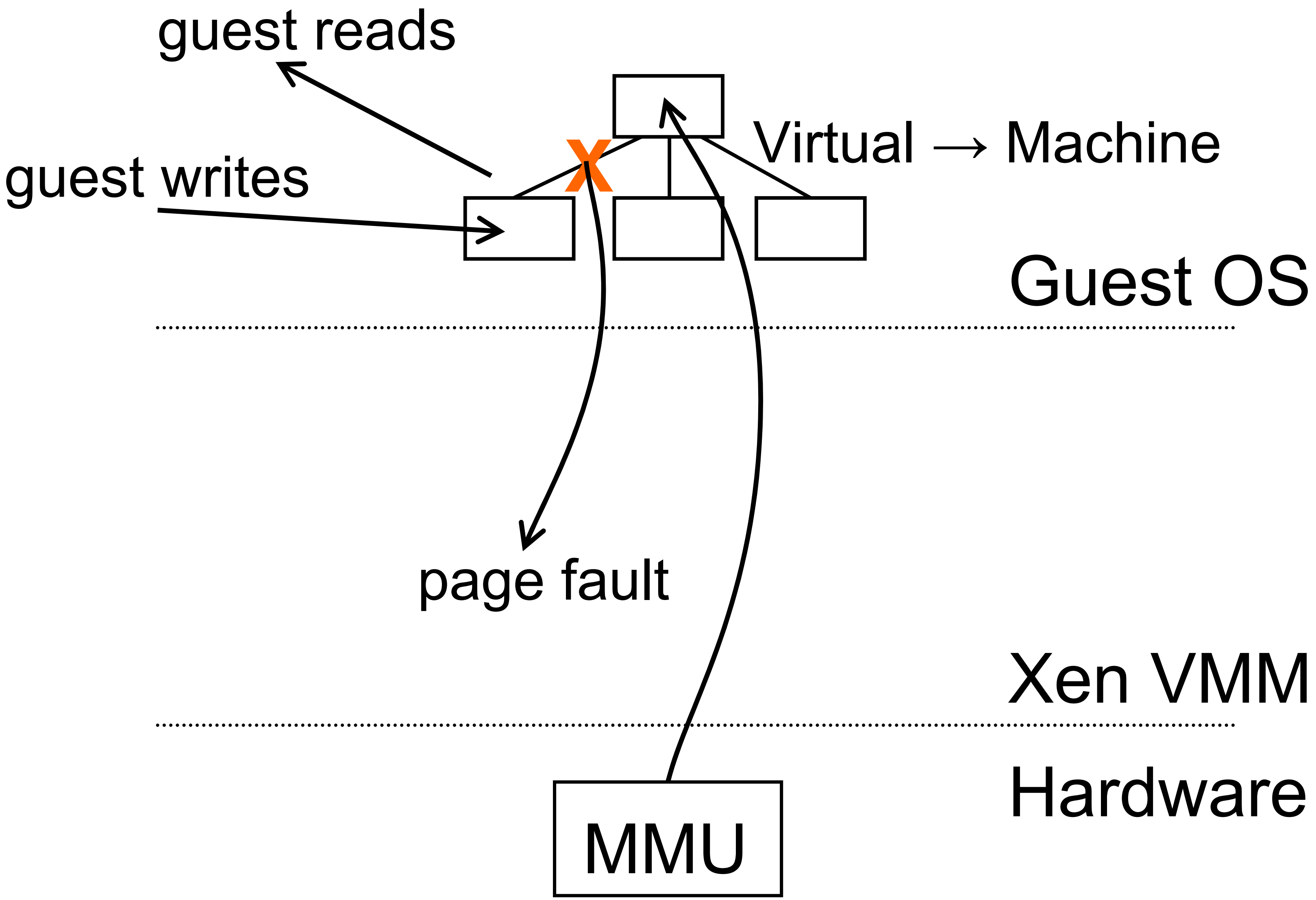
Writeable Page Tables (1)



