Xen and the Art of Virtualization

Ian Pratt

University of Cambridge

and XenSource

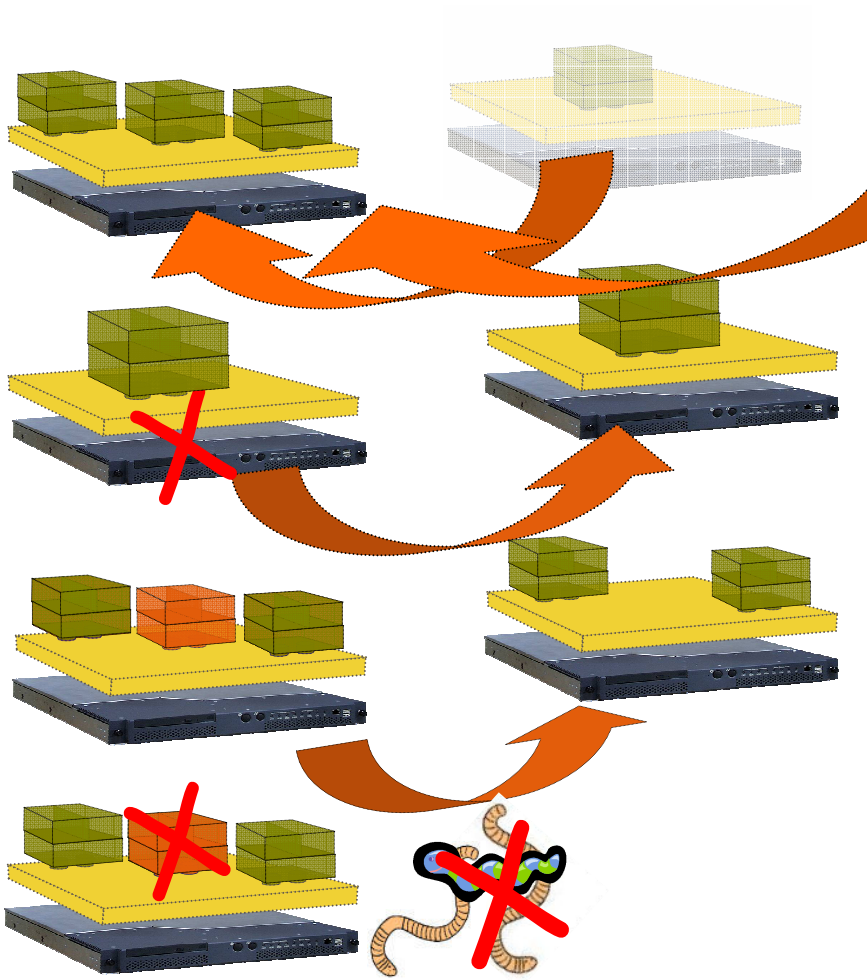
Outline

- Virtualization Overview
- Xen Architecture
- VM Relocation deep dive
- Debugging and profiling
- Xen Research
- Demo
- Questions

Virtualization Overview

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

Virtualization Benefits



- ***Consolidate*** **under-utilized servers**
- ***Avoid downtime*** **with VM Relocation**
- **Dynamically *re-balance workload* to guarantee application SLAs**
- **Enforce *security* policy**

Virtualization Possibilities

- Standing outside the OS looking in:
  - Firewalling / network IDS / Inverse Firewall
  - VPN tunneling; LAN authentication
  - Virus, mal-ware and exploit detection
  - OS patch-level status monitoring
  - Performance monitoring and instrumentation
  - Storage backup and optimization
  - Debugging support

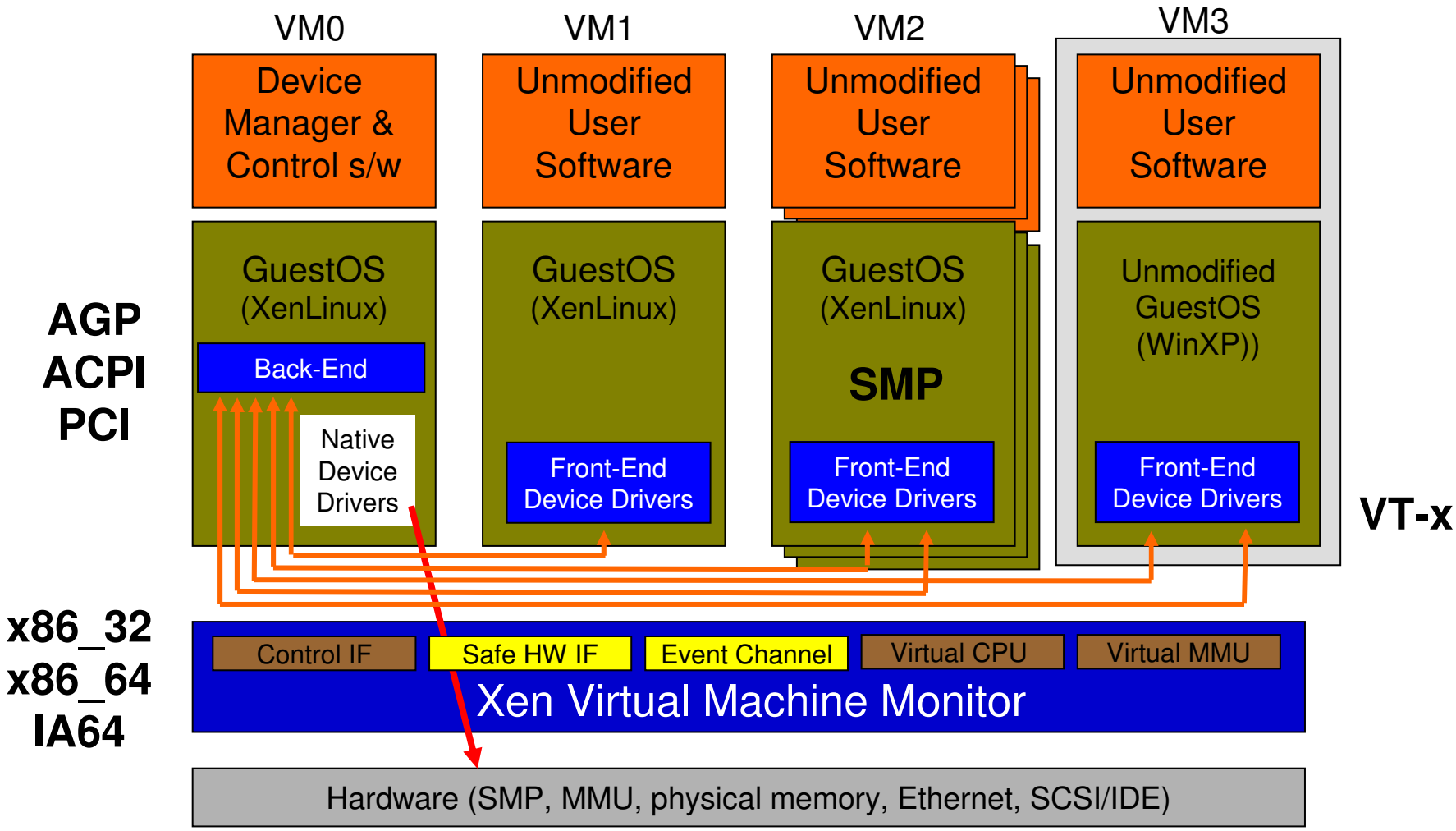
Virtualization Benefits

- Separating the OS from the hardware
  - Users no longer forced to upgrade OS to run on latest hardware
- Device support is part of the platform
  - Write one device driver rather than N
  - Better for system reliability/availability
  - Faster to get new hardware deployed
- Enables “Virtual Appliances”
  - Applications encapsulated with their OS
  - Easy configuration and management

Xen 3.0 Highlights

- x86, x86_64, ia64 and initial Power support
- Leading performance
- Secure isolation and QoS control
- SMP guest OSes
- Hotplug CPUs, memory and devices
- Guest save/restore and live relocation
- VT/AMDV support: "HVM"
  - Run unmodified guest kernels
  - Support for progressive paravirtualization

Xen 3.0 Architecture



Para-Virtualization in Xen

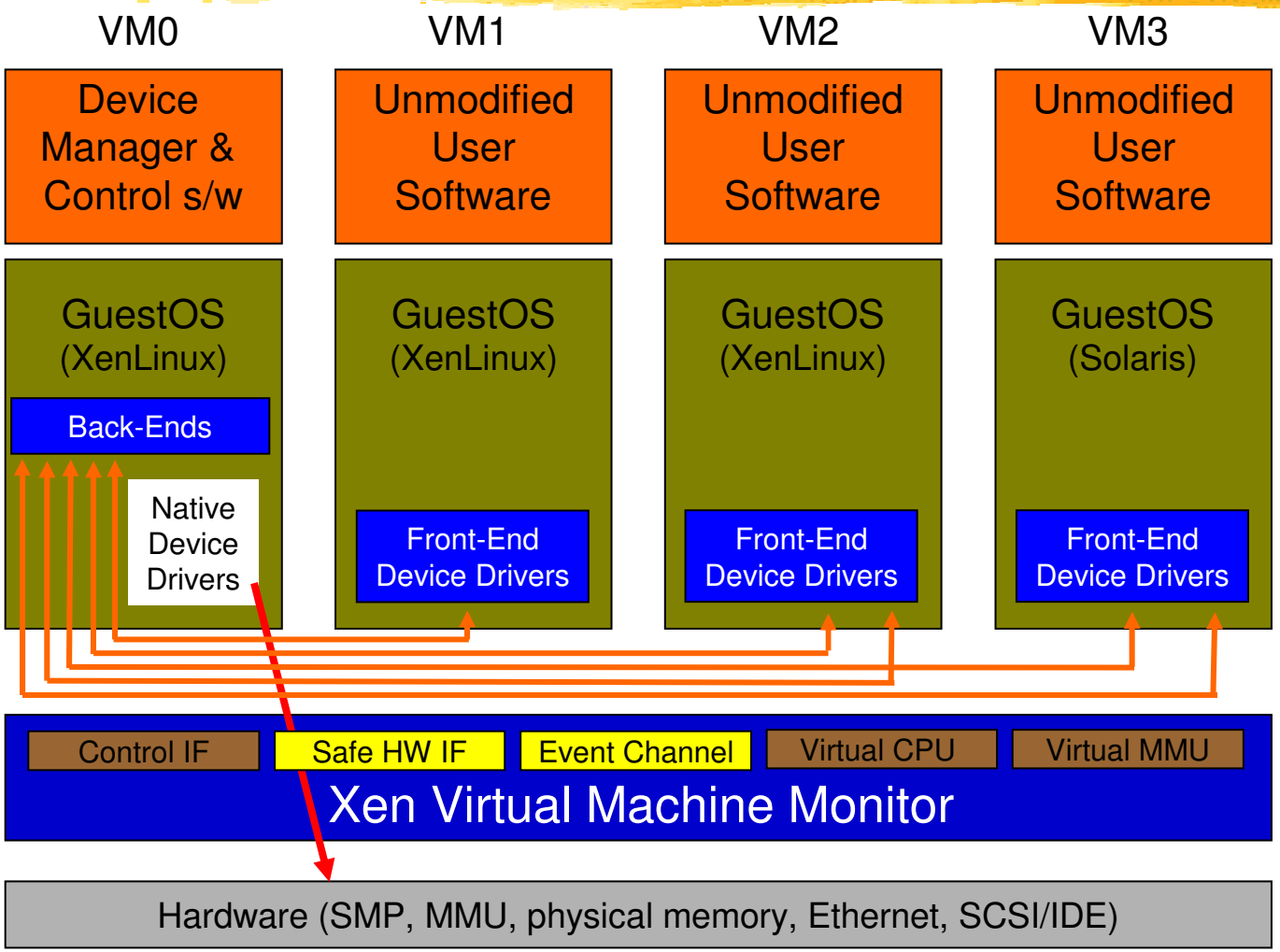
- Xen extensions to x86 arch
  - Like x86, but Xen invoked for privileged ops
  - Avoids binary rewriting
  - Minimize number of privilege transitions into Xen
  - Modifications relatively simple and self-contained
- Modify kernel to understand virtualised env.
  - Wall-clock time vs. virtual processor time
    - Desire both types of alarm timer
  - Expose real resource availability
    - Enables OS to optimise its own behaviour

Xen 3 API support

- Linux 2.6.16/17/18 and -rc/-tip
- 2.6.5 2.6.9.EL 2.4.21.EL
- Available in distros: FC4, FC5, FC6, SuSELinux10, SLES10, Ubuntu, Gentoo,…
- NetBSD 3, FreeBSD 7.0, OpenSolaris 10, Plan9, minix, …

- Linux upstream submission process agreed at Kernel Summit

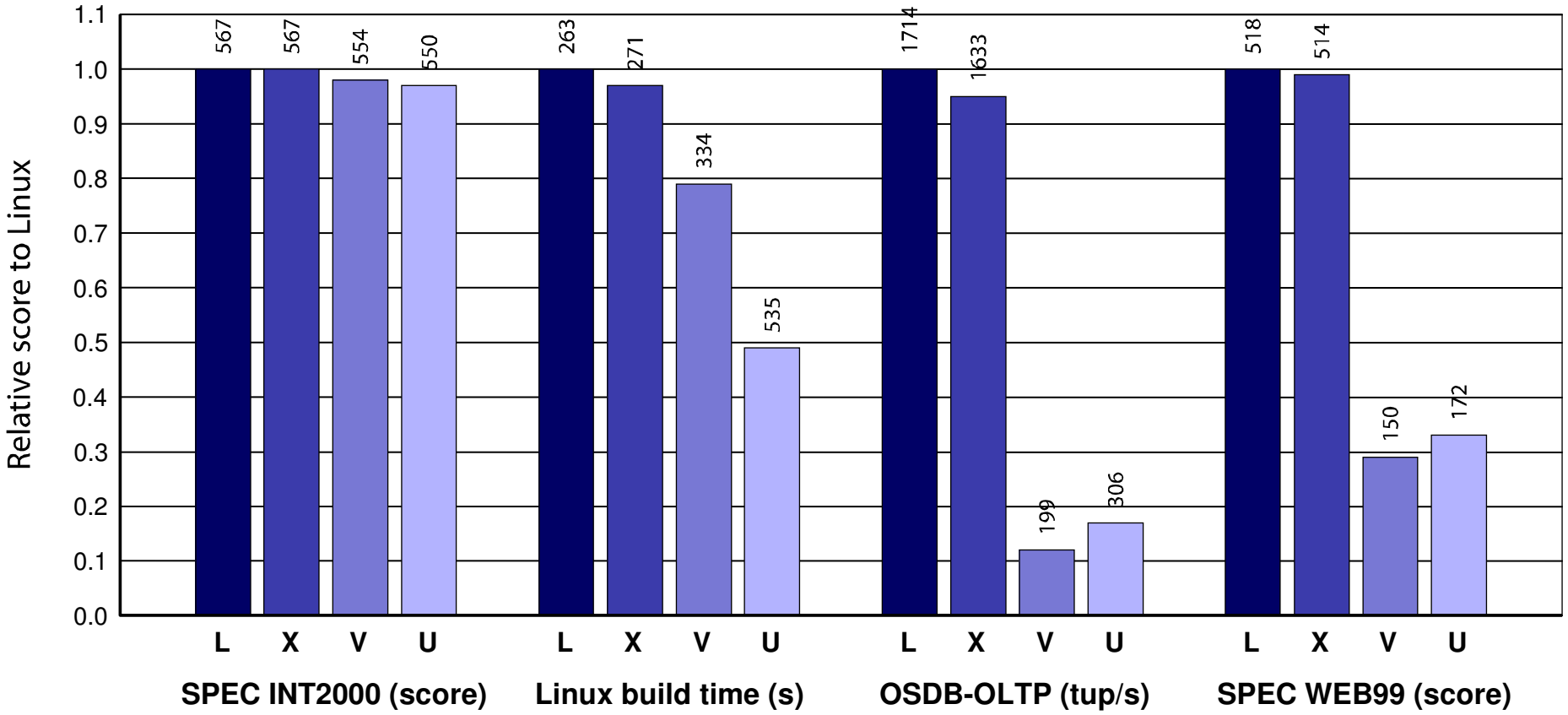
Xen 2.0 Architecture



I/O Architecture

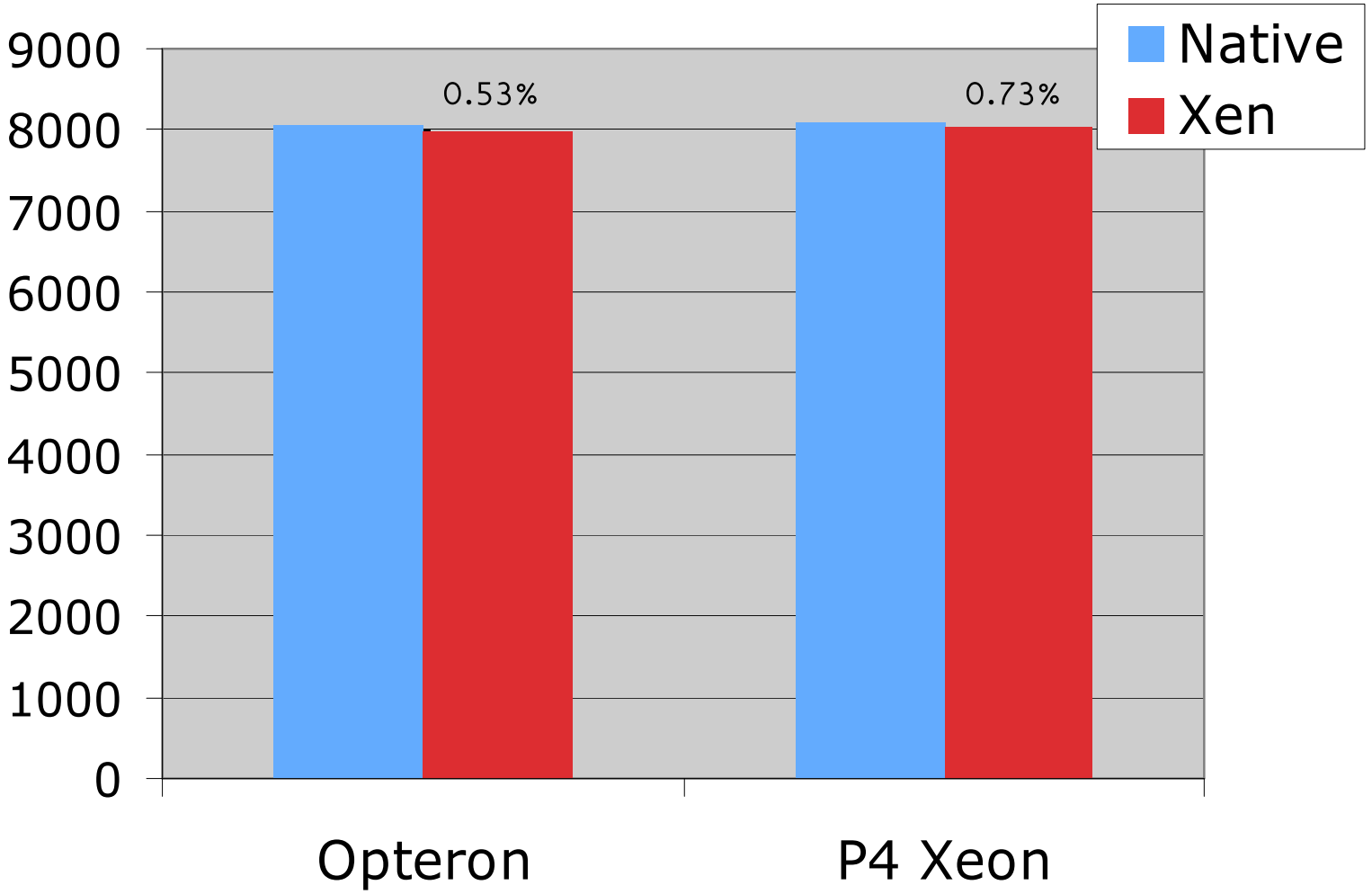
- Xen *IO-Spaces* delegate guest OSes protected access to specified h/w devices
  - Virtual PCI configuration space
  - Virtual interrupts
  - (Need IOMMU for full DMA protection)
- Devices are virtualised and exported to other VMs via *Device Channels*
  - Safe asynchronous shared memory transport
  - 'Backend' drivers export to 'frontend' drivers
  - Net: use normal bridging, routing, iptables
  - Block: export any blk dev e.g. sda4,loop0,vg3
- (Infiniband / "Smart NICs" for direct guest IO)

System Performance

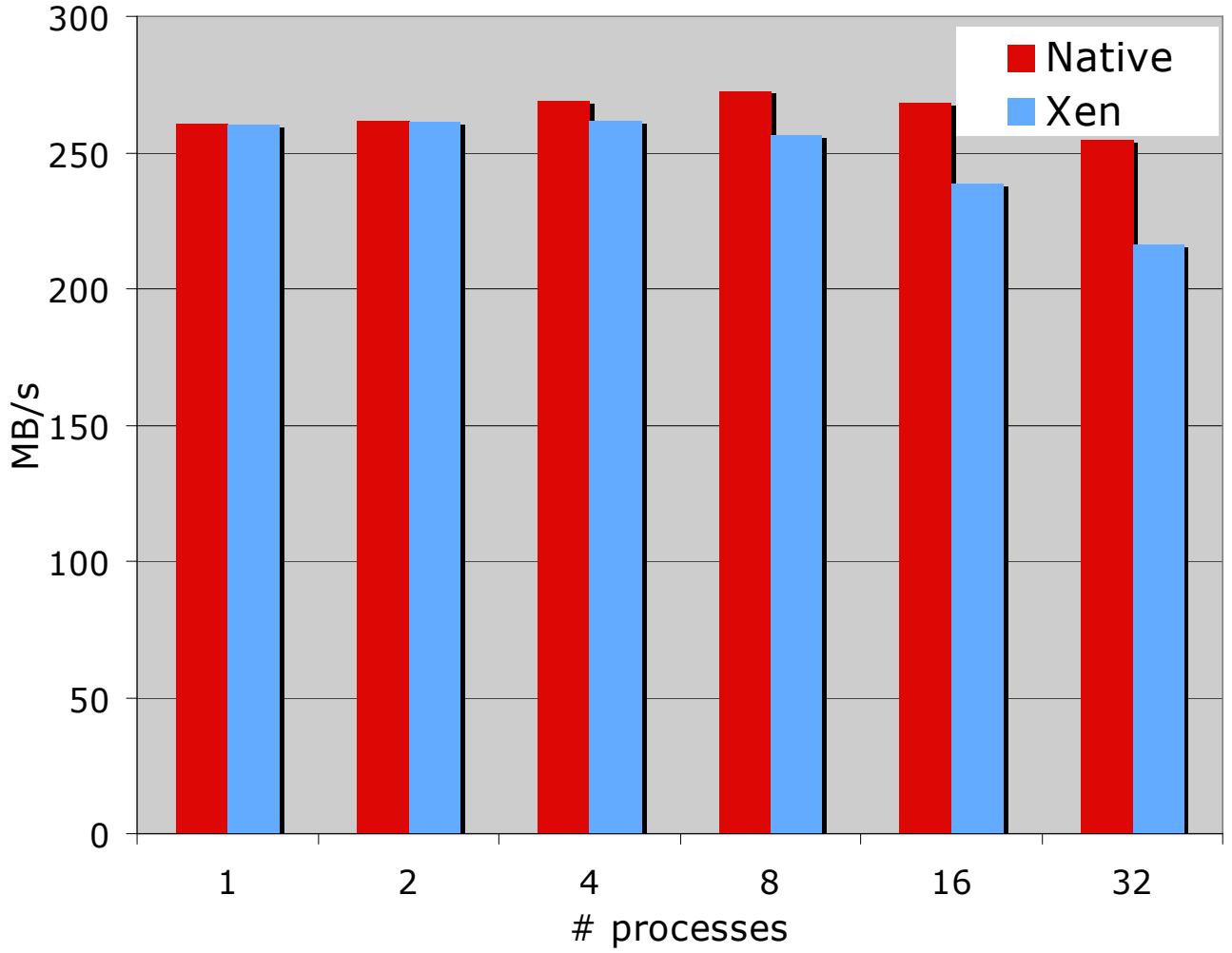


Benchmark suite running on Linux (L), Xen (X), VMware Workstation (V), and UML (U)