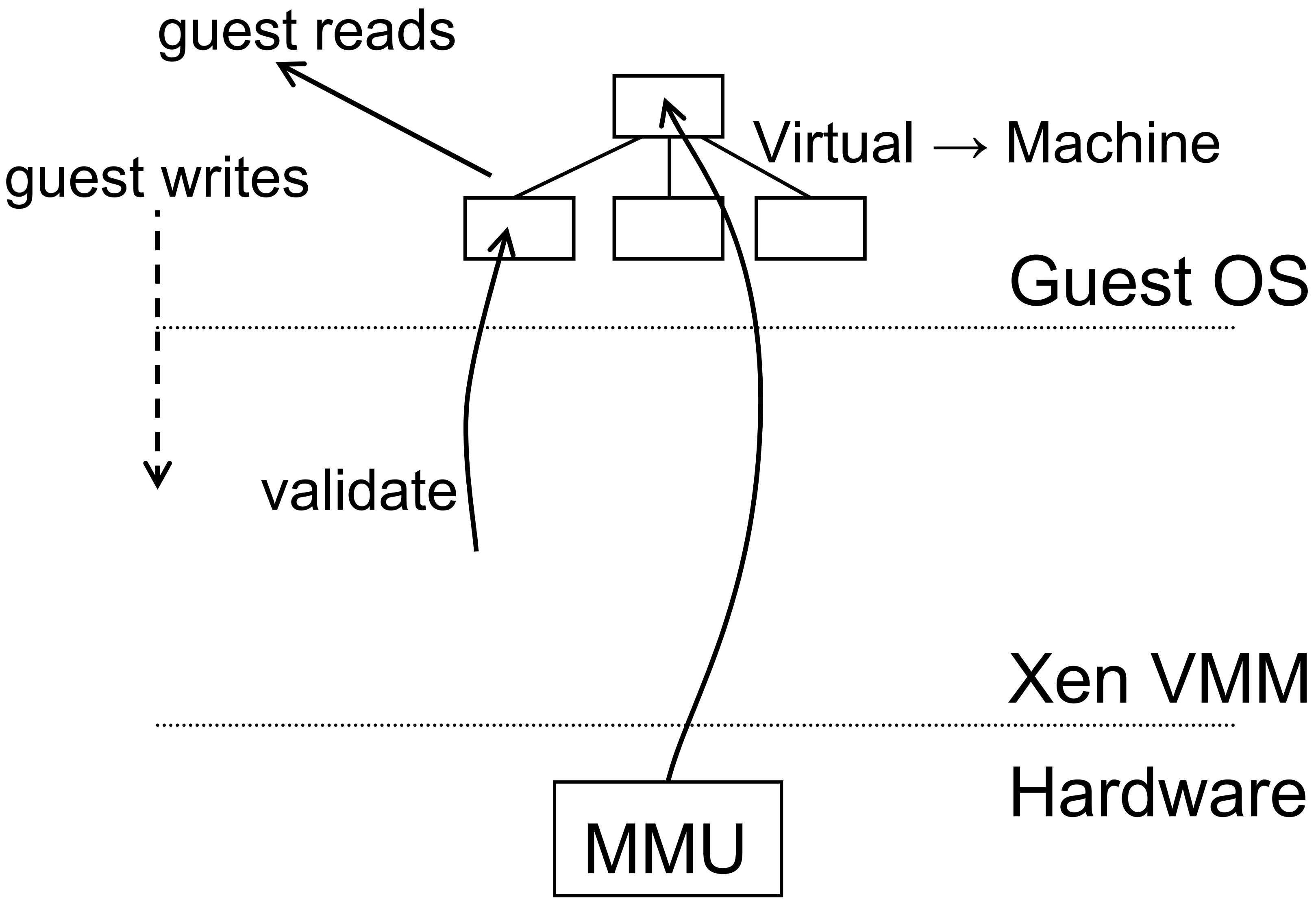
Writeable Page Tables (2)



Writeable Page Tables (3)



Writeable Page Tables (4)



Segmentation Support

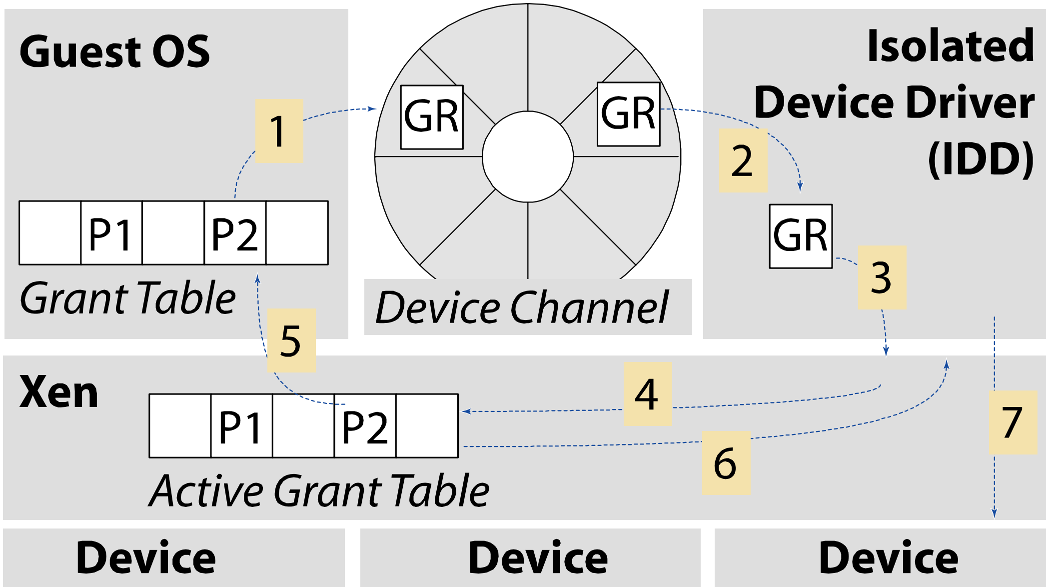
- Segmentation req'd by thread libraries
  - Xen supports virtualised GDT and LDT
  - Segment must not overlap Xen 64MB area
  - NPT TLS library uses 4GB segs with –ve offset!
    - Emulation plus binary rewriting required ☹
- x86_64 has no support for segment limits
  - Forced to use paging, but only have 2 prot levels
  - Xen ring 0; OS and user in ring 3 w/ PT switch
    - Opteron's TLB flush filter CAM makes this fast

I/O Architecture

- Xen *IO-Spaces* delegate guest OSes protected access to specified h/w devices
  - Virtual PCI configuration space
  - Virtual interrupts
- Devices are virtualised and exported to other VMs via *Device Channels*
  - Safe asynchronous shared memory transport
  - ‘Backend’ drivers export to ‘frontend’ drivers
  - Net: use normal bridging, routing, iptables
  - Block: export any blk dev e.g. sda4,loop0,vg3