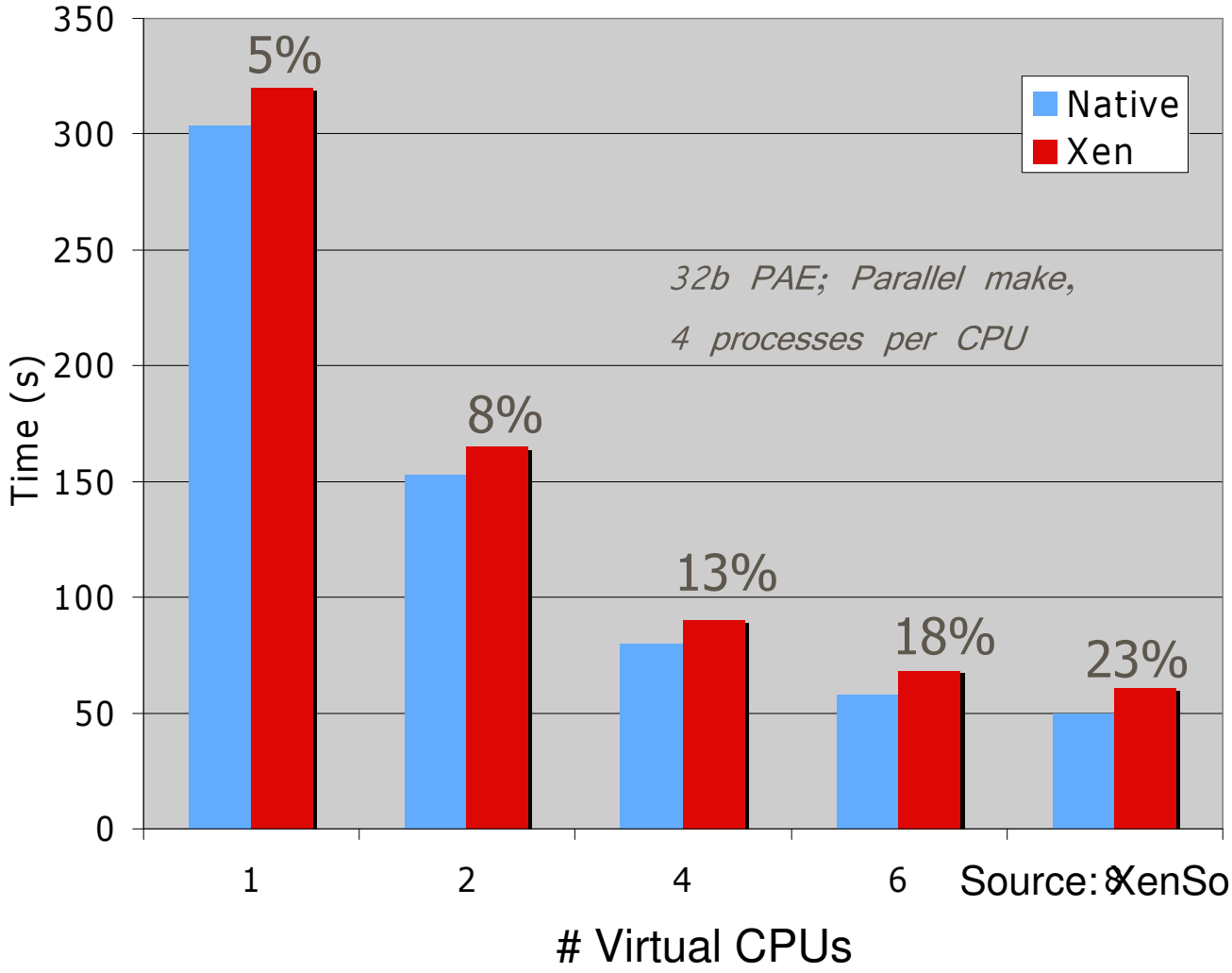
specjbb2005



dbench

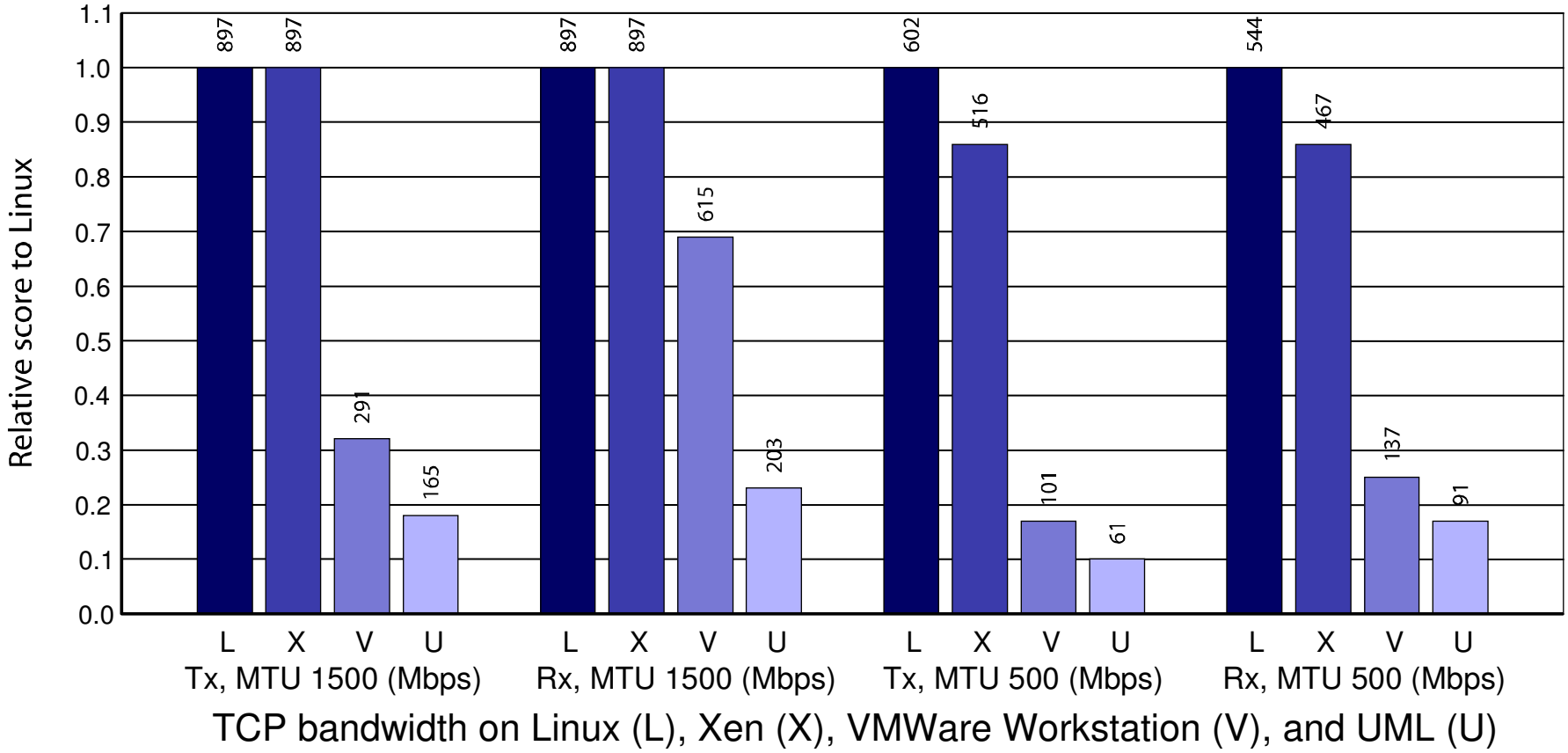


Kernel build



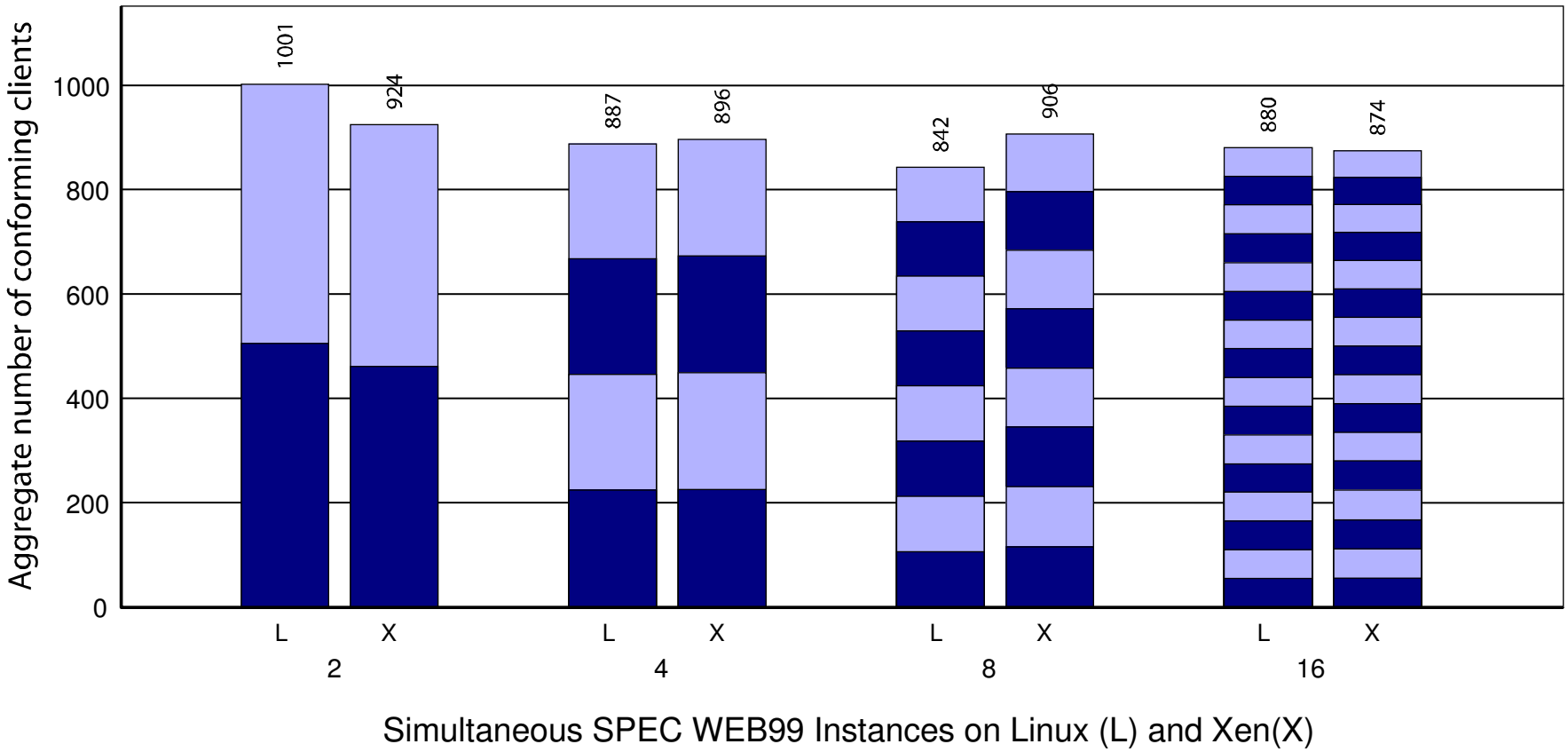
Source: XenSource, Inc: 10/06

TCP results



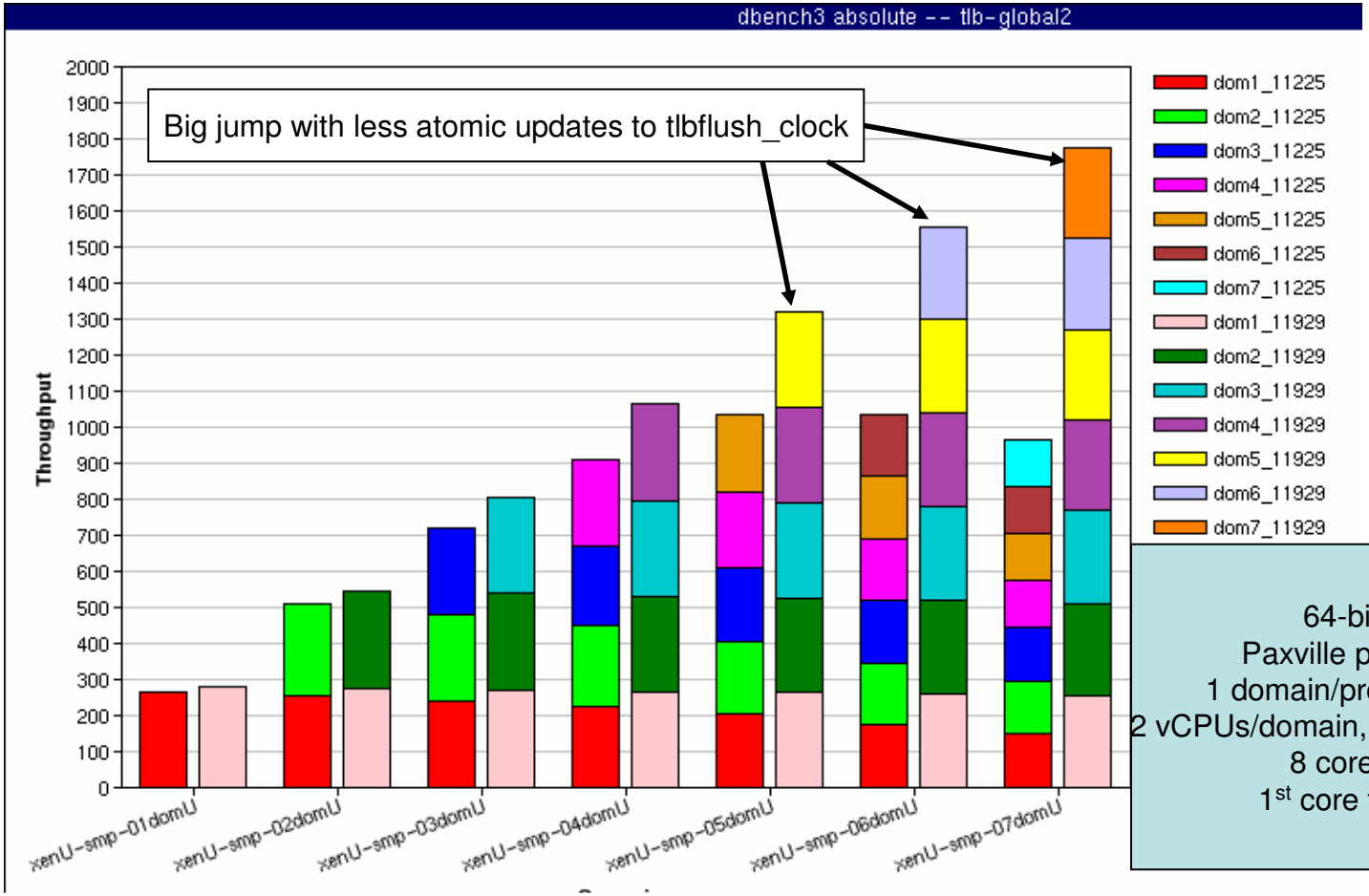
TCP bandwidth on Linux (L), Xen (X), VMWare Workstation (V), and UML (U)

Scalability



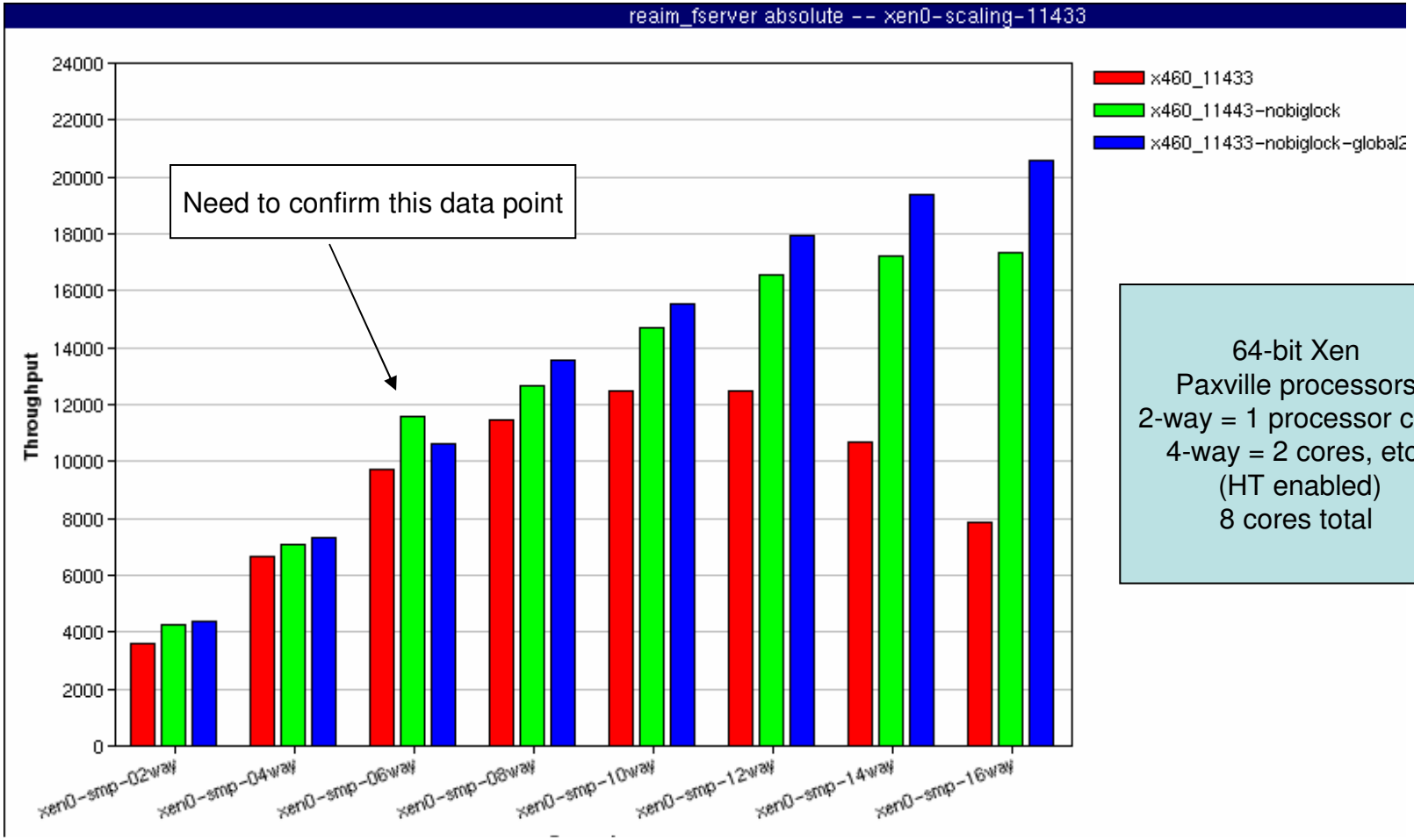
Scalability

- Scaling many domains
  - One domain per processor core, 1 to many cores/domains
  - Presented data showing problems at OLS on 64-bit domains
    - Since then, tested 32-bit domains, which look fine
    - 64-bit did not scale because of excessive TLB flushes, which put too much pressure on global update of tlbflush_clock in Xen. Jun has fixed with TLB global bit patch –no need to flush all TLB entries for syscalls, etc, which also does not update tlbflush_clock. Patch also improves single domain performance by keeping user TLB entries.