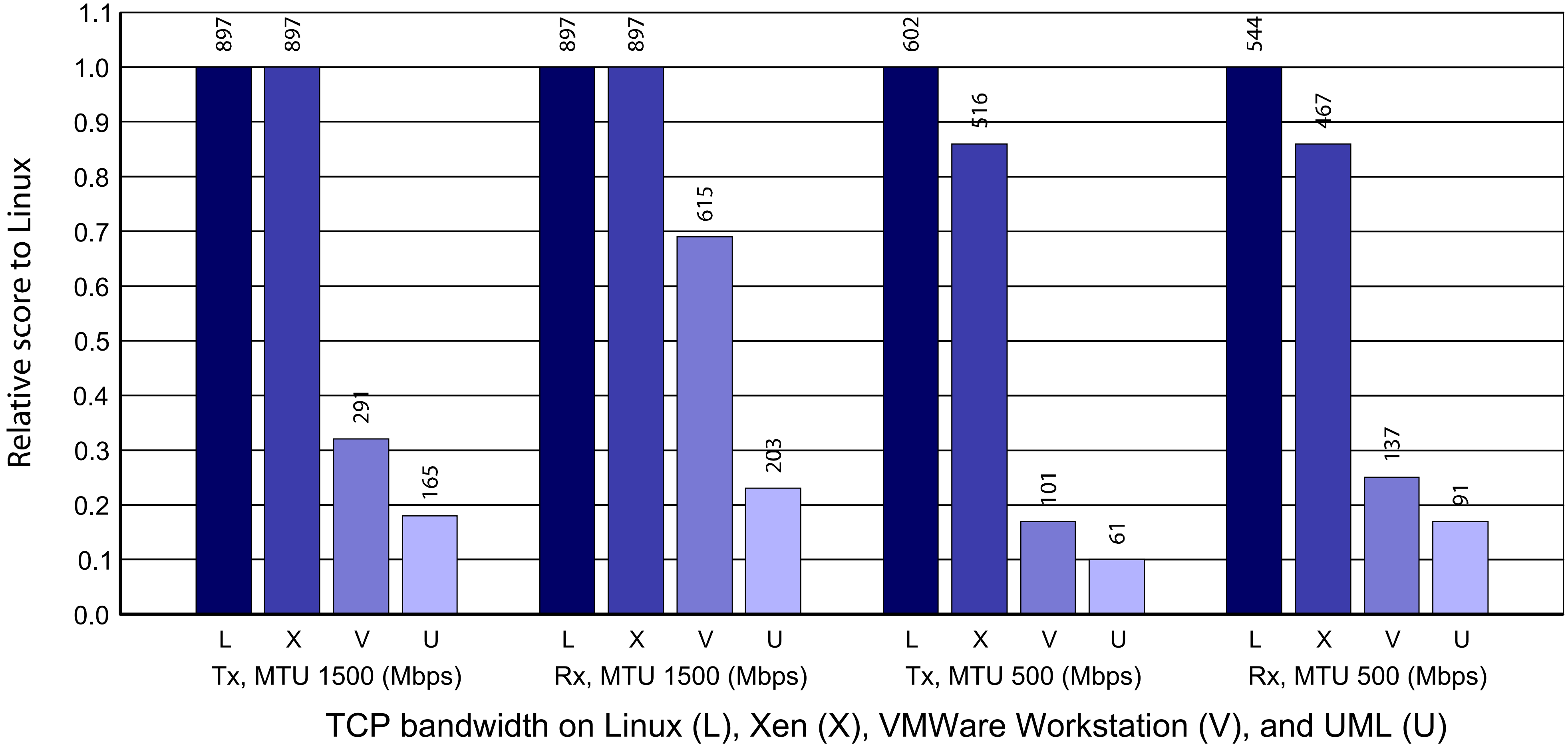
Device Channel Interface

Guest Requests DMA:
1. Grant Reference for Page P2 placed on device channel
2. IDD removes GR
3. Sends pin request to Xen



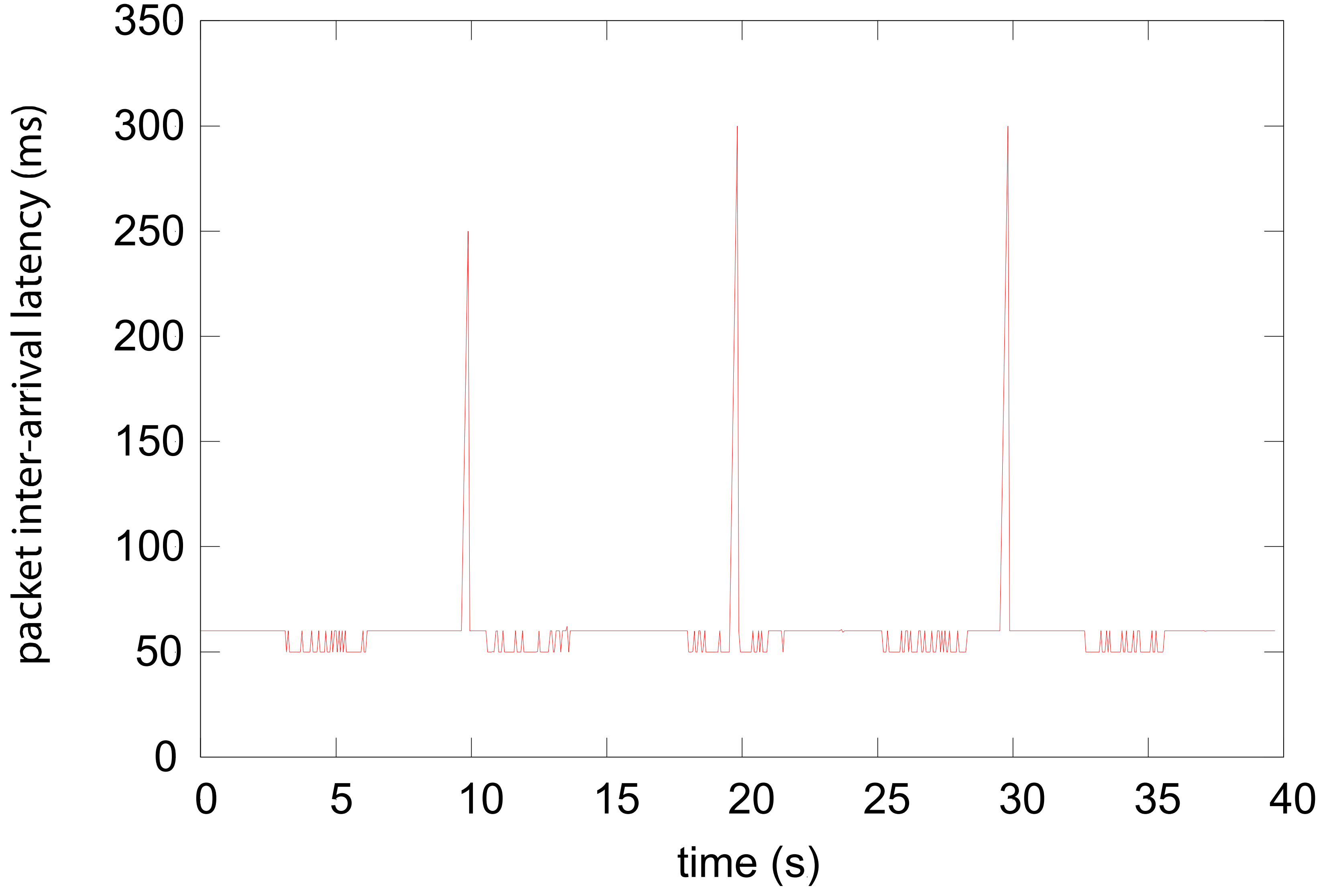
4. Xen looks up GR in active grant table
5. GR validated against Guest (if necessary)
6. Pinning is acknowledged to IDD
7. IDD sends DMA request to device

TCP results



Relative score to Linux
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
897
897
291
165
897
897
615
203
602
516
101
61
544
467
137
91
L X V U
Tx, MTU 1500 (Mbps)
L X V U
Rx, MTU 1500 (Mbps)
L X V U
Tx, MTU 500 (Mbps)
L X V U
Rx, MTU 500 (Mbps)
TCP bandwidth on Linux (L), Xen (X), VMWare Workstation (V), and UML (U)