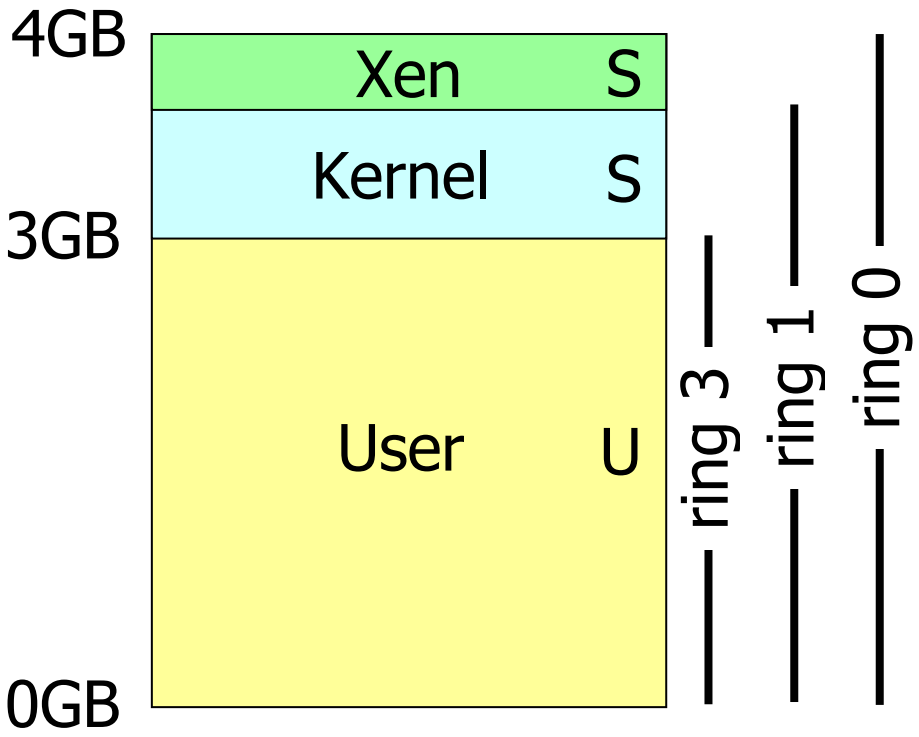
Scalability

- Scaling a large domain, 1 to many vCPUs
  - Elimination of writable page-tables in xen-unstable helps a lot (red bars below) but still have significant scaling problems beyond 6-way
  - Now need fine grain locking in mm functions; should be easier to implement with writable page-tables gone. (green bars have domain big-lock removed)
  - TLB global bit optimization also helps scalability (blue bars)



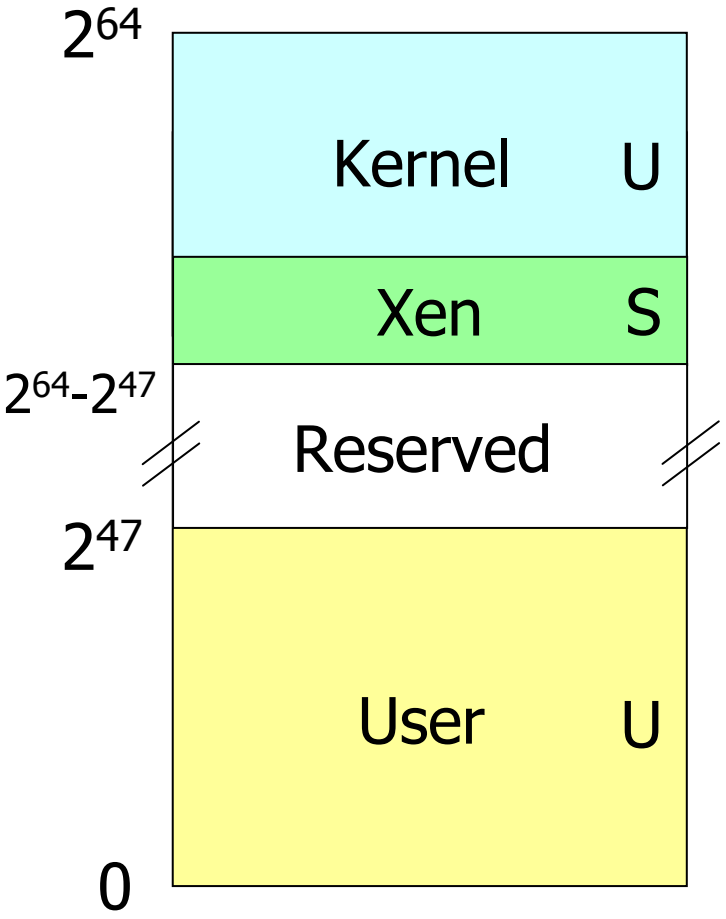
64-bit Xen
Paxville processors
2-way = 1 processor core
4-way = 2 cores, etc,
(HT enabled)
8 cores total

x86_32



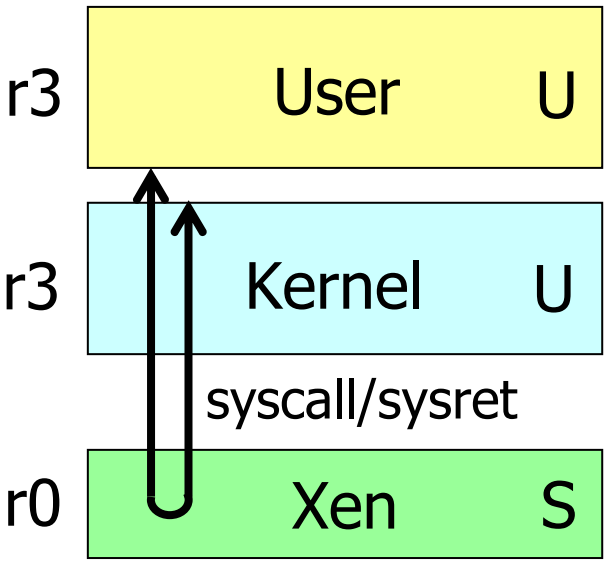
- Xen reserves top of VA space
- Segmentation protects Xen from kernel
- System call speed unchanged
- Xen 3 now supports PAE for >4GB mem

x86_64



- Large VA space makes life a lot easier, but:
- No segment limit support
- → Need to use page-level protection to protect hypervisor

x86_64



- Run user-space and kernel in ring 3 using different pagetables
  - Two PGD's (PML4's): one with user entries; one with user plus kernel entries
- System calls require an additional syscall/ret via Xen
- Per-CPU trampoline to avoid needing GS in Xen

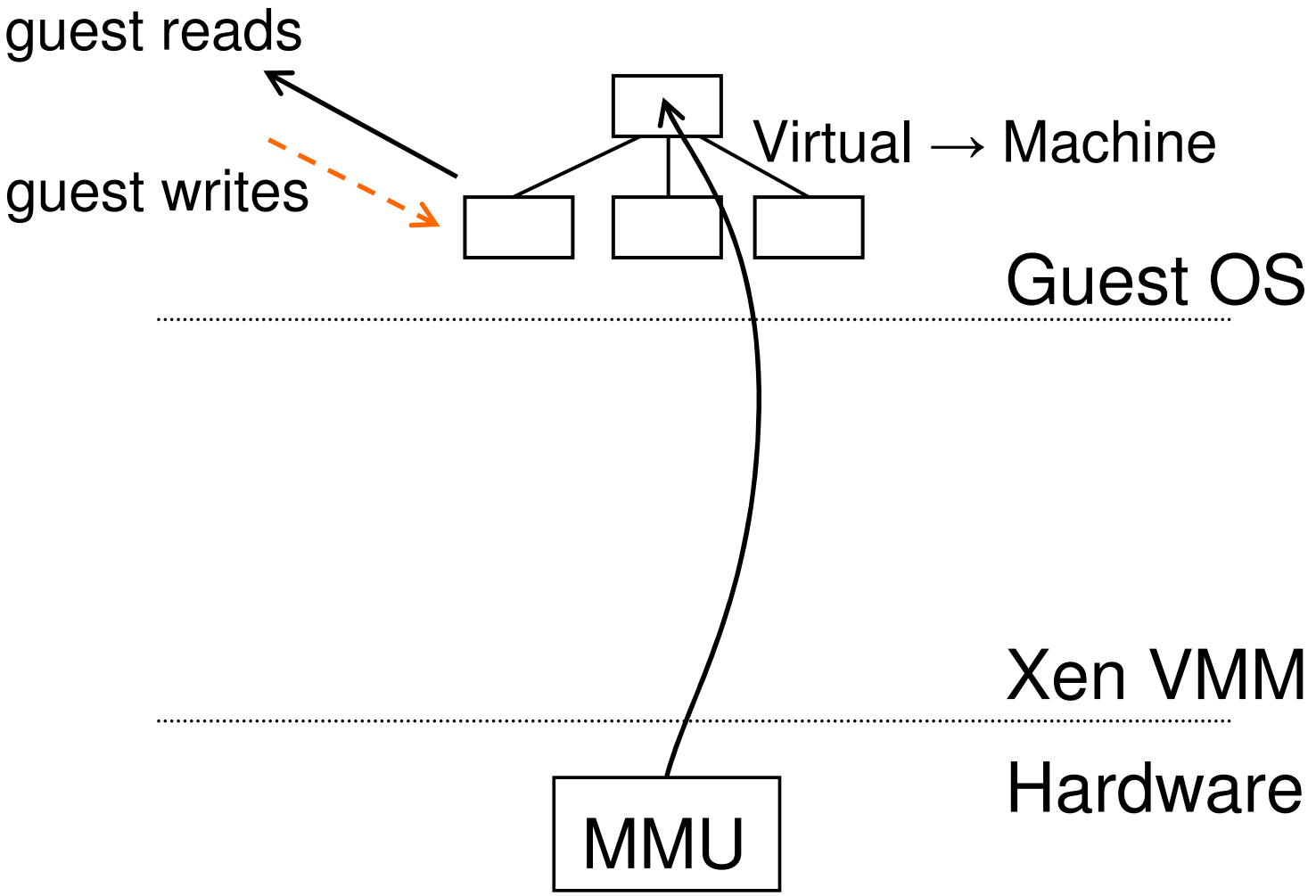
x86 CPU virtualization

- 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 user-space to guest OS system calls
- MMU virtualisation: shadow vs. direct-mode

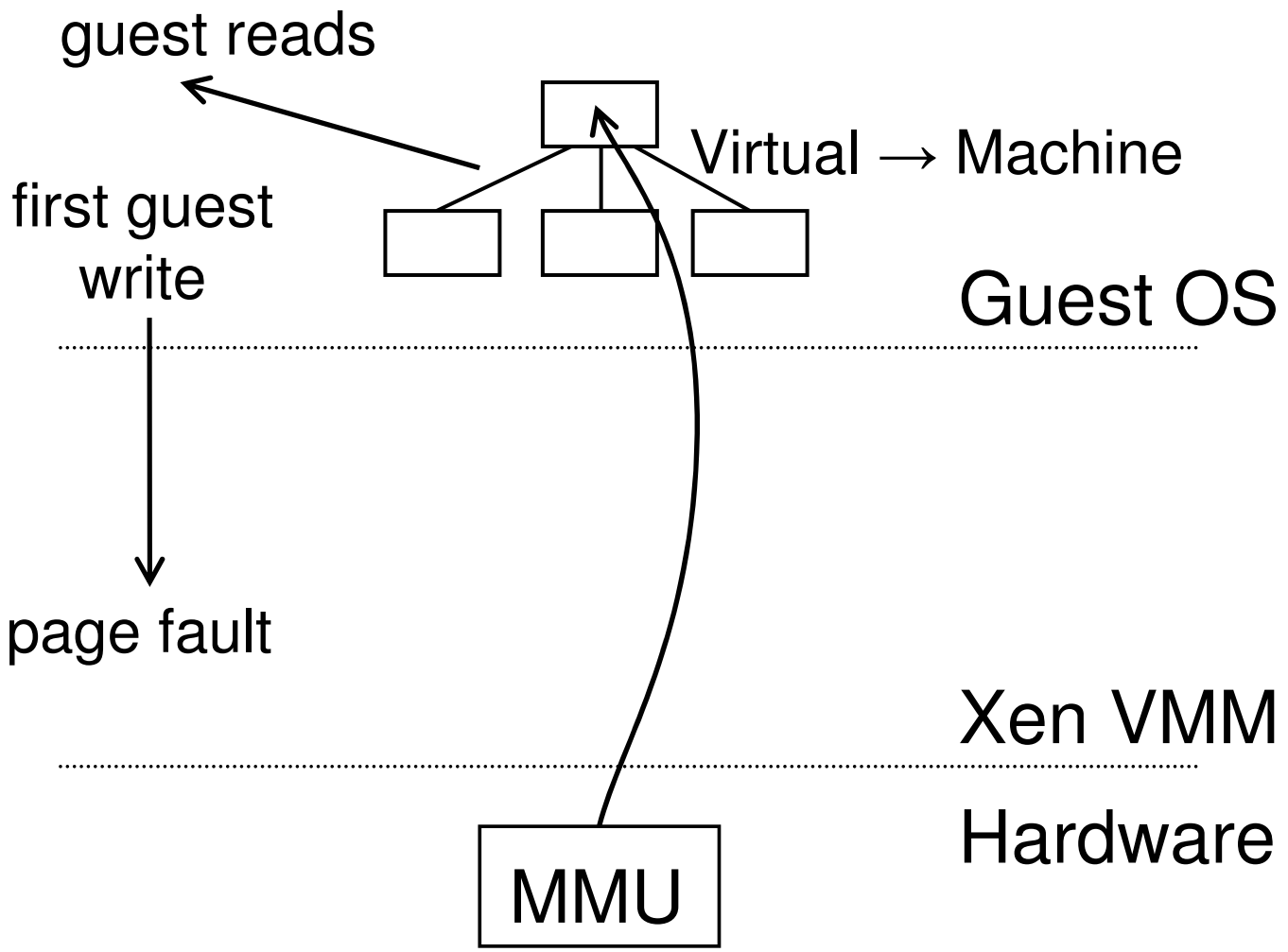
Para-Virtualizing the MMU

- Guest OSes allocate and manage own PTs
  - Hypercall to change PT base
- Xen must validate PT updates before use
  - Allows incremental updates, avoids revalidation
- Validation rules applied to each PTE:
  1. Guest may only map pages it owns*
  2. Pagetable pages may only be mapped RO
- Xen traps PTE updates and emulates, or 'unhooks' PTE page for bulk updates

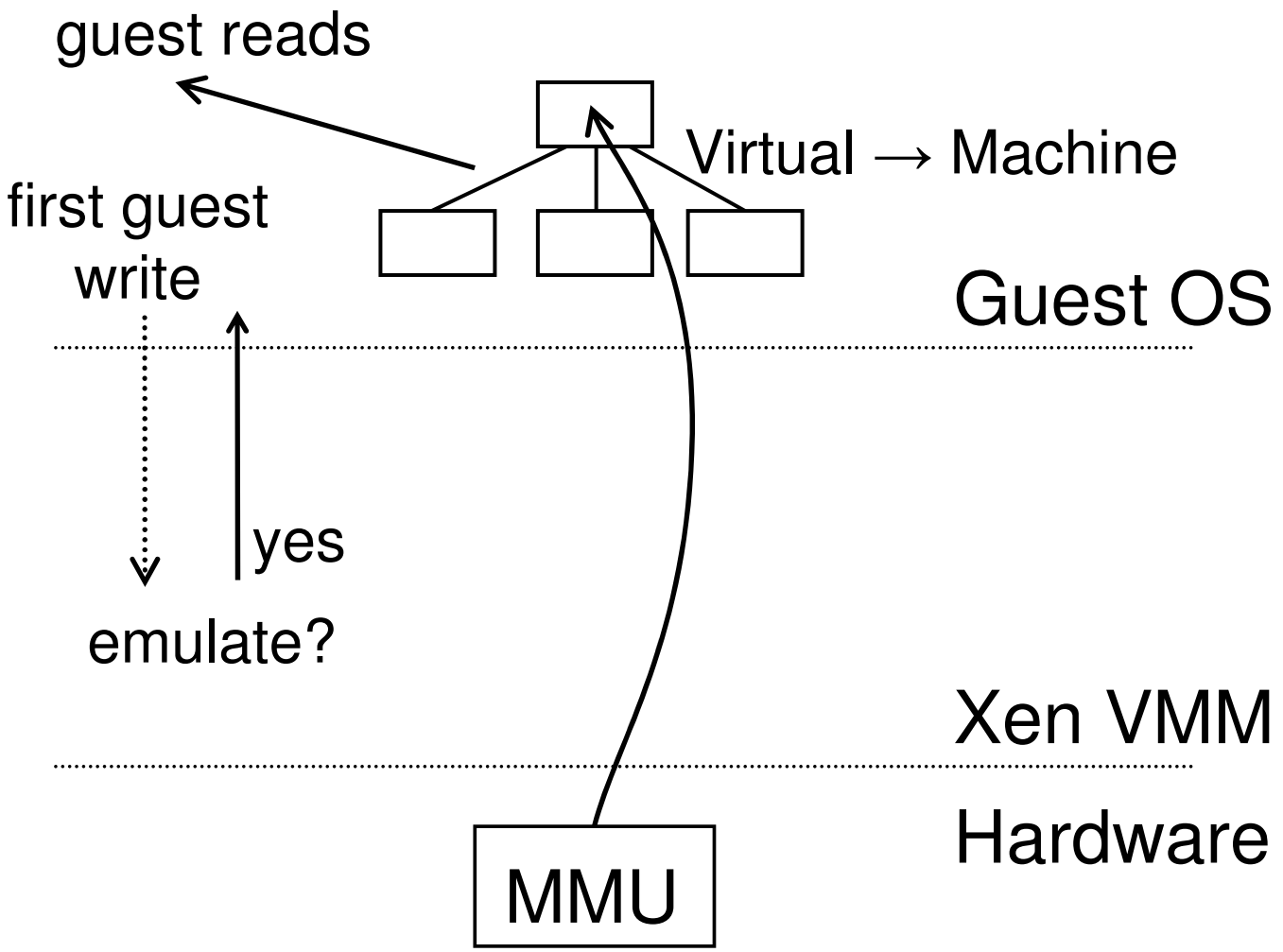
MMU Virtualization : Direct-Mode



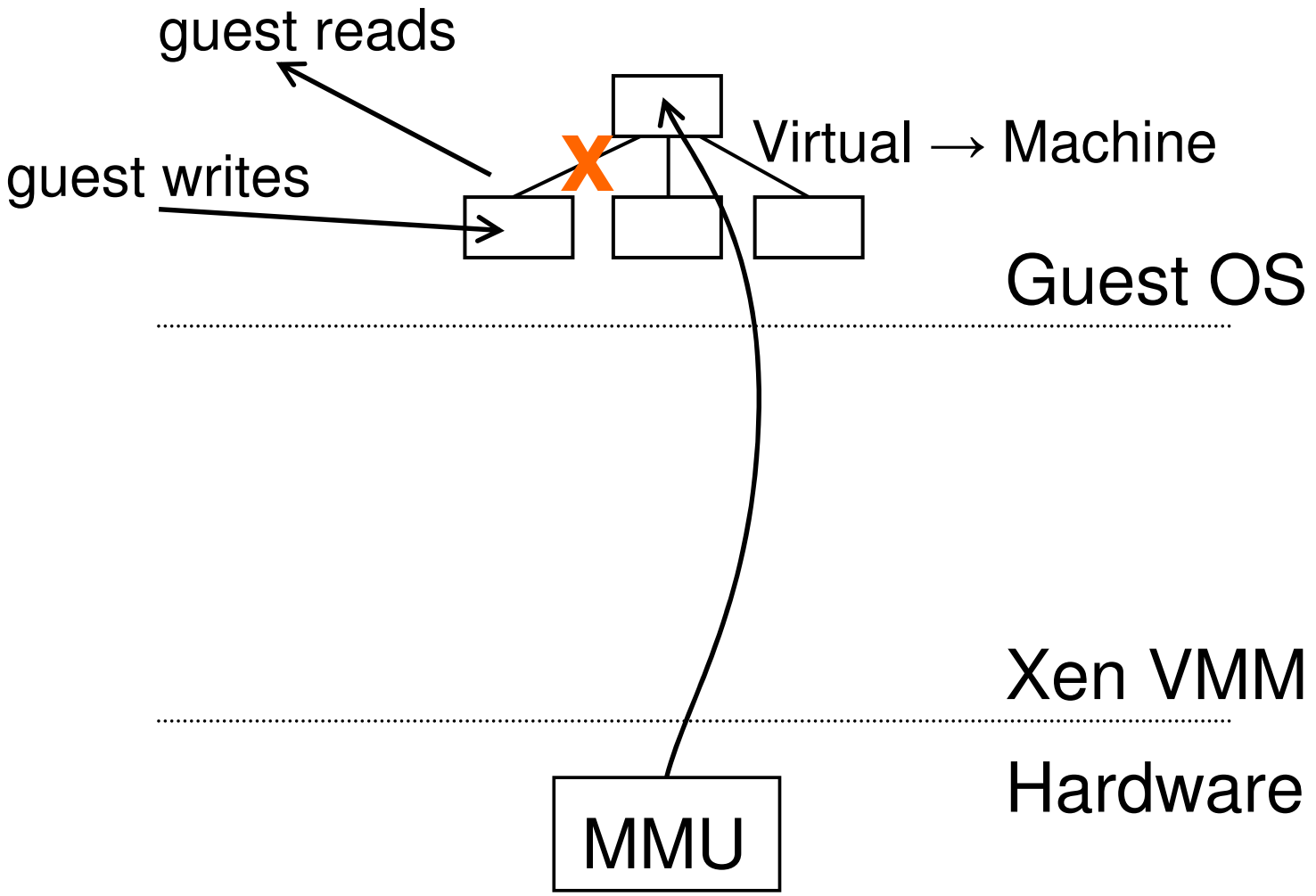
Writeable Page Tables : 1 – Write fault



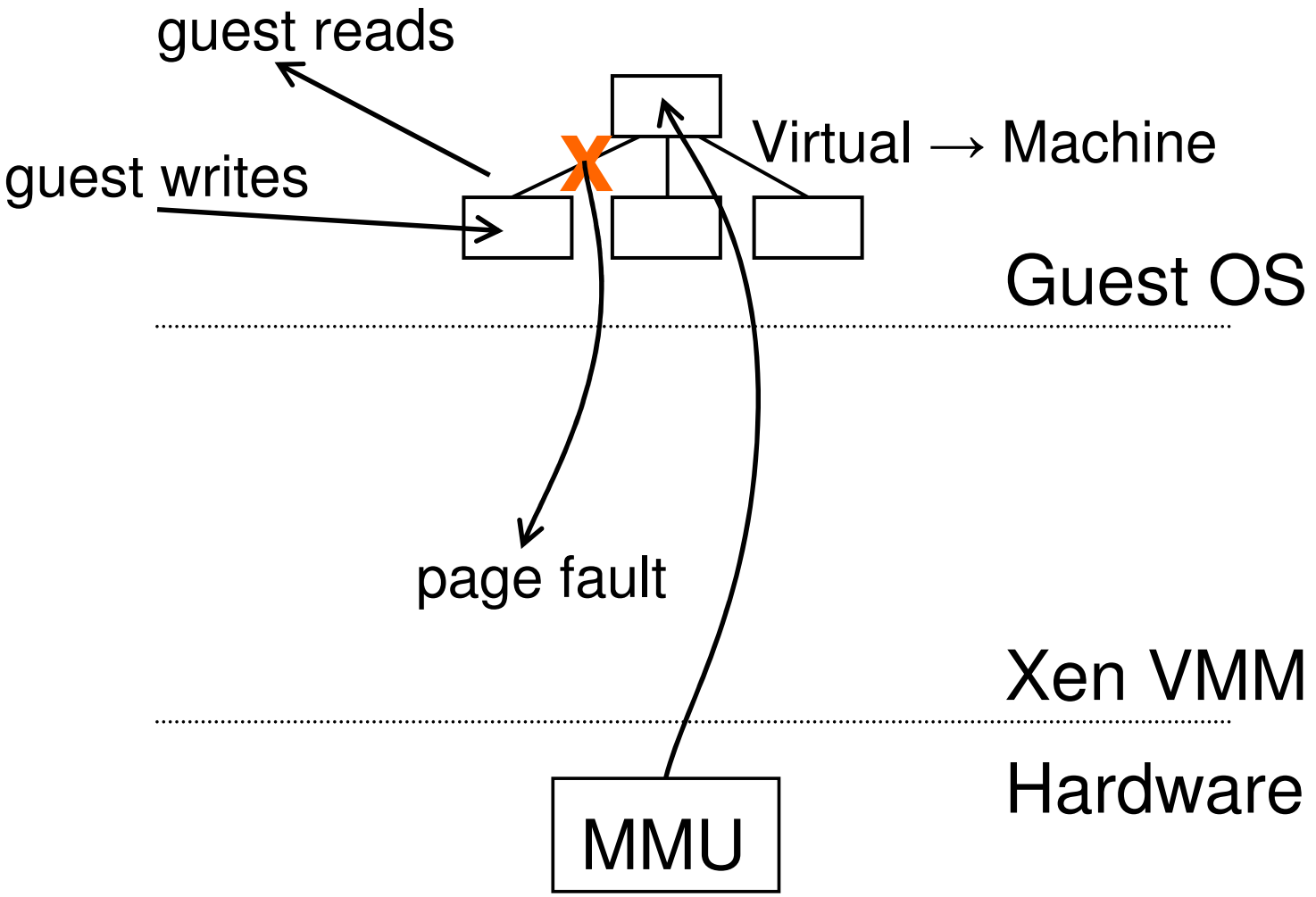
Writeable Page Tables : 2 – Emulate?



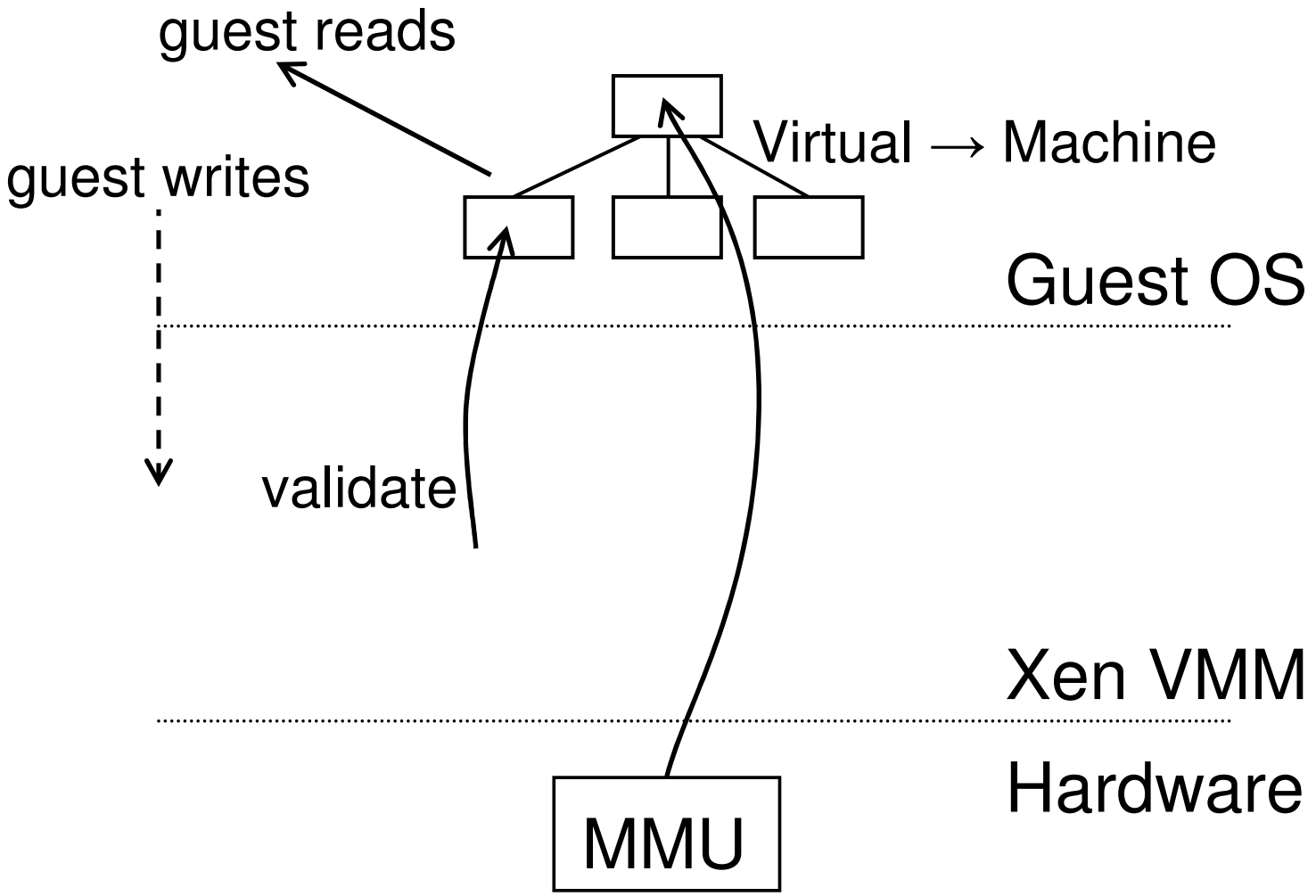
Writeable Page Tables : 3 - Unhook



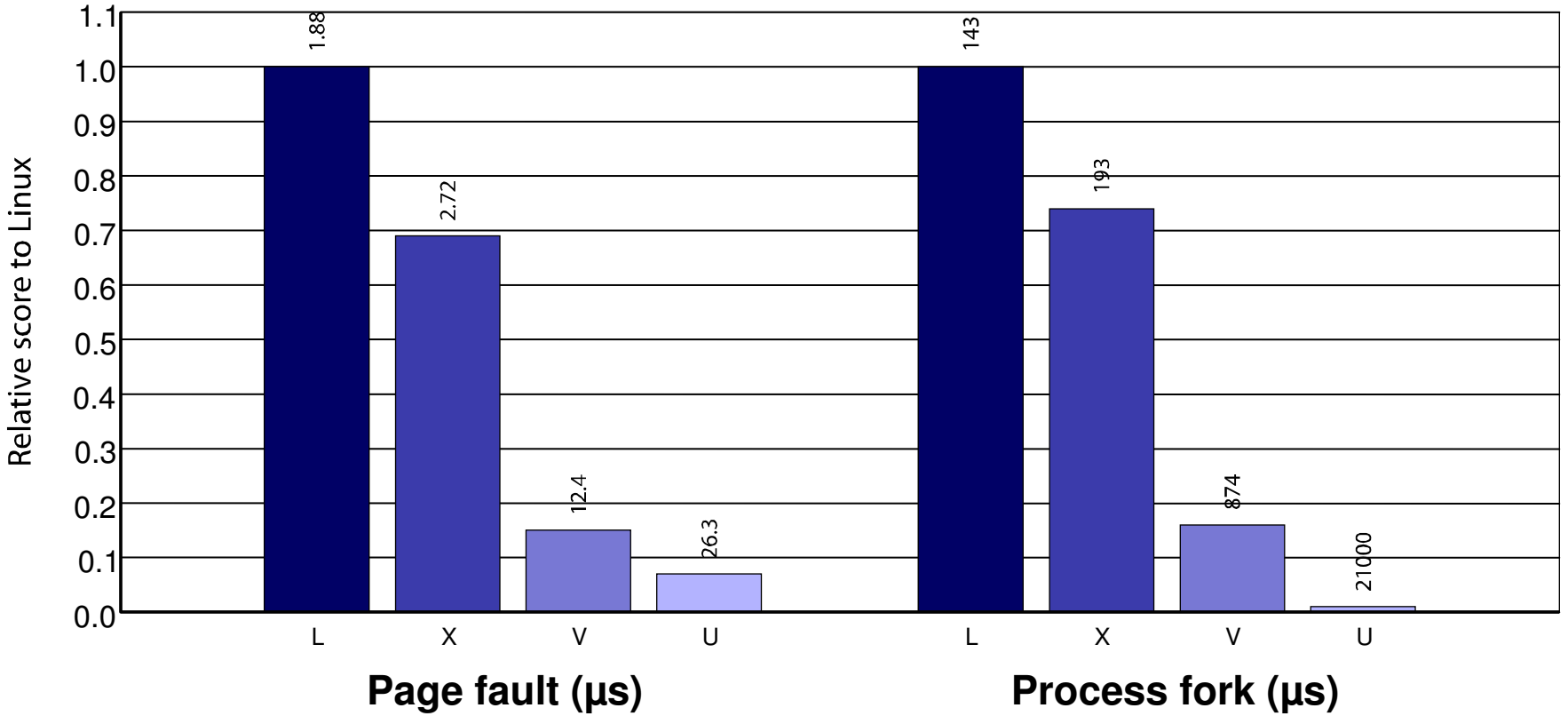
Writeable Page Tables : 4 - First Use



Writeable Page Tables : 5 – Re-hook



MMU Micro-Benchmarks



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

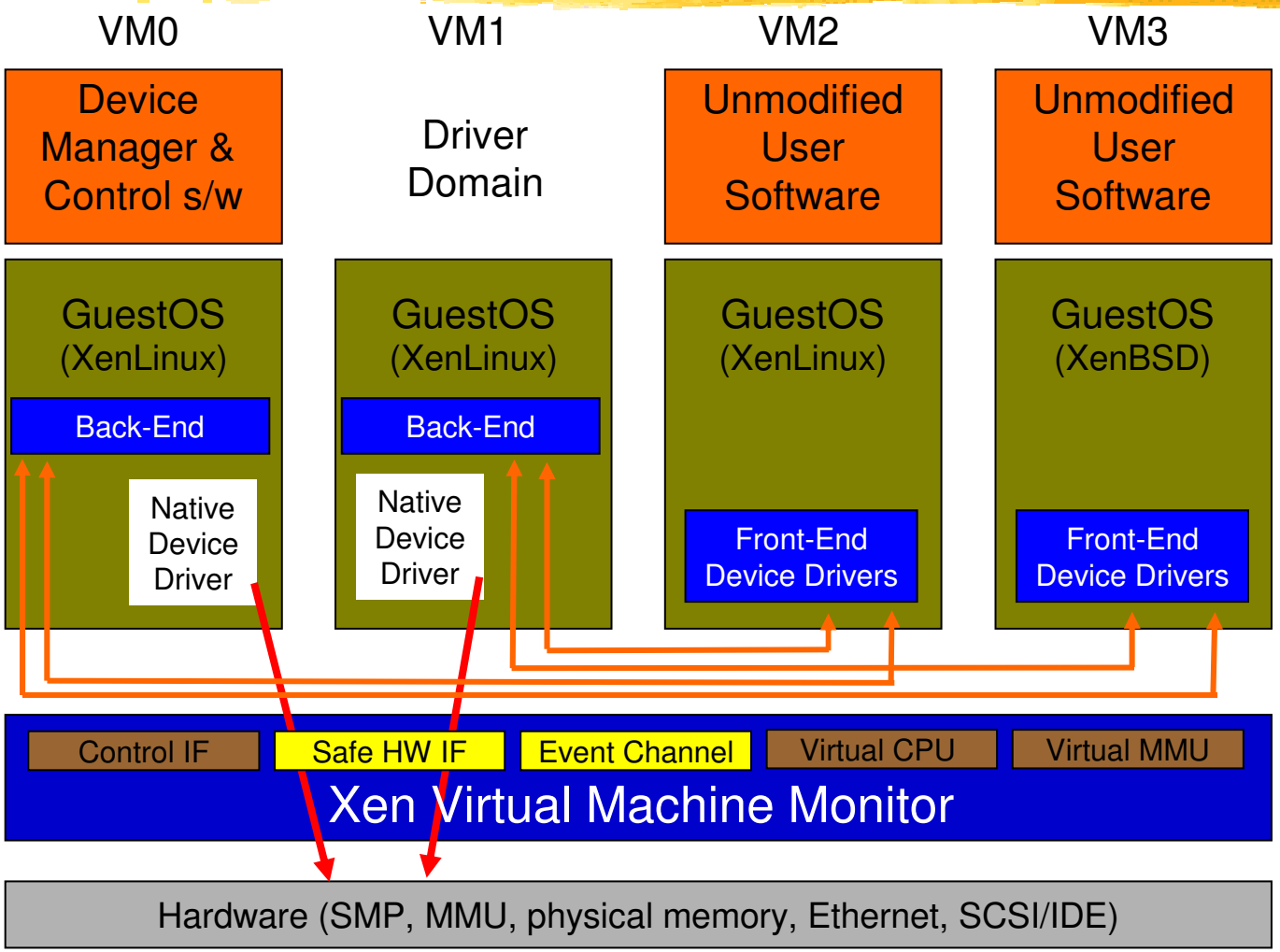
SMP Guest Kernels

- Xen support multiple VCPUs per guest
  - Virtual IPI's sent via Xen event channels
  - Currently up to 32 VCPUs supported
- Simple hotplug/unplug of VCPUs
  - From within VM or via control tools
  - Optimize one active VCPU case by binary patching spinlocks
- NB: Many applications exhibit poor SMP scalability – often better off running multiple instances each in their own OS

SMP Guest Kernels

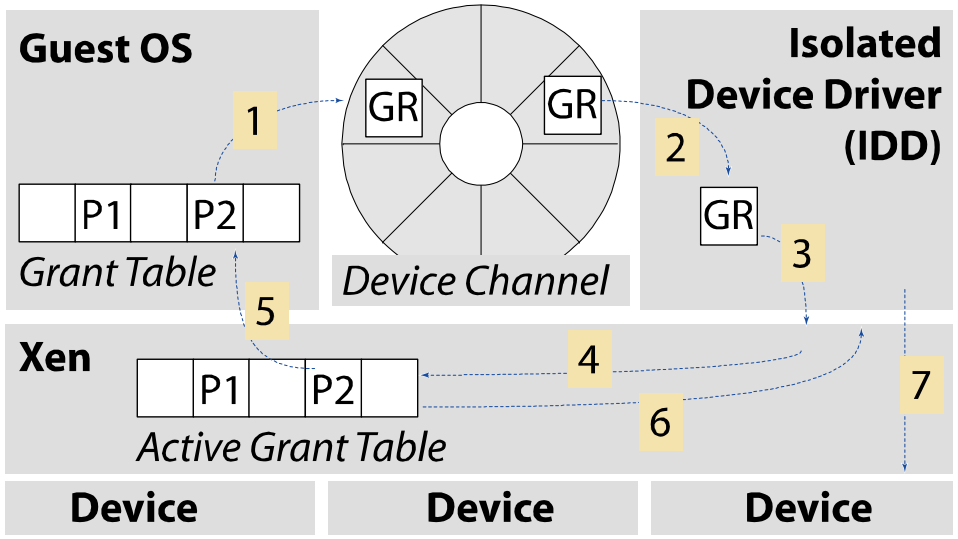
- Takes great care to get good SMP performance while remaining secure
  - Requires extra TLB syncronization IPIs
- SMP scheduling is a tricky problem
  - Wish to run all VCPUs at the same time
  - But, strict gang scheduling is not work conserving
  - Opportunity for a hybrid approach
- Paravirtualized approach enables several important benefits
  - Avoids many virtual IPIs
  - Allows ‘bad preemption’ avoidance
  - Auto hot plug/unplug of CPUs

Driver Domains



Device Channel Interface

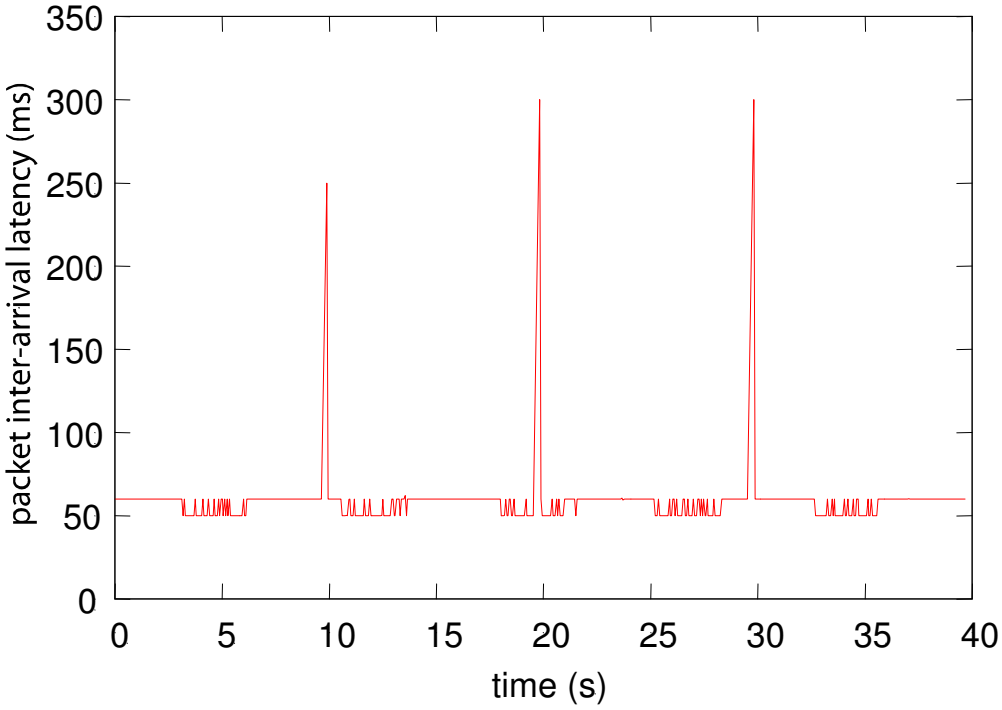
Guest Requests DMA:
1. Grant Reference for Page P2 placed on device channel
2. IDD removes GR
3. Sends pin request to Xen



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

Isolated Driver VMs

- Run device drivers in separate domains
- Detect failure e.g.
  - Illegal access
  - Timeout
- Kill domain, restart
- E.g. 275ms outage from failed Ethernet driver

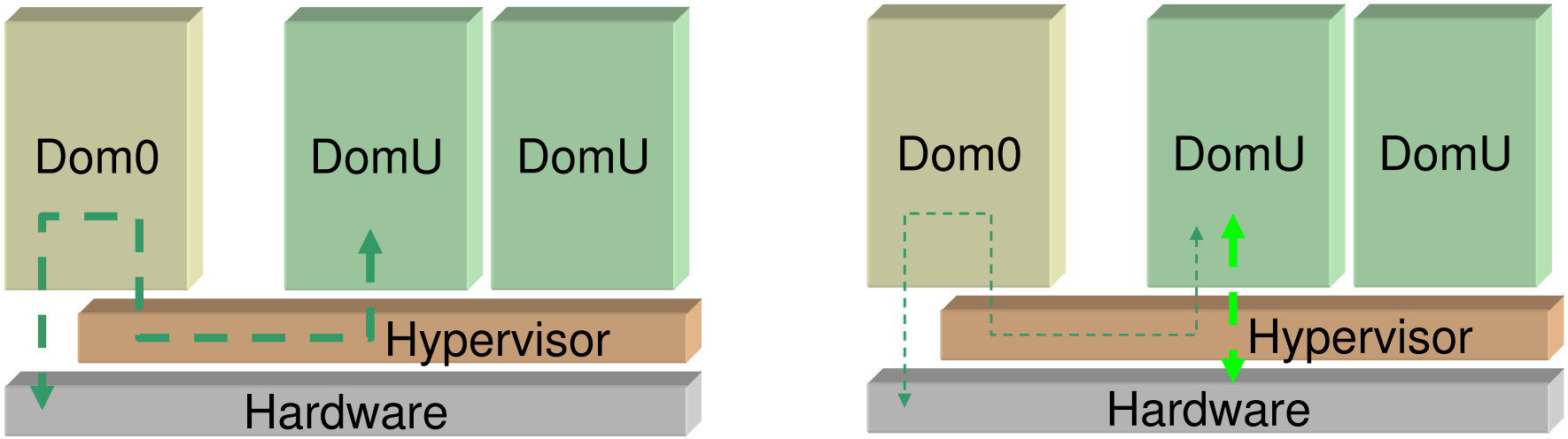


IO Virtualization

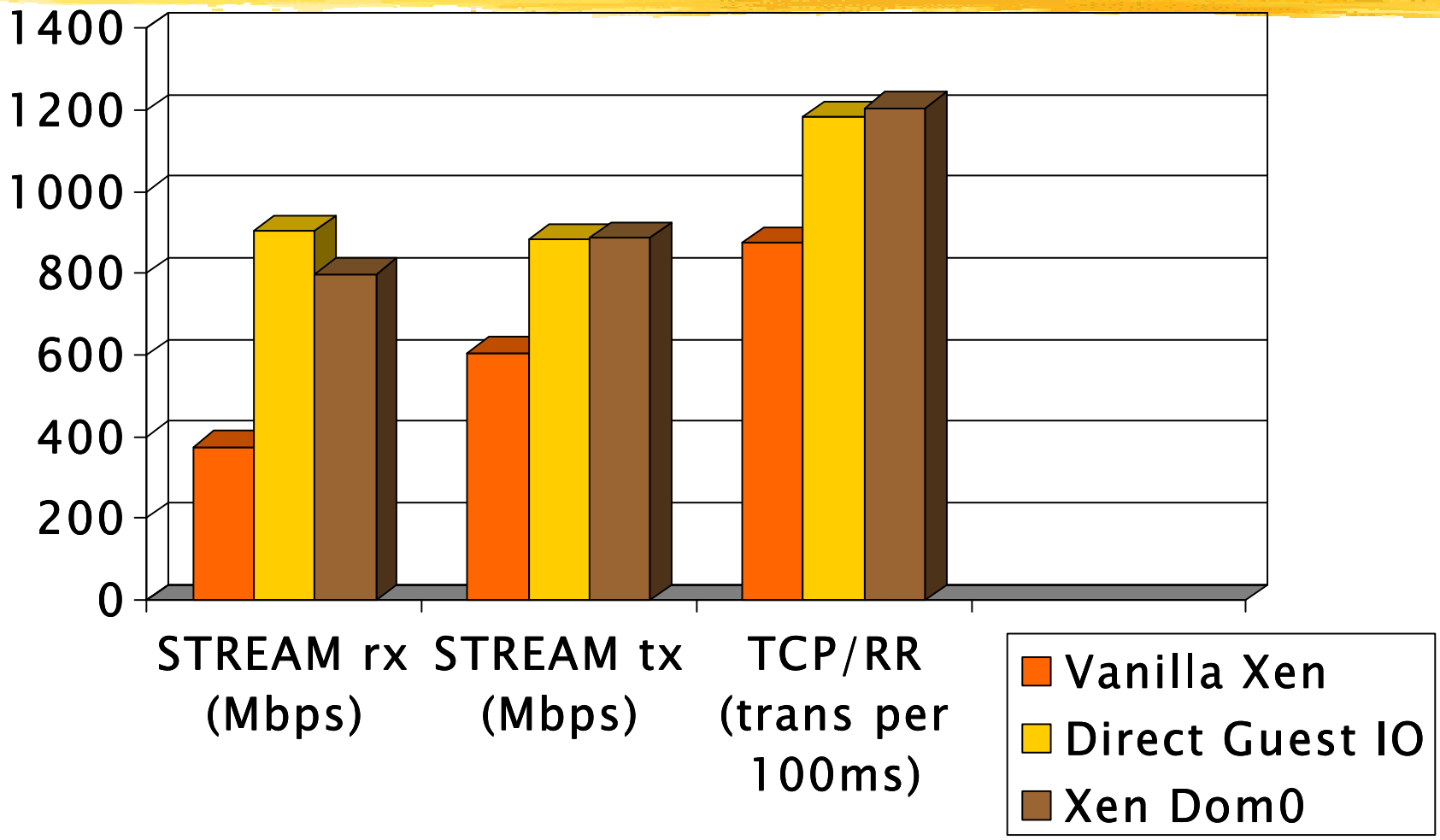
- IO virtualization in s/w incurs overhead
  - Latency vs. overhead tradeoff
    - More of an issue for network than storage
  - Can burn 10-30% more CPU than native
- Direct h/w access from VMs
  - Multiplexing and protection in h/w
  - Xen infiniband support
  - Smart NICs / HCAs