Isolated Driver VMs



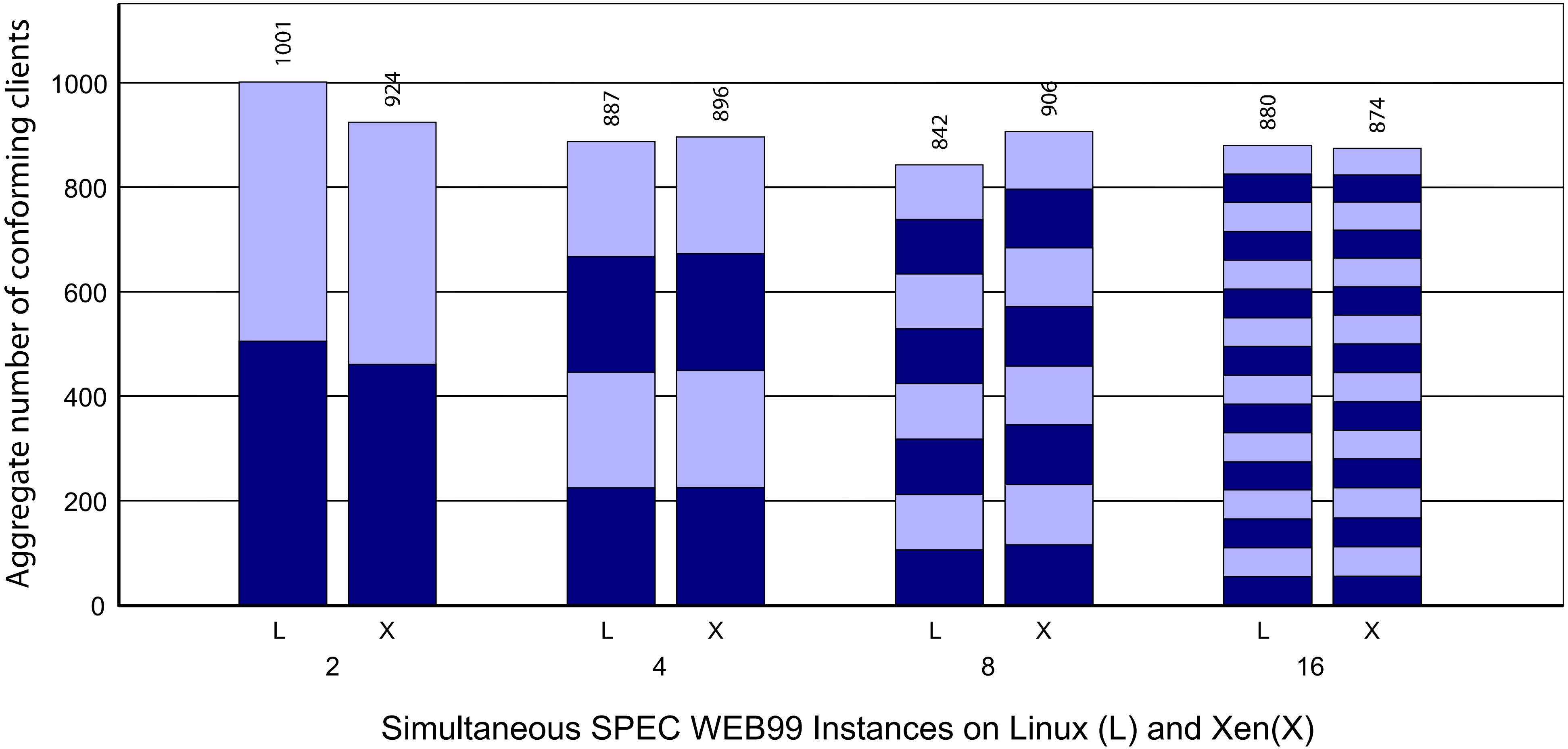
Live migration for clusters

- Pre-copy approach: VM continues to run
- ‘lift’ domain on to shadow page tables
  - Bitmap of dirtied pages; scan; transmit dirtied
  - Atomic ‘zero bitmap & make PTEs read-only’
- Iterate until no forward progress, then stop VM and transfer remainder
- Rewrite page tables for new MFNs; Restart
- Migrate MAC or send unsolicited ARP-Reply
- Downtime typically 10’s of milliseconds
  - (though very application dependent)