Smart NICs e.g. Solarflare

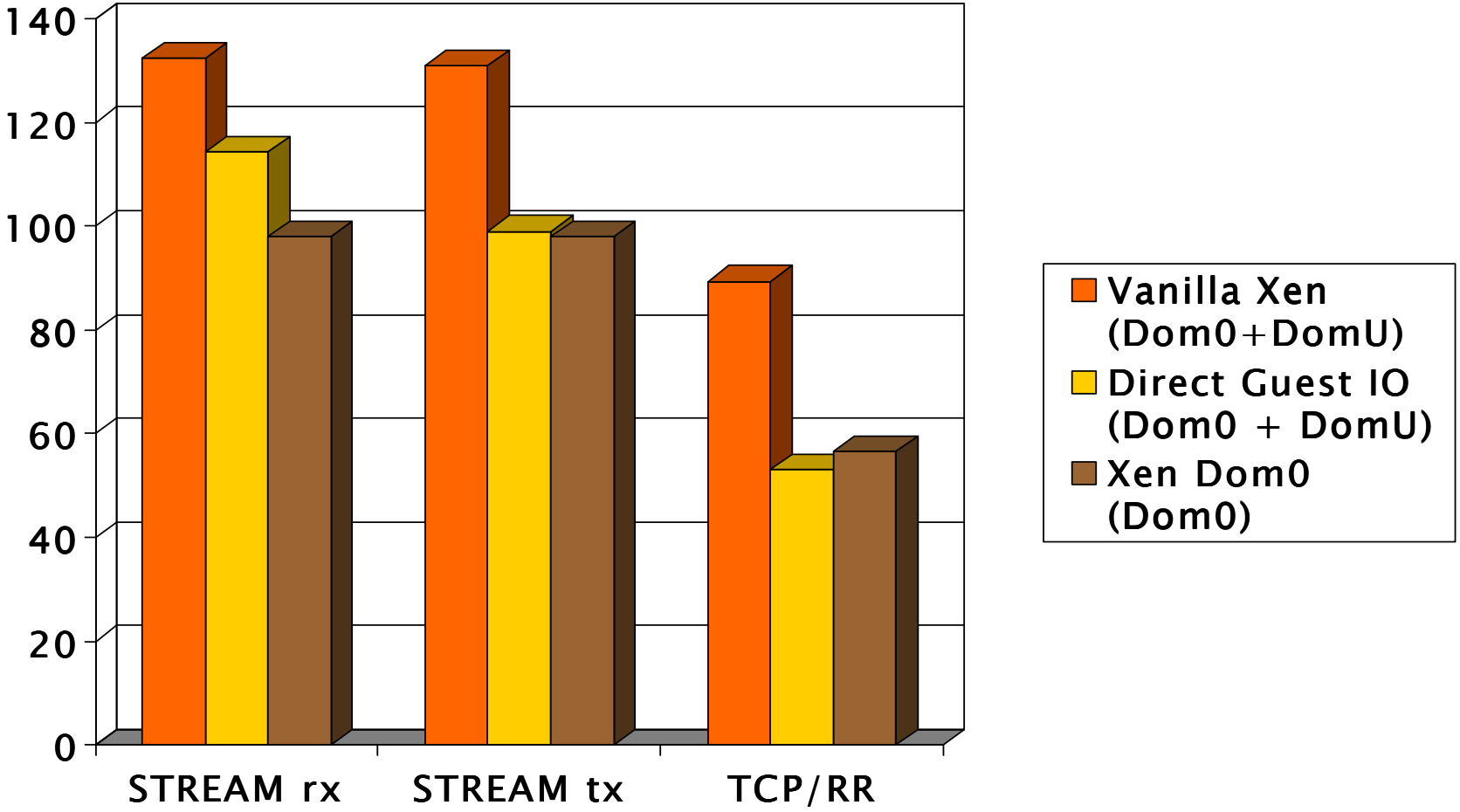
- Accelerated routes set up by Dom0
  - Then DomU can access hardware directly
- NIC has many Virtual Interfaces (VIs)
  - VI = Filter + DMA queue + event queue
- Allow untrusted entities to access the NIC without compromising system integrity



Early results



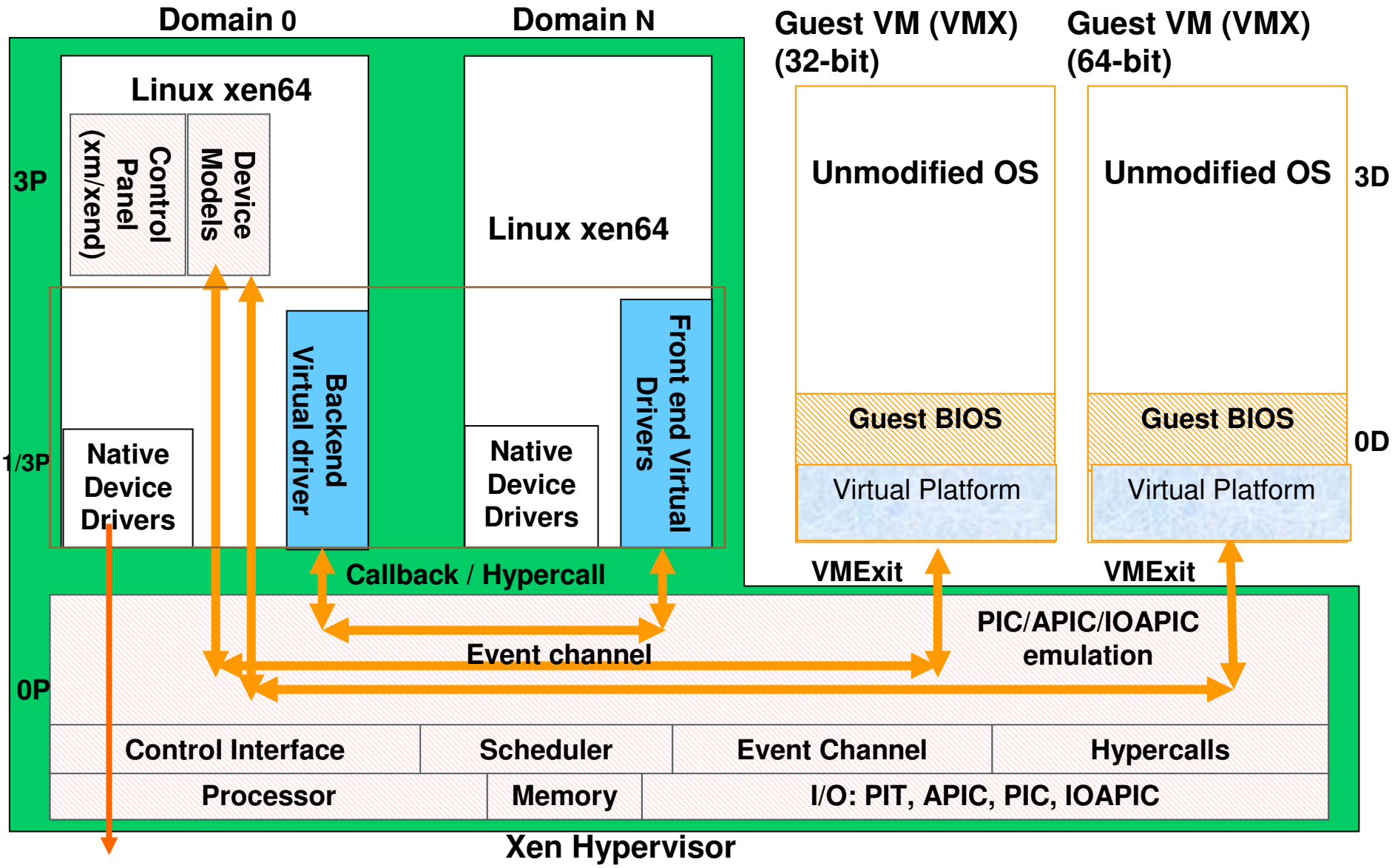
CPU Usage


140
120
100
80
60
40
20
0
STREAM rx
STREAM tx
TCP/RR
Vanilla Xen (Dom0+DomU)
Direct Guest IO (Dom0 + DomU)
Xen Dom0 (Dom0)

VT-x / AMD-V : hvm

- Enable Guest OSes to be run without modification
  - E.g. legacy Linux, Windows XP/2003
- CPU provides vmexits for certain privileged instrs
- Shadow page tables used to virtualize MMU
- Xen provides simple platform emulation
  - BIOS, apic, iopaic, rtc, Net (pcnet32), IDE emulation
- Install paravirtualized drivers after booting for high-performance IO
- Possibility for CPU and memory paravirtualization
  - Non-invasive hypervisor hints from OS

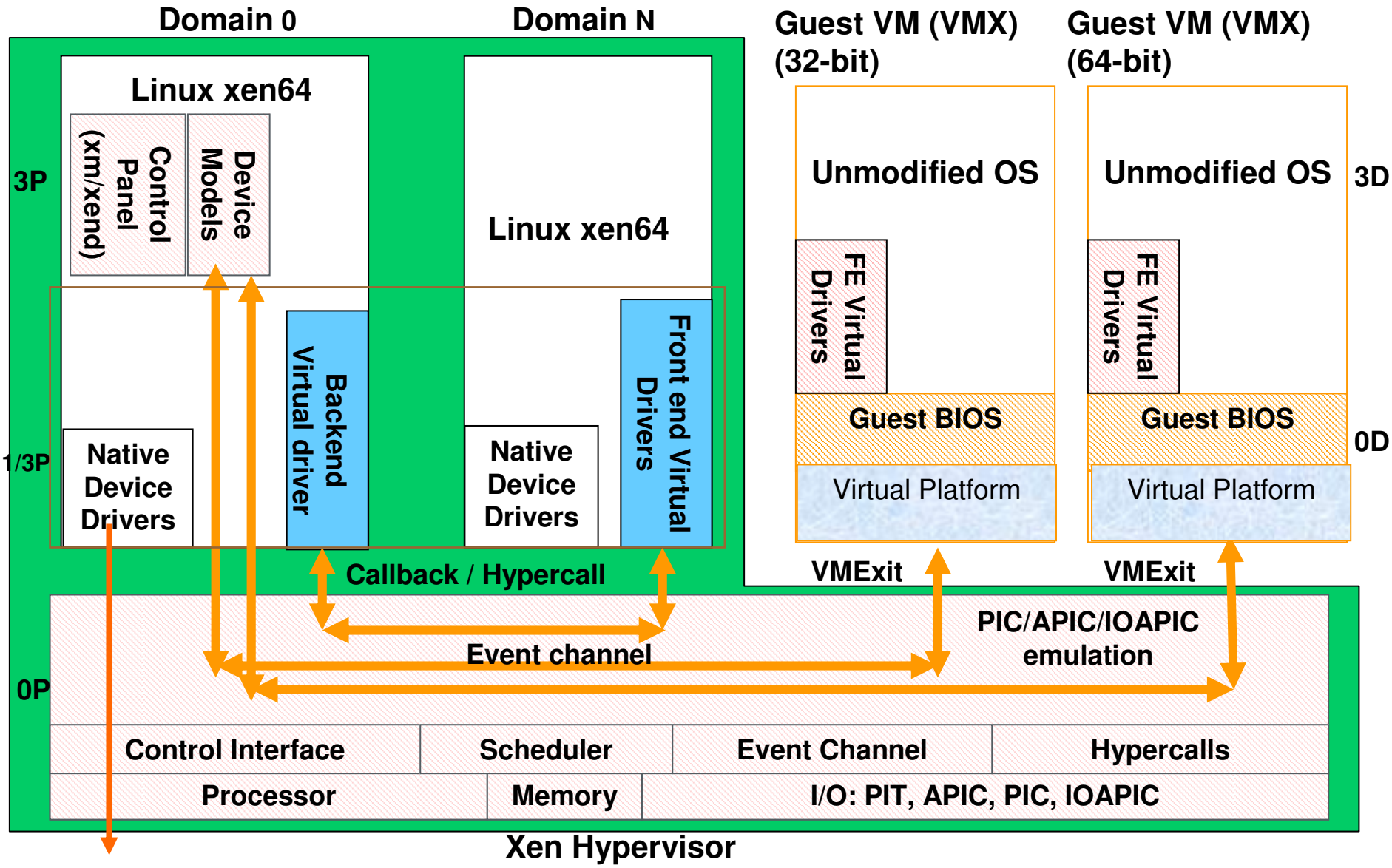
HVM Architecture



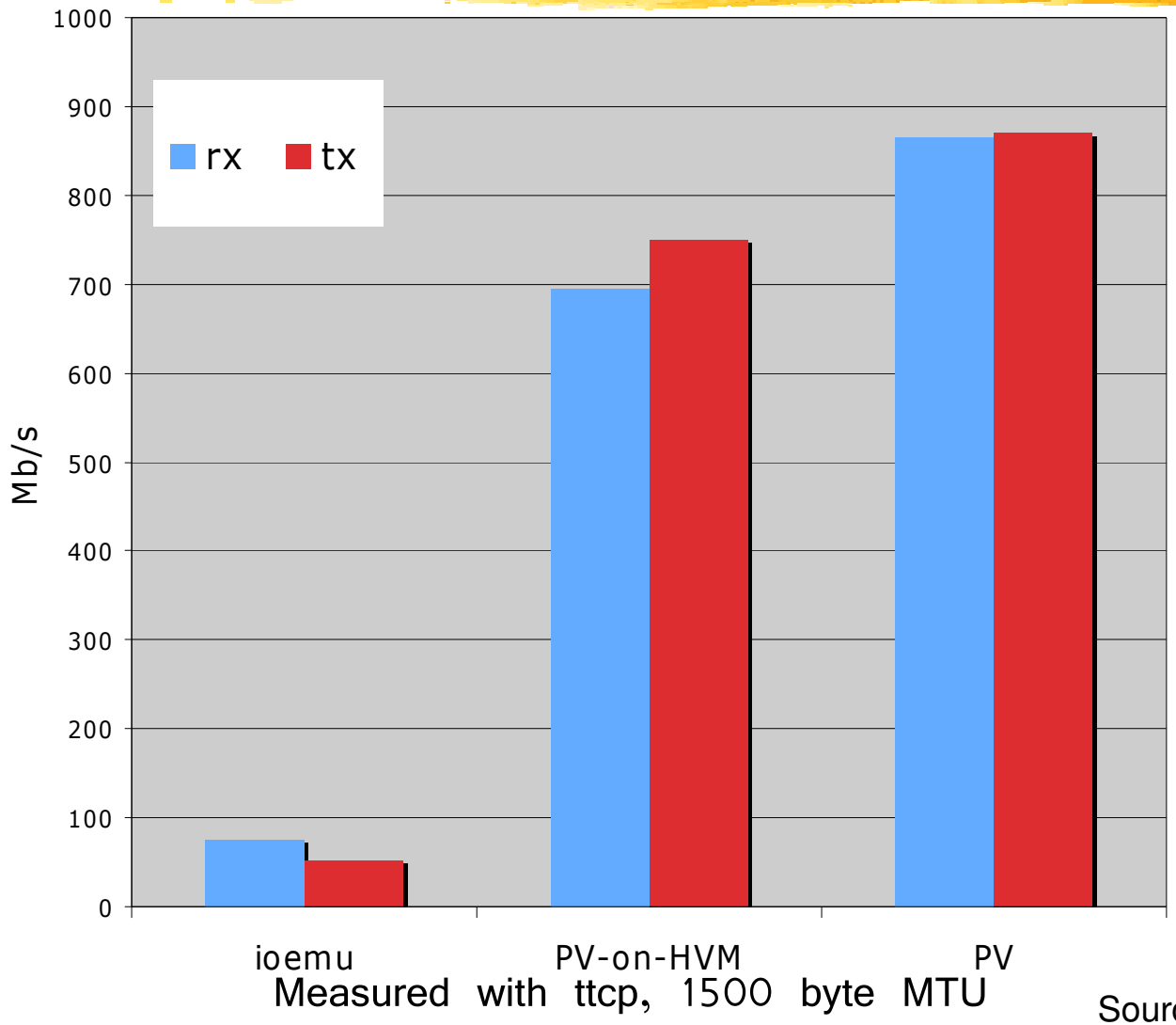
Progressive paravirtualization

- Hypercall API available to HVM guests
- Selectively add PV extensions to optimize
  - Net and Block IO
  - XenPIC (event channels)
  - MMU operations
    - multicast TLB flush
    - PTE updates (faster than page fault)
    - Page sharing
  - Time
  - CPU and memory hotplug

PV Drivers

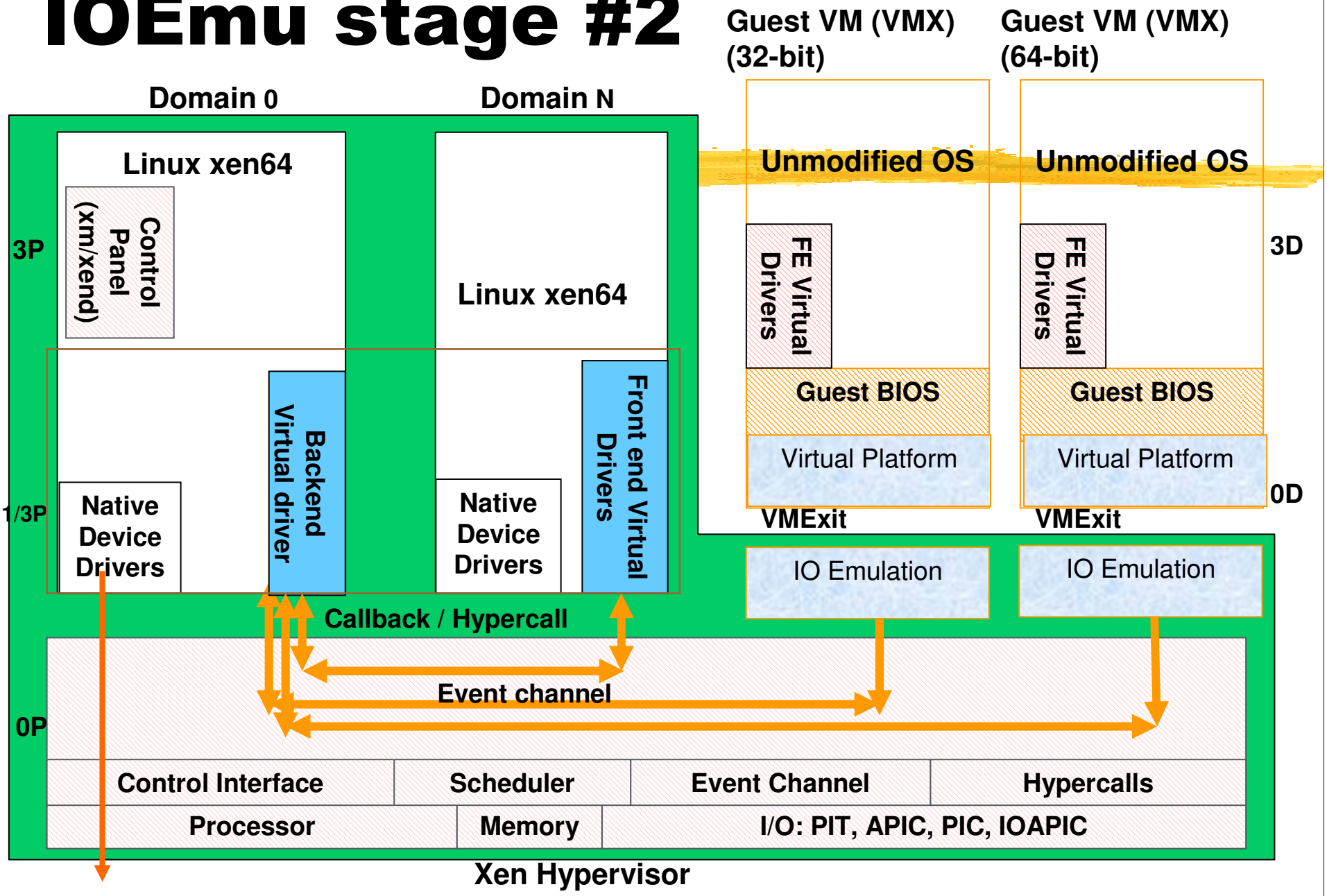

Domain 0
Domain N
Guest VM (VMX) (32-bit)
Guest VM (VMX) (64-bit)
Linux xen64
Control Panel (xm/xend)
Device Models
Linux xen64
Unmodified OS
Unmodified OS
FE Virtual Drivers
FE Virtual Drivers
3P
3D
Guest BIOS
Guest BIOS
0D
1/3P
Native Device Drivers
Backend Virtual driver
Native Device Drivers
Front end Virtual Drivers
Virtual Platform
Virtual Platform
Callback / Hypercall
VMExit
VMExit
PIC/APIC/IOAPIC emulation
Event channel
0P
Control Interface
Scheduler
Event Channel
Hypercalls
Processor
Memory
I/O: PIT, APIC, PIC, IOAPIC
Xen Hypervisor

HVM I/O Performance



Source: XenSource, Sep 06

IOEmu stage #2

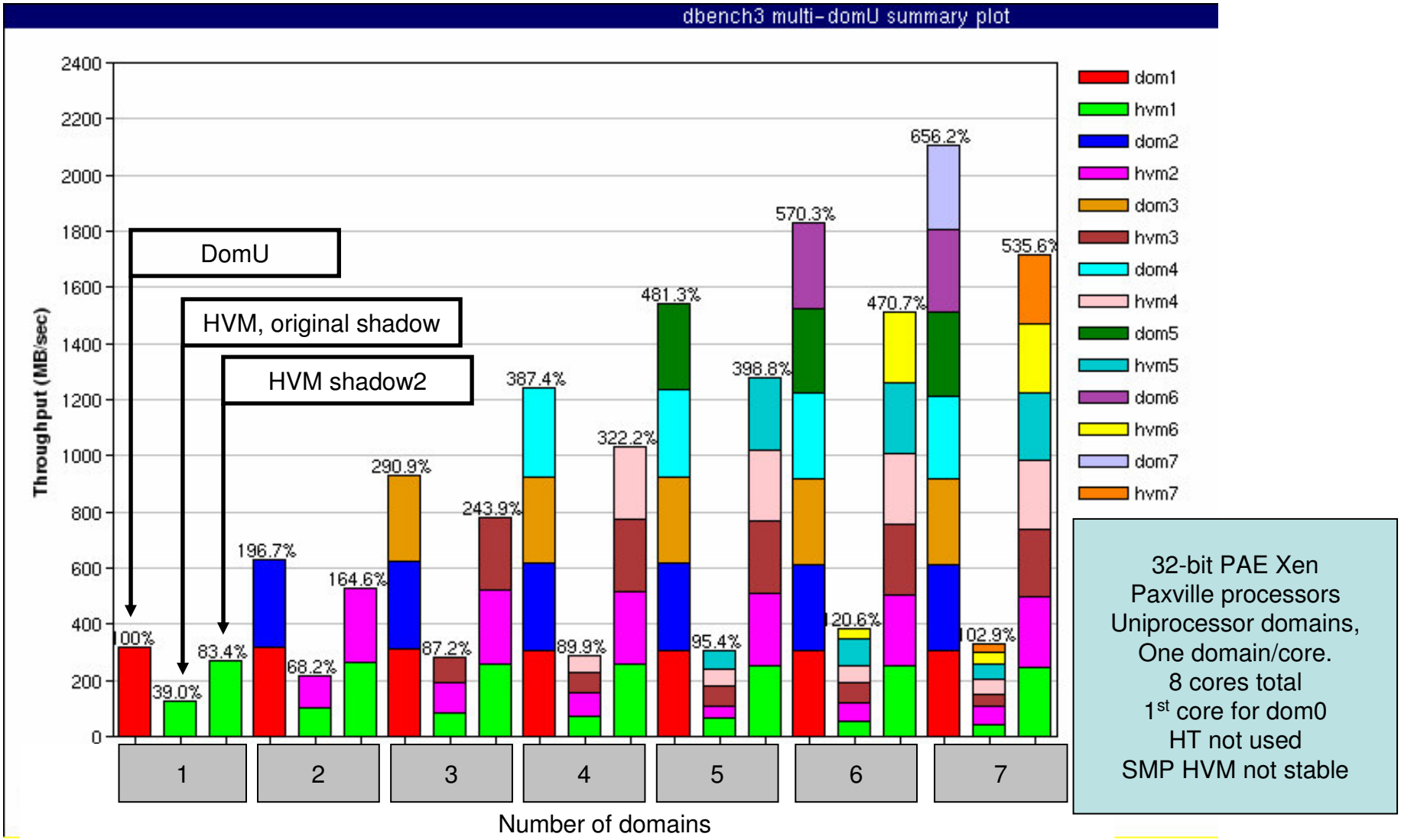


HVM Performance

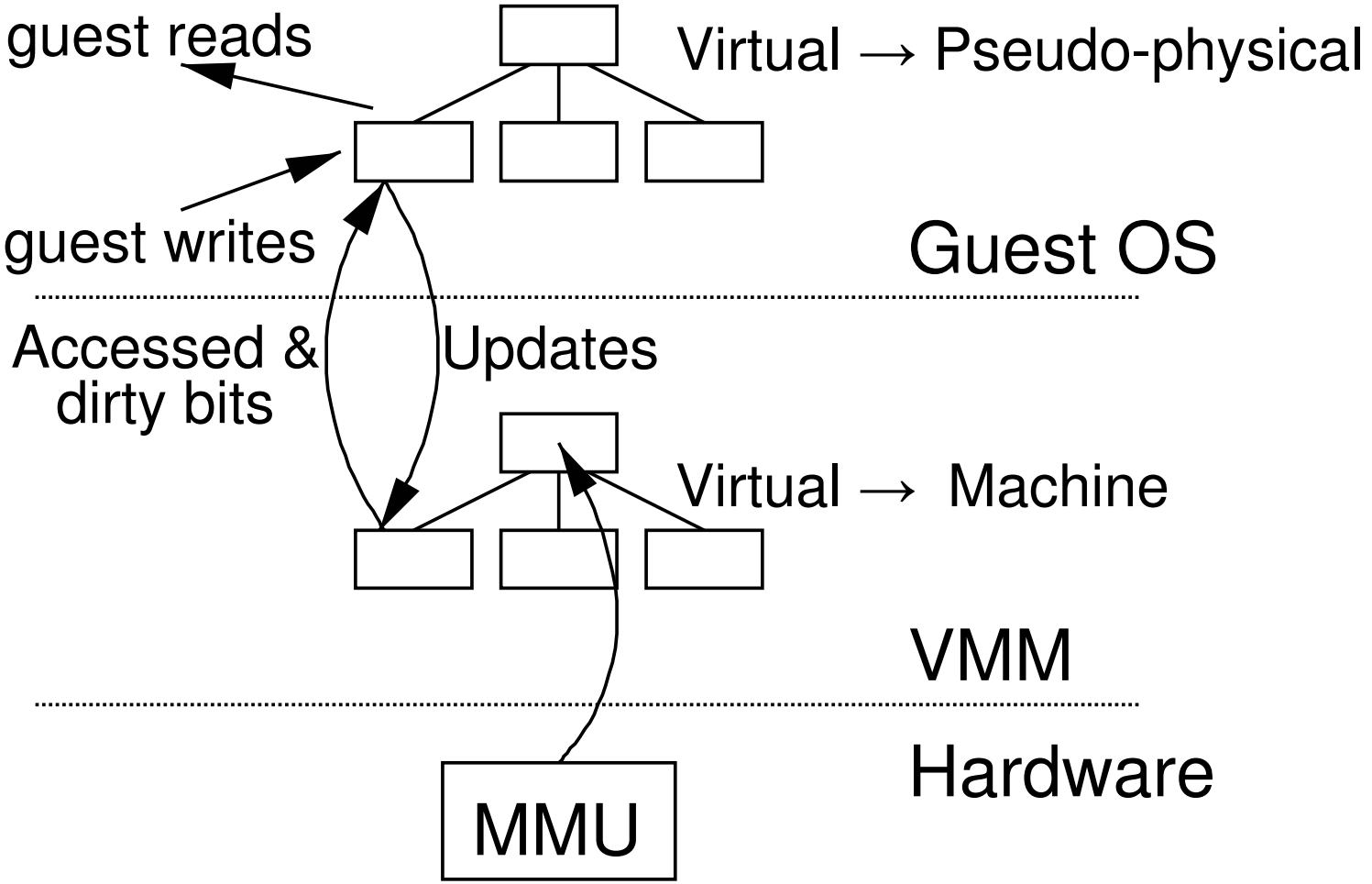
- Very application dependent
  - 5% SPECJBB (0.5% for fully PV)
  - OS-intensive applications suffer rather more
  - Performance certainly in-line with existing products
  - Hardware support gets better every new CPU
- More optimizations forthcoming
  - “V2E” for trap-intensive code sequences
  - New shadow pagetable code

Full Virtualization

- Shadow2 code shows vast improvement in processor performance/scaling
  - Boosts single domain throughput significantly -Dbench (running in tmpfs filesystem, **NO I/O**)
  - Many HVM domain scalability also improved dramatically
    - 1 to 7 HVMs scaled 1 : 6.47 with shadow2. Same test scaled 1 : 2.63 with old shadow code



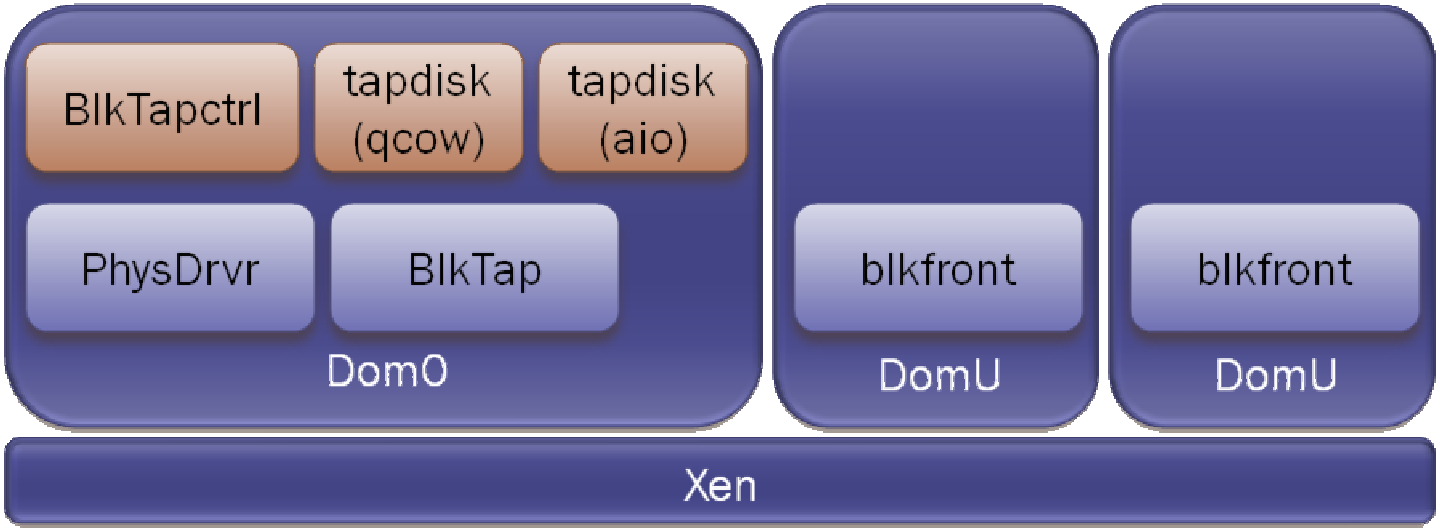
Shadow Pagetables



Virtual Disk Storage

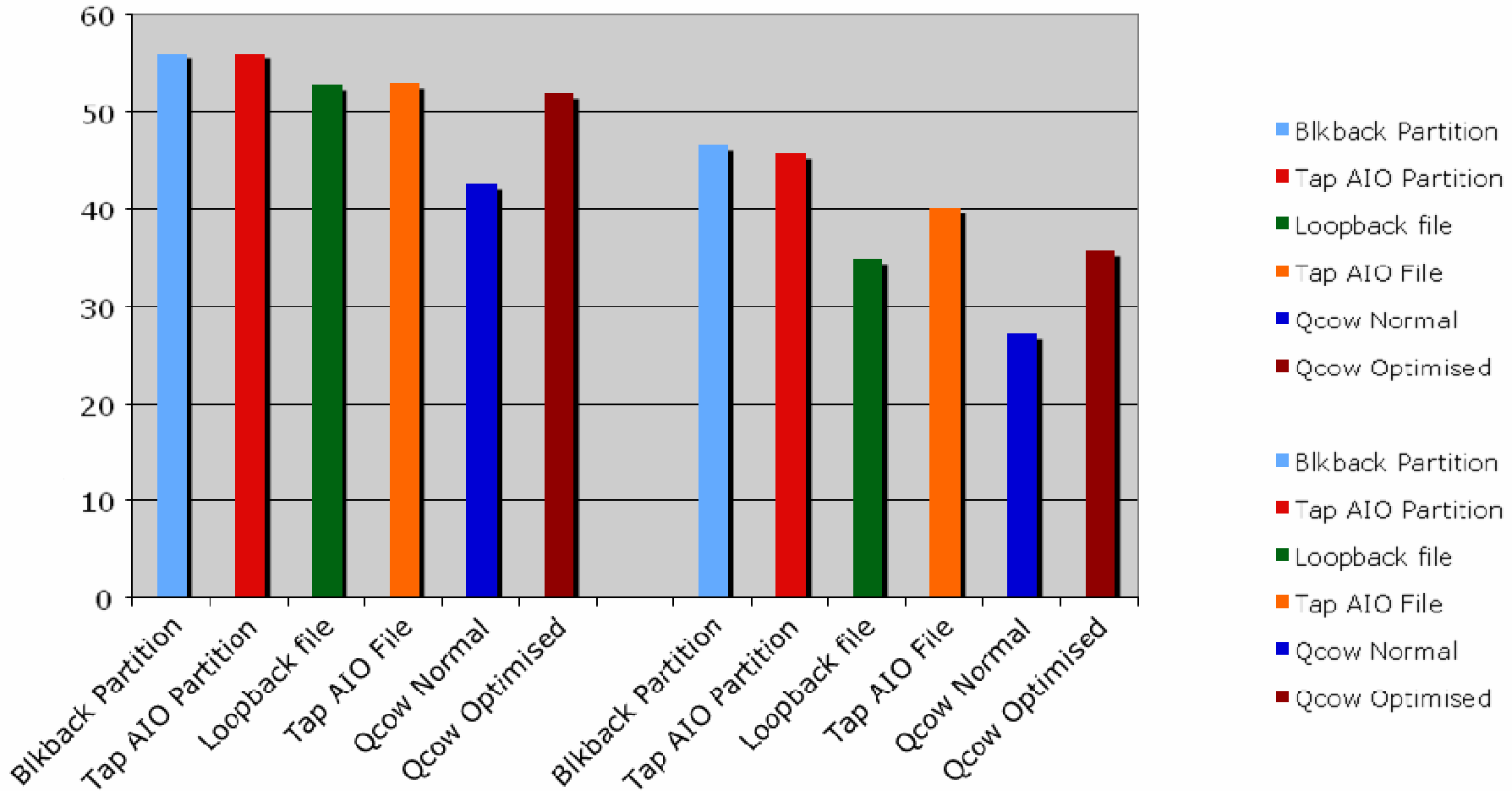
- LVM Logical Volumes are typically used to store guest virtual disk images today
  - Not as intuitive as using files
  - Copy-on-write and snapshot support far from ideal
- Storing disk images in files using Linux's "loop" driver doesn't work well
- The "Blktap" driver new in Xen 3.0.3 provides an alternative :
  - Allows all block requests to be serviced in user-space using zero-copy AIO approach

Blktap and tapdisk plug-ins



- Char device mapped by user-space driver
- Request/completion queues and data areas
- Grant table mapping for zero-copy to/from guest
- Flat files and qcow
- Sparse allocation, CoW, encryption, compression
- Correct metadata update safety
- Optimized qcow format

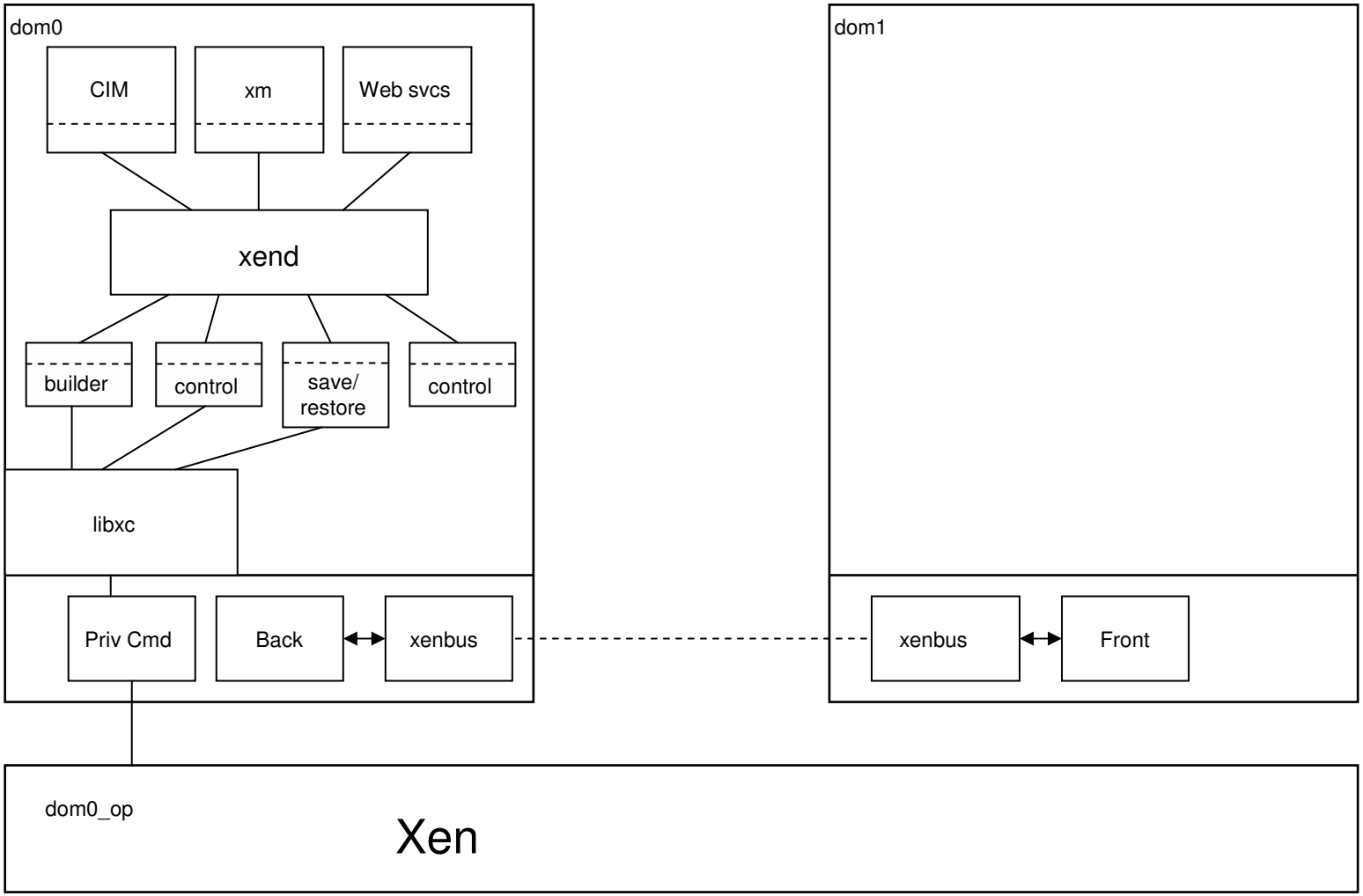
Blktap IO performance



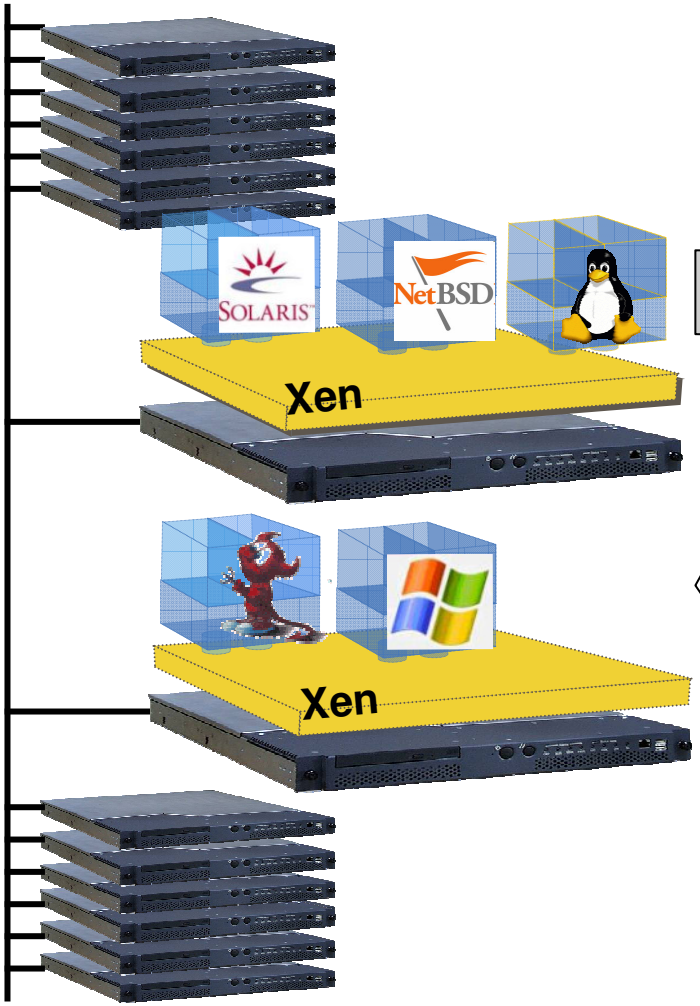
Time API

- Shared info page contains per VCPU time records
  - “at TSC X the time was Y, and the current TSC frequency has been measured as Z”
  - gettime: Read current TSC and extrapolate
- When VCPUs migrate, update record for new physical CPU
- Periodic calibration of TSC's against pit/hpet
  - Works even on systems with un-synced TSCs
  - Update record after CPU frequency changes (p-state)
  - Also, resynchronize records after CPU halt
  - Only issue is thermal thottling

Xen Tools

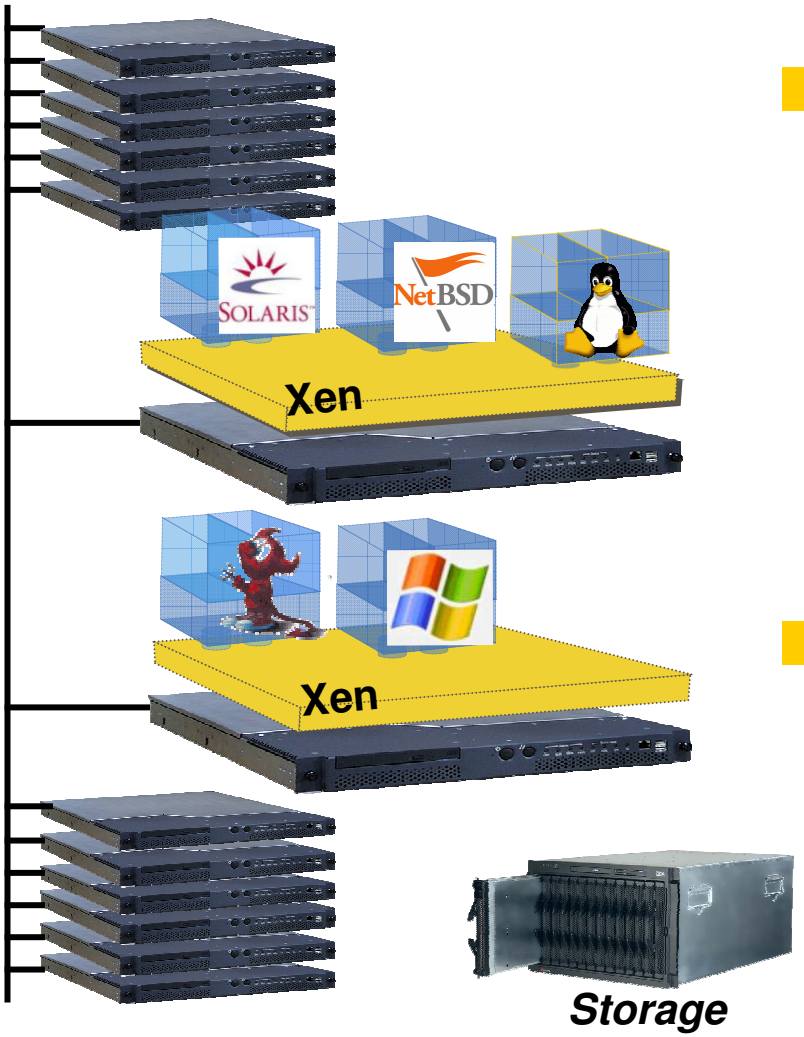

dom0
CIM
xm
Web svcs
xend
builder
control
save/
restore
control
libxc
Priv Cmd
Back
xenbus
dom1
xenbus
Front
dom0_op
Xen

VM Relocation : Motivation



- VM relocation enables:
  - High-availability
    - Machine maintenance
  - Load balancing
    - Statistical multiplexing gain

Assumptions

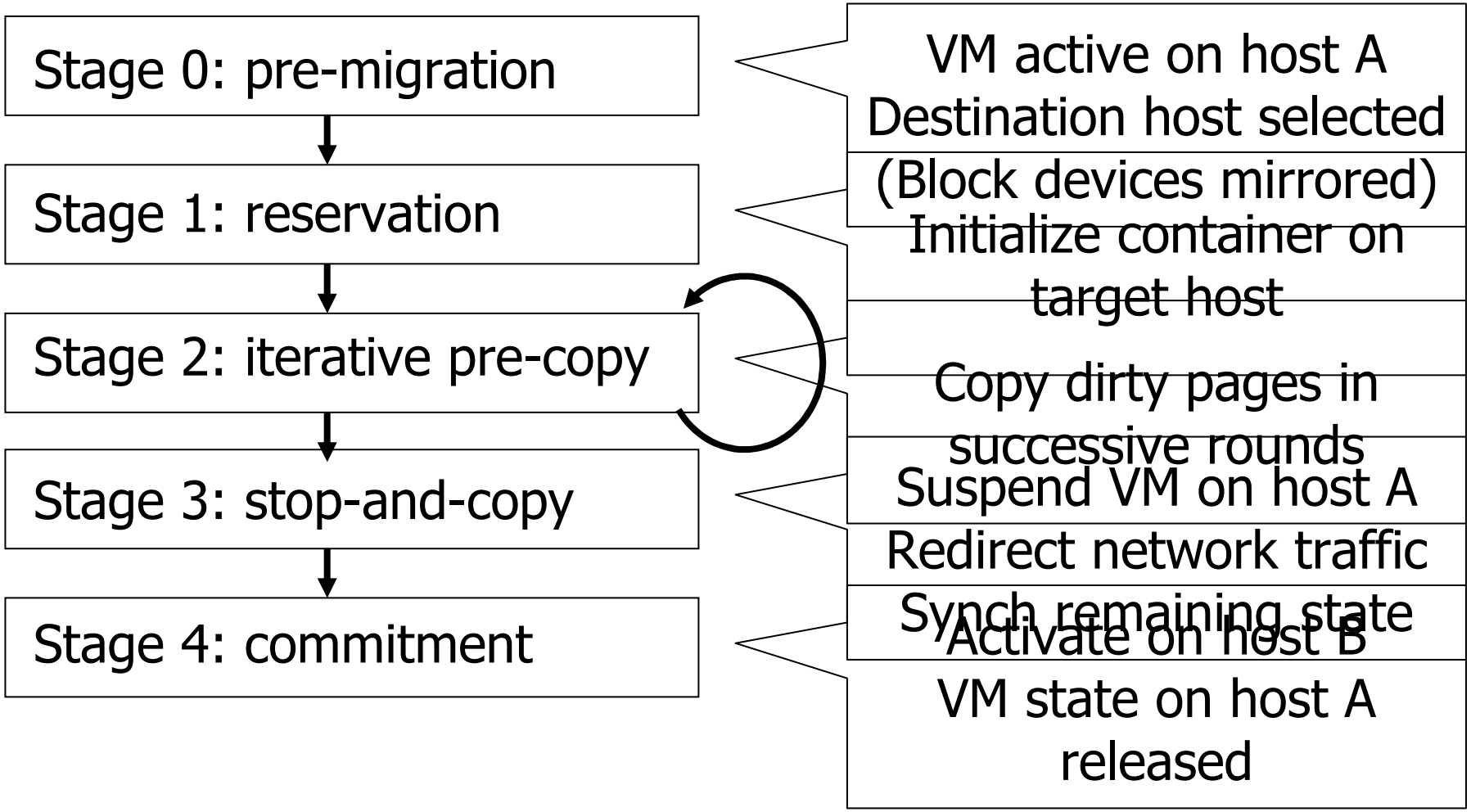


- Networked storage
  - NAS: NFS, CIFS
  - SAN: Fibre Channel
  - iSCSI, network block dev
  - drdb network RAID
- Good connectivity
  - common L2 network
  - L3 re-routeing