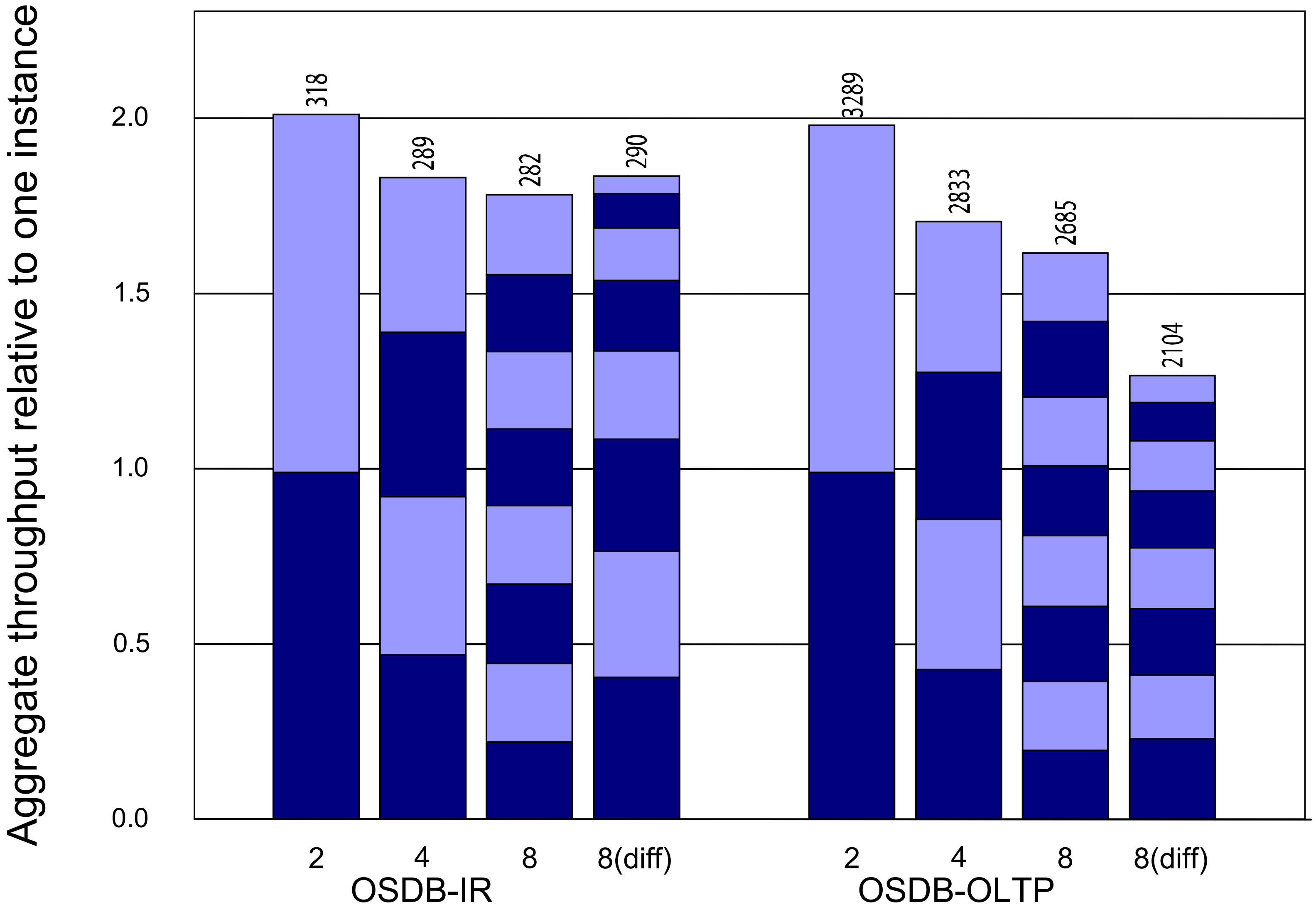
Scalability

- Scalability principally limited by Application resource requirements
  - several 10's of VMs on server-class machines
- Balloon driver used to control domain memory usage by returning pages to Xen
  - Normal OS paging mechanisms can deflate quiescent domains to <4MB
  - Xen per-guest memory usage <32KB
- Additional multiplexing overhead negligible

Scalability



Resource Differentation



Simultaneous OSDB-IR and OSDB-OLTP Instances on Xen

On-Going Work

- xend web control interface
- Cluster management tools
  - Load balancing
- SMP guest OSes (have SMP hosts already)
- Support for Intel VT/LT x86 extensions
  - Will enable full virtualization
- VM Checkpointing
  - Debugging and fault tolerance

Conclusions

- Xen is a complete and robust GPL VMM
- Outstanding performance and scalability
- Excellent resource control and protection
- Linux 2.6 port required no modifications to core code

- http://xen.sf.net