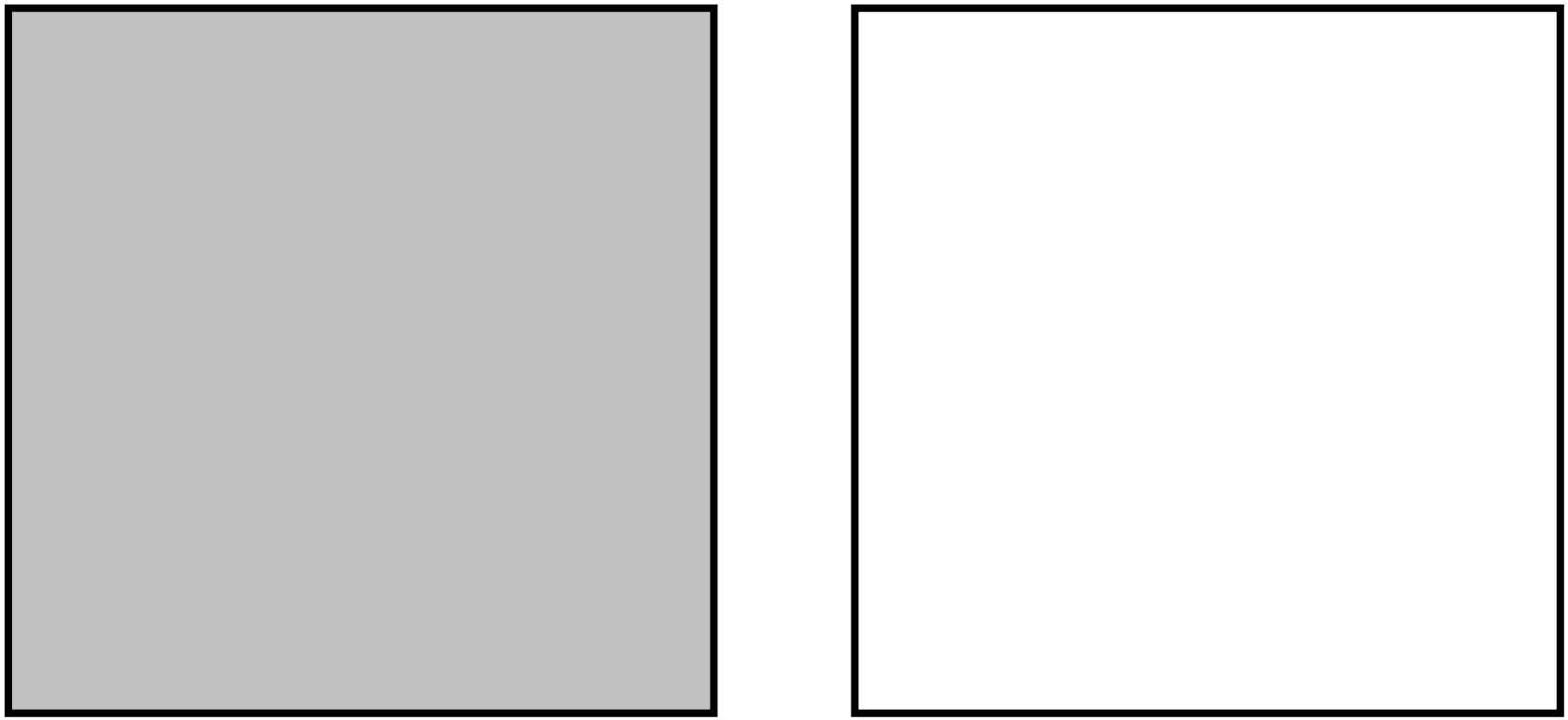
Challenges

- VMs have lots of state in memory
- Some VMs have soft real-time requirements
  - E.g. web servers, databases, game servers
  - May be members of a cluster quorum
  - → **Minimize down-time**
- Performing relocation requires resources
  - → **Bound and control resources used**

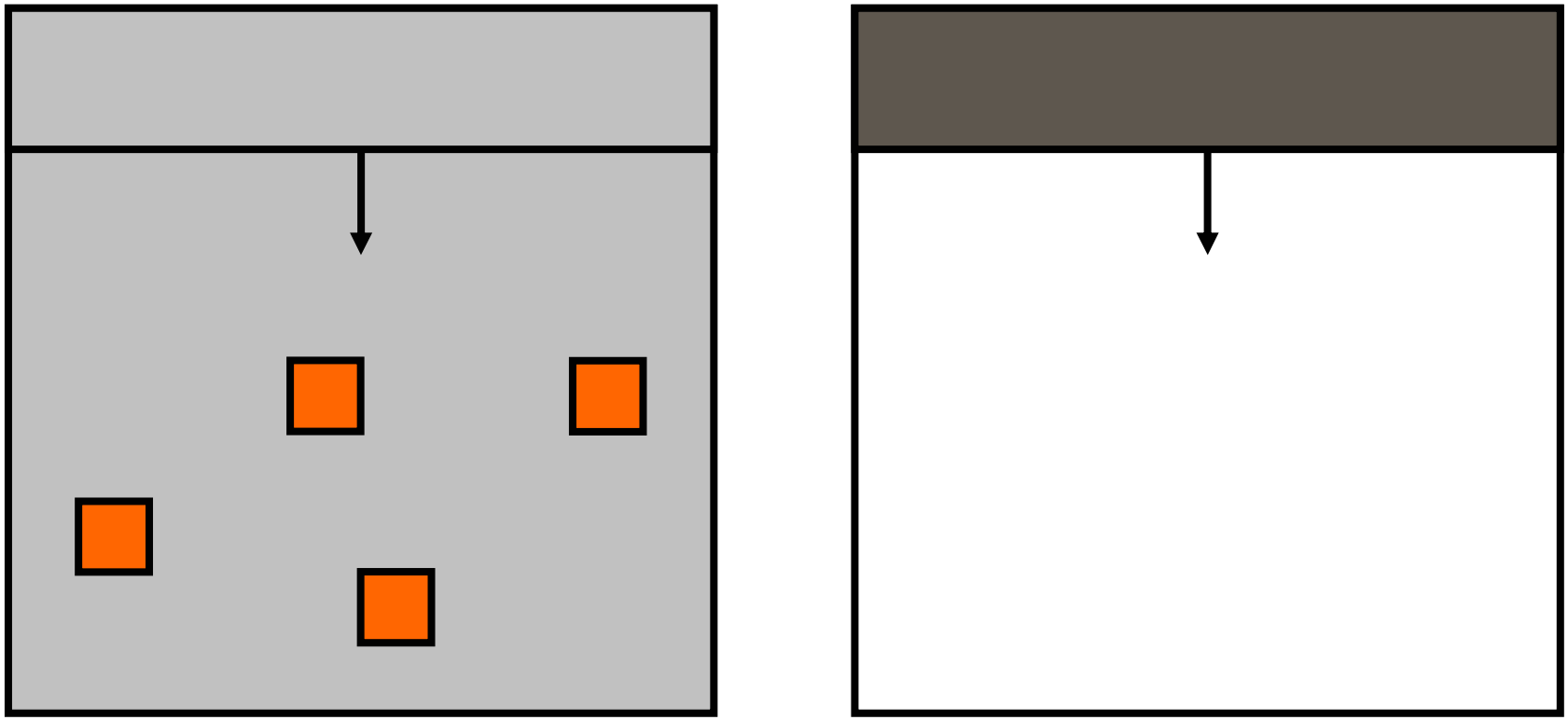
Relocation Strategy



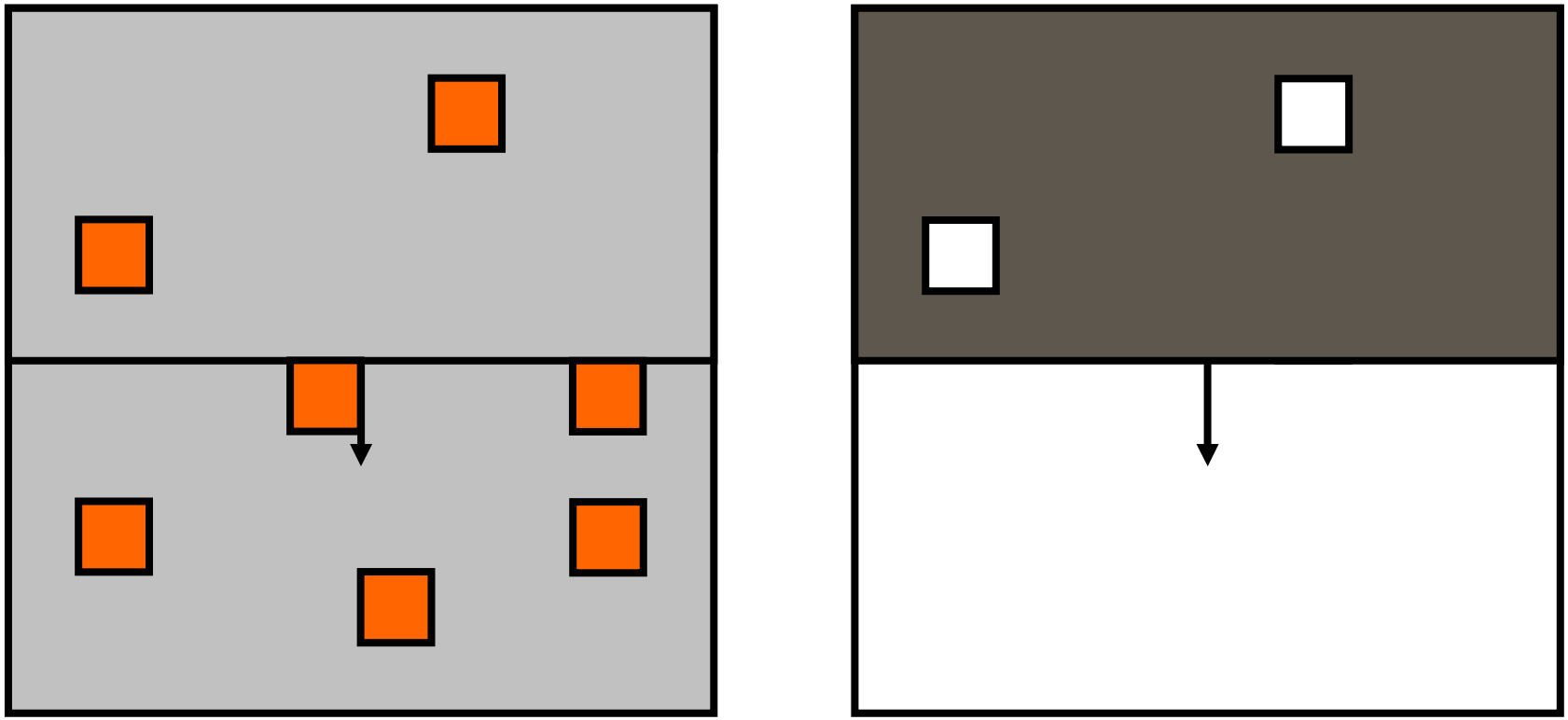
Pre-Copy Migration: Round 1



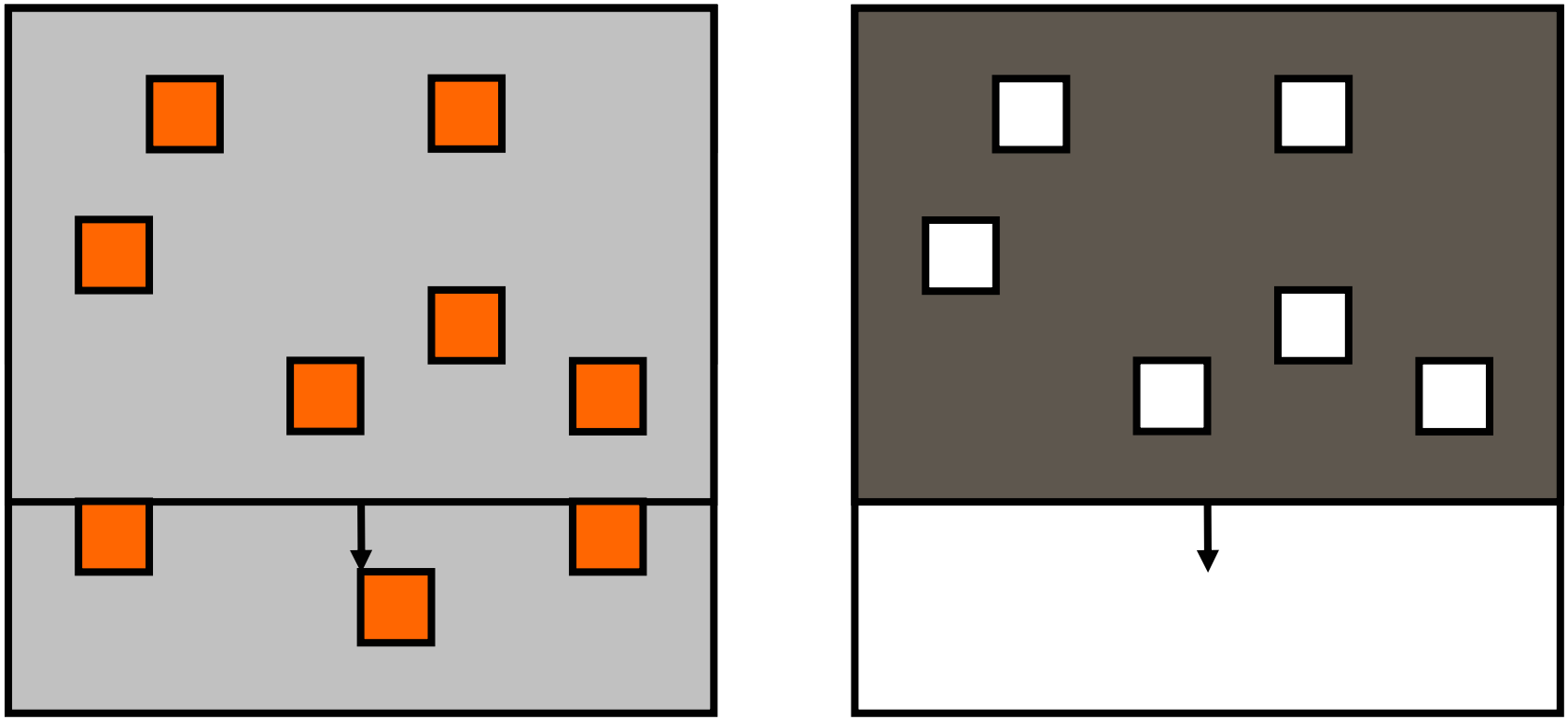
Pre-Copy Migration: Round 1



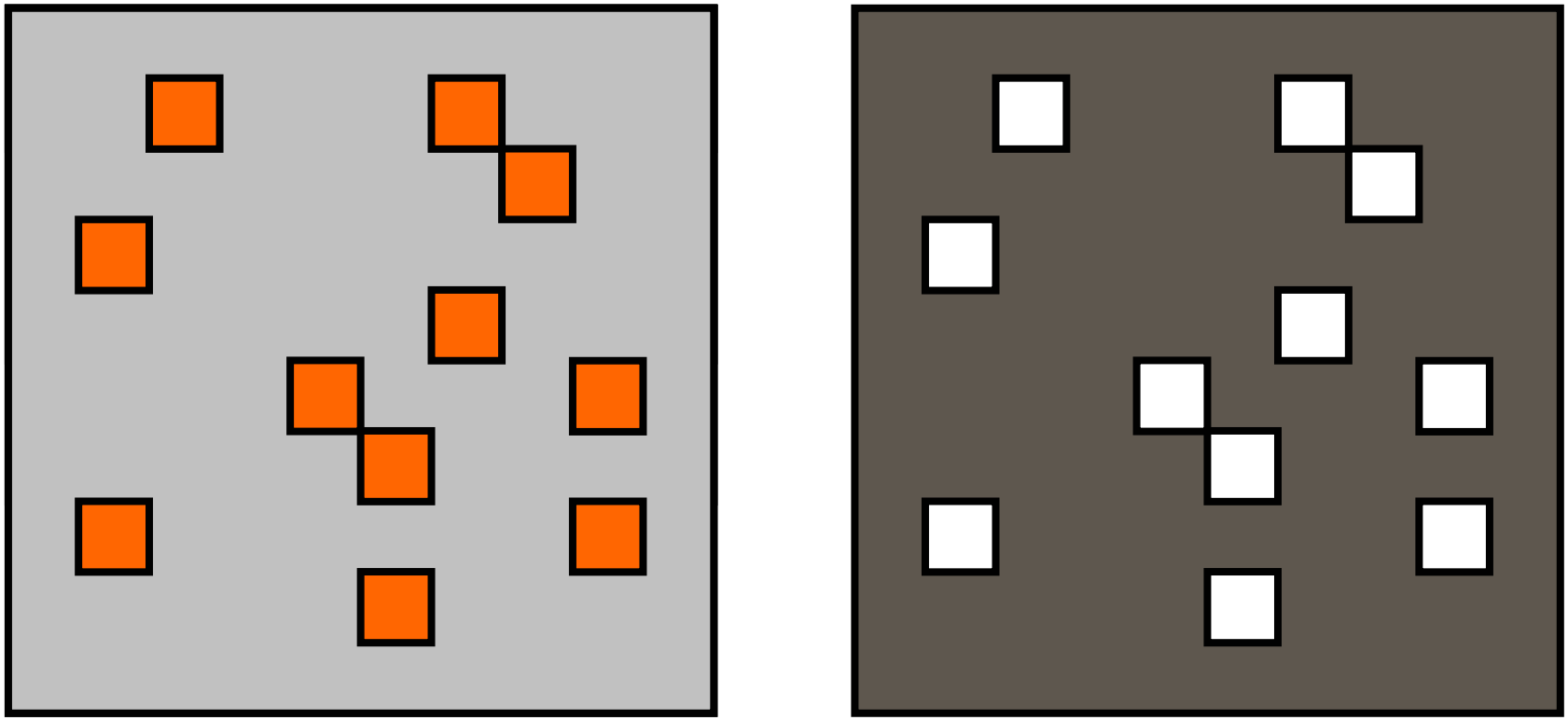
Pre-Copy Migration: Round 1



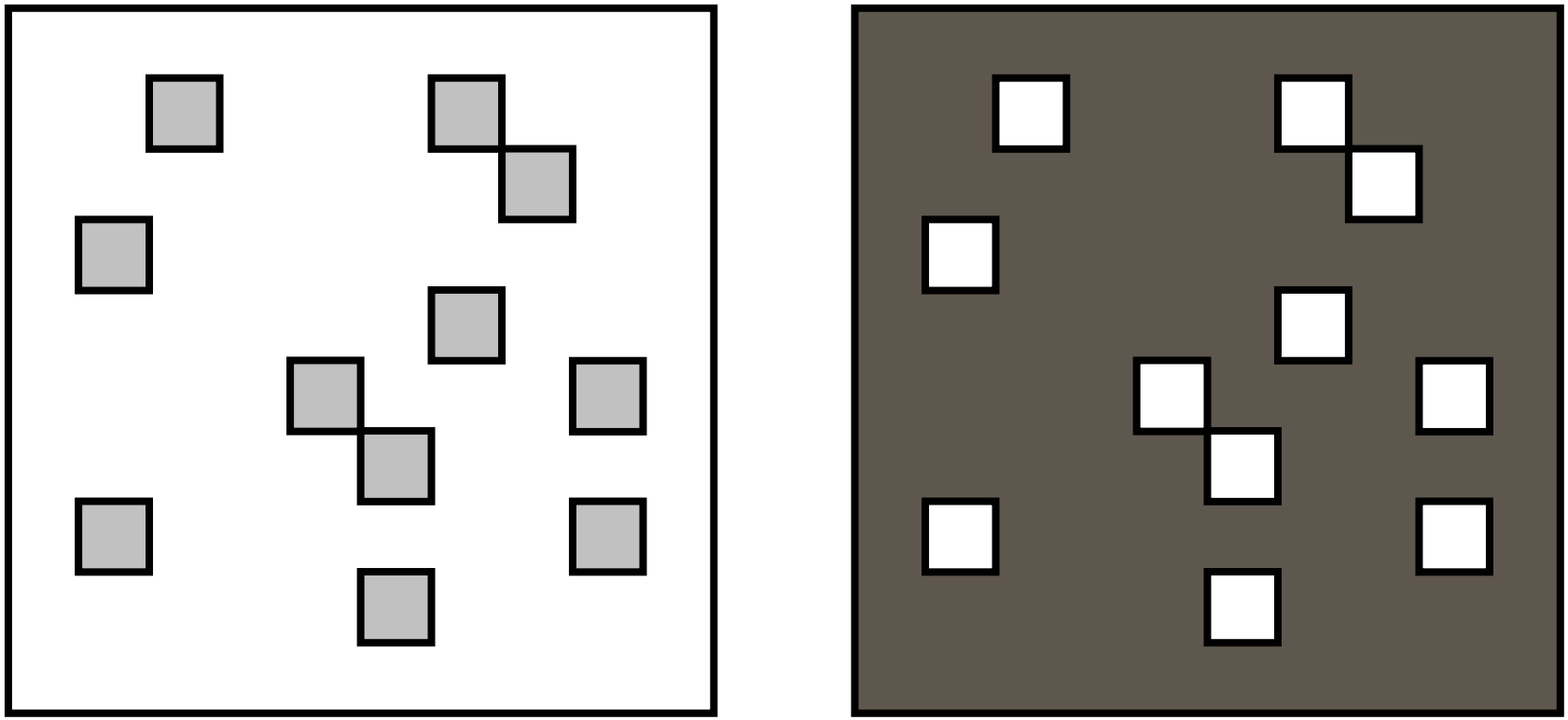
Pre-Copy Migration: Round 1



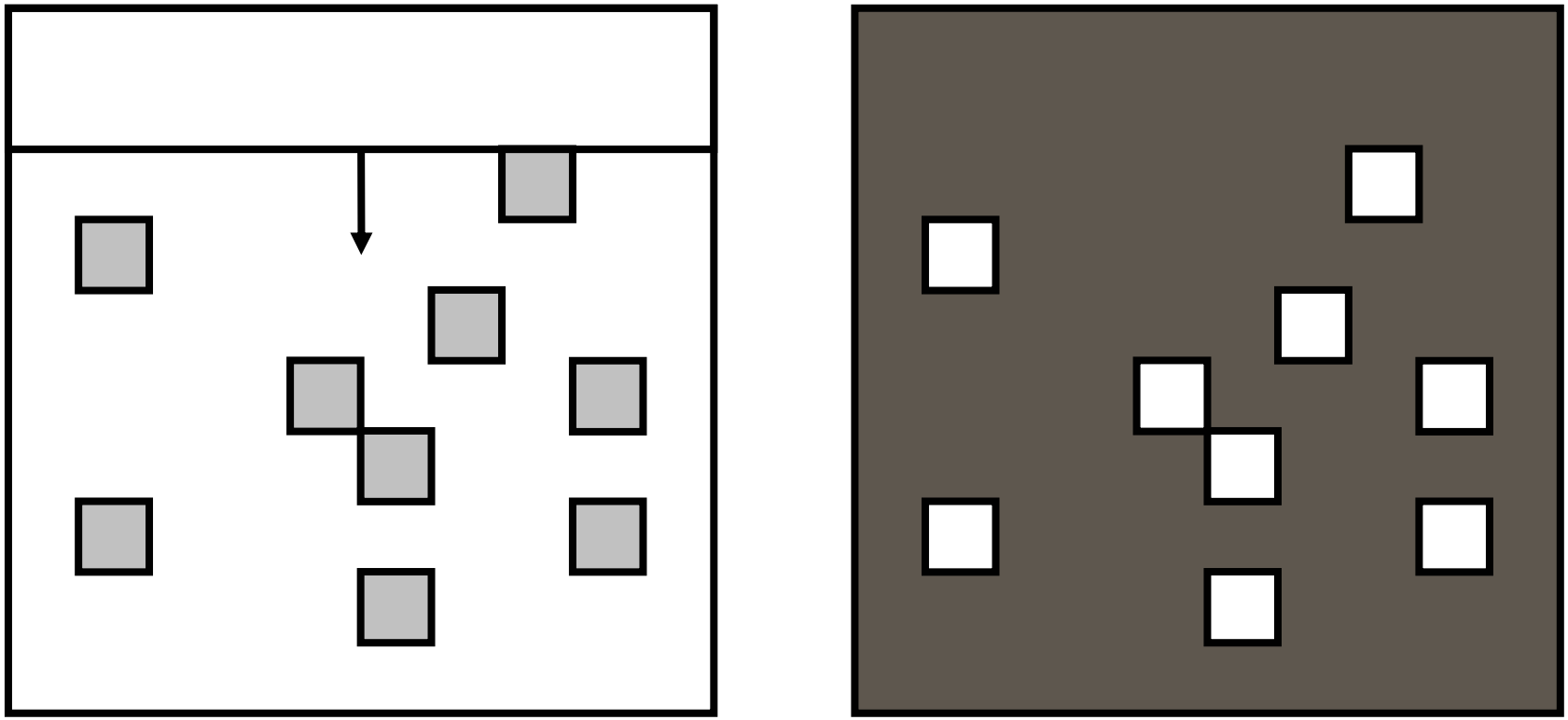
Pre-Copy Migration: Round 1



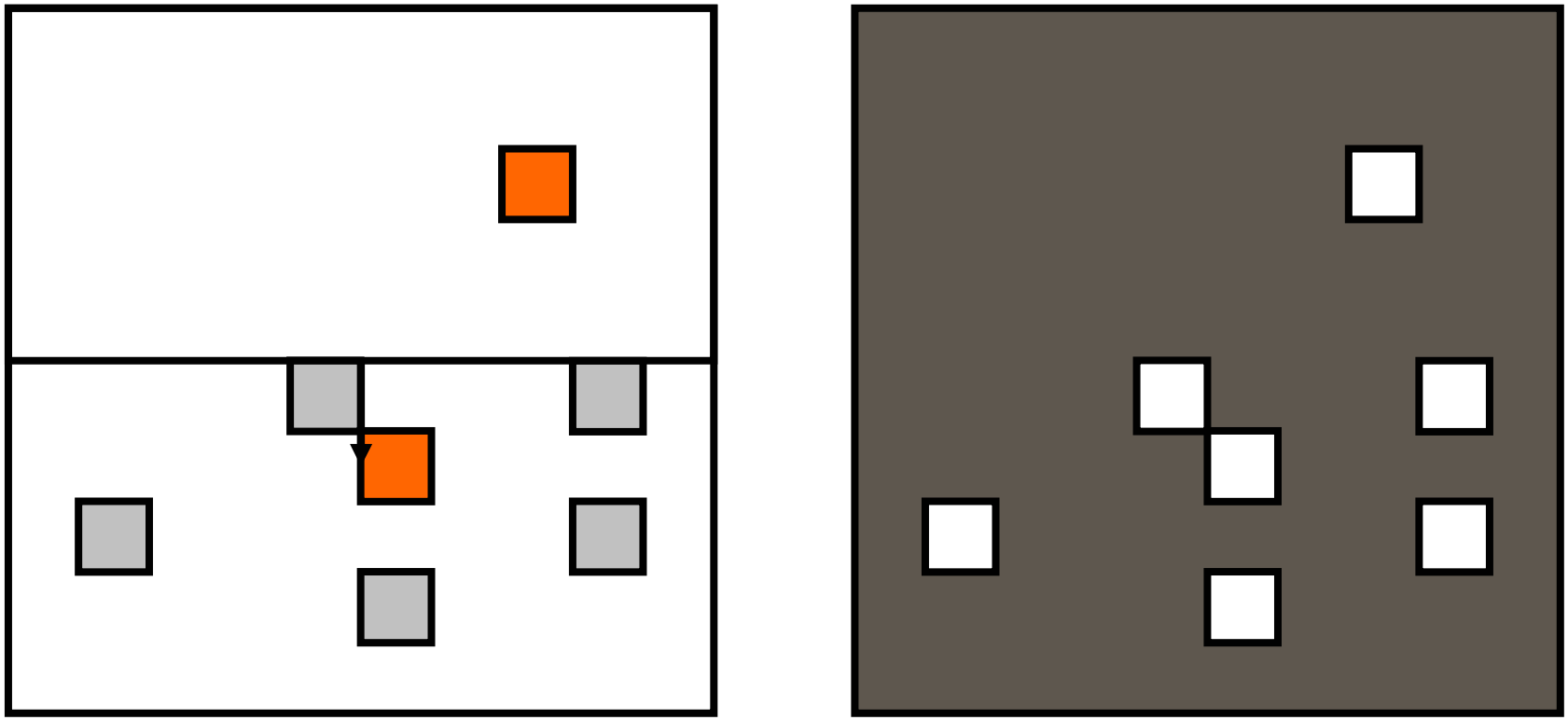
Pre-Copy Migration: Round 2



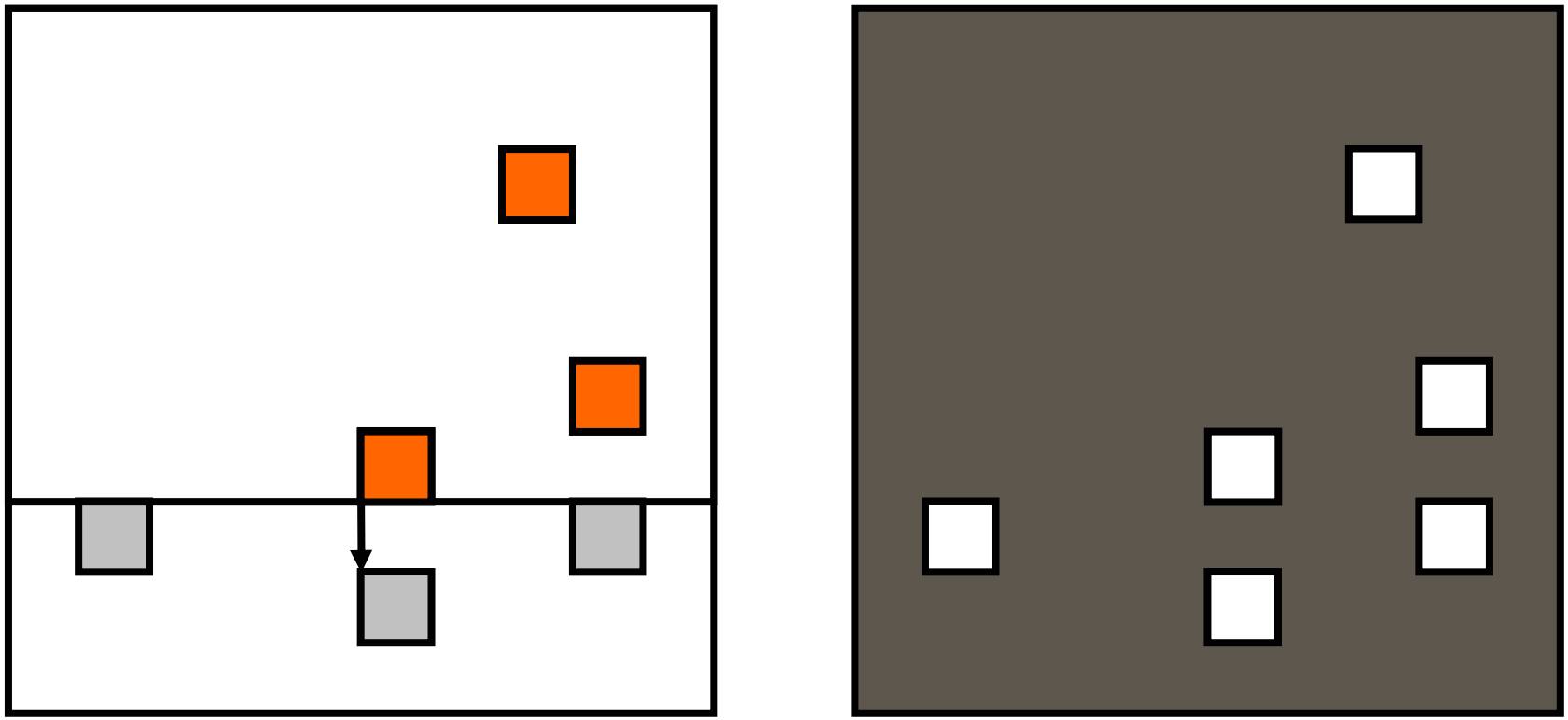
Pre-Copy Migration: Round 2



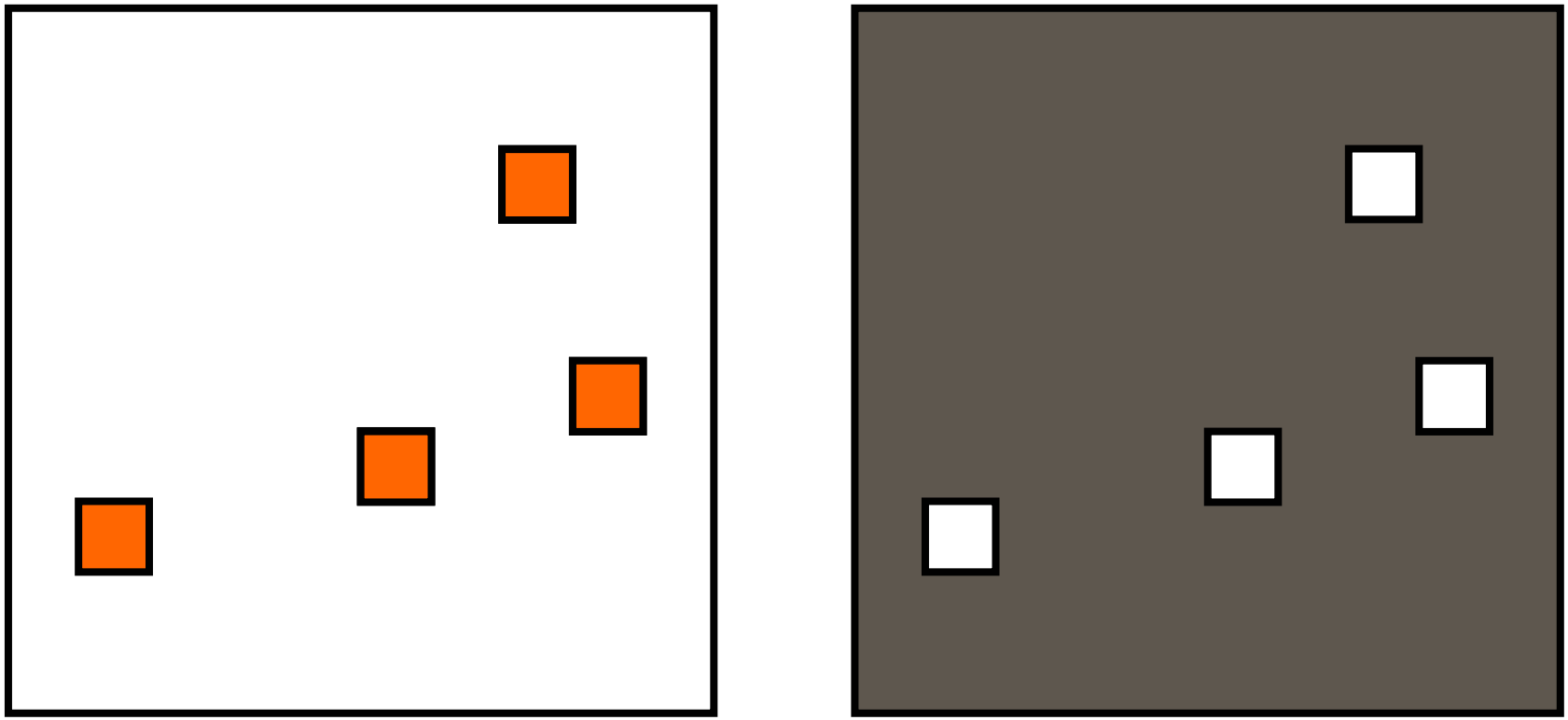
Pre-Copy Migration: Round 2



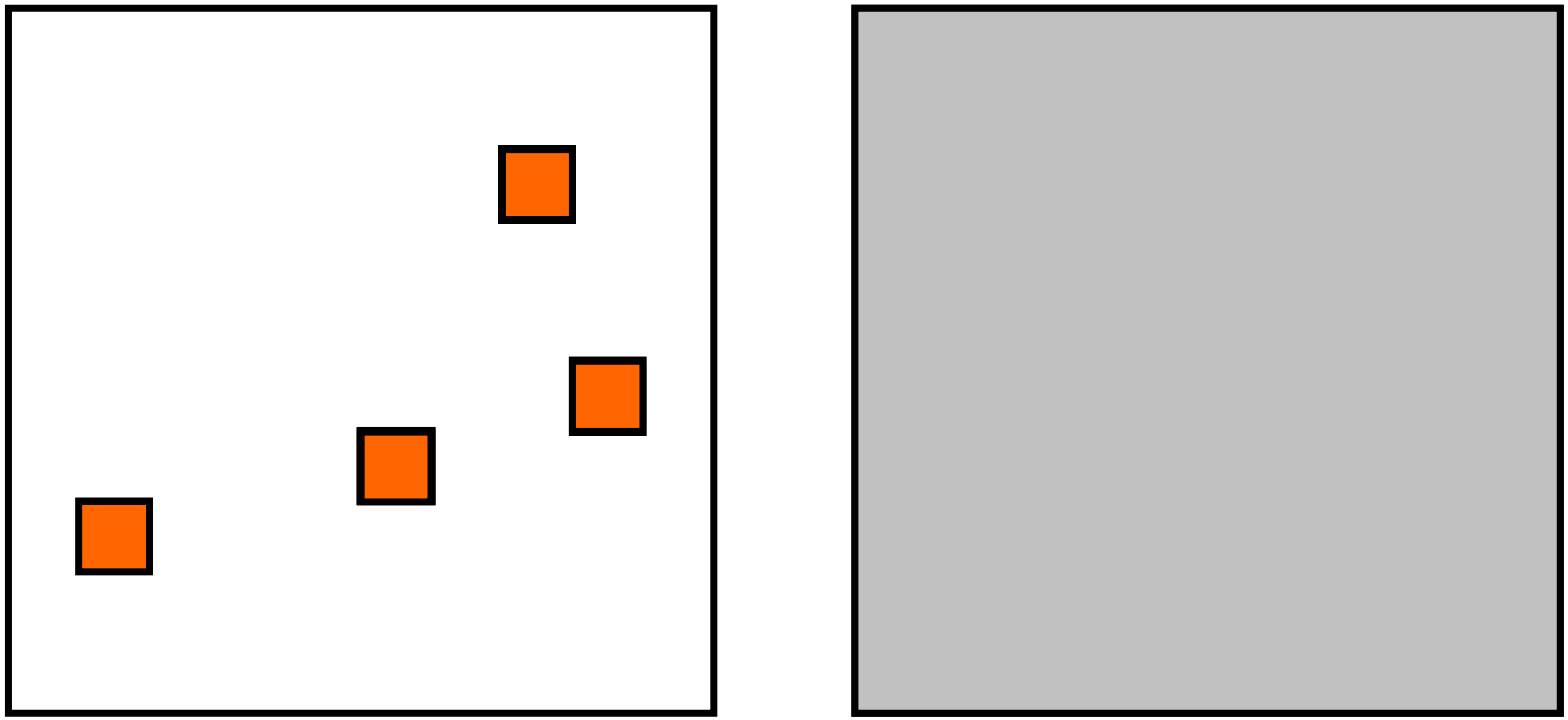
Pre-Copy Migration: Round 2



Pre-Copy Migration: Round 2



Pre-Copy Migration: Final



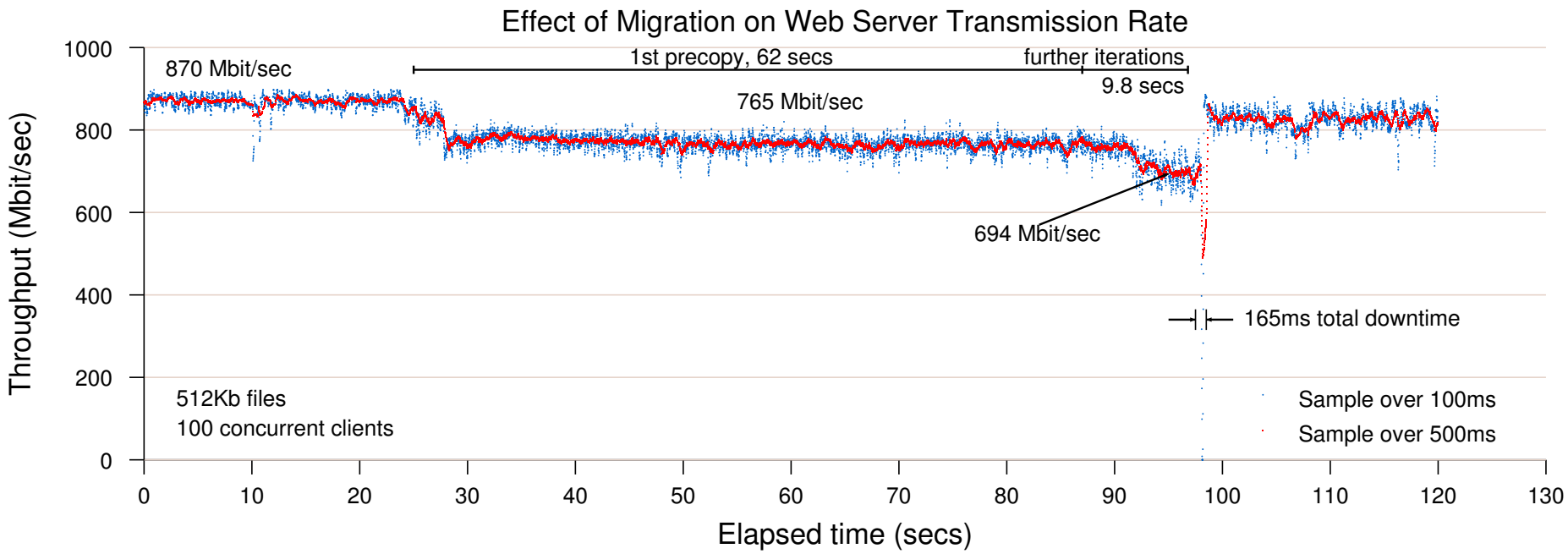
Writable Working Set

- Pages that are dirtied must be re-sent
  - Super hot pages
    - e.g. process stacks; top of page free list
  - Buffer cache
  - Network receive / disk buffers
- Dirtying rate determines VM down-time
  - Shorter iterations → less dirtying → …

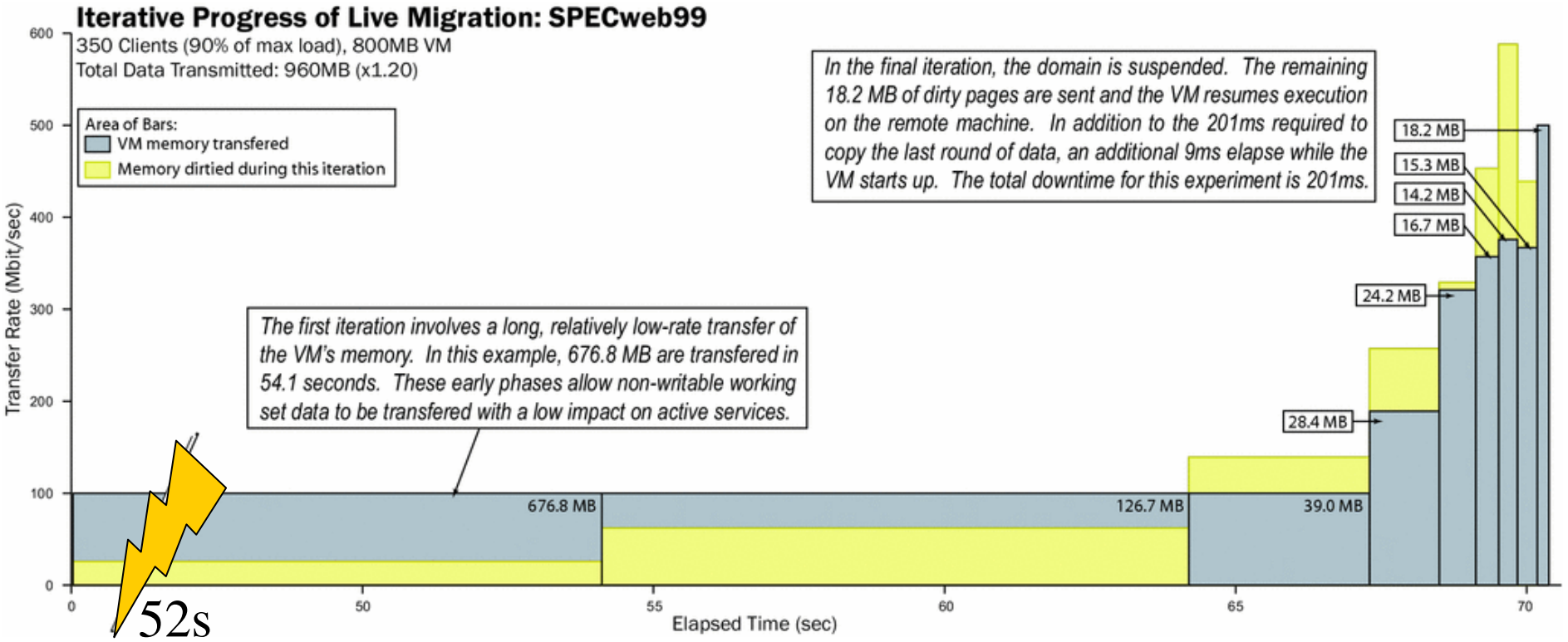
Rate Limited Relocation

- Dynamically adjust resources committed to performing page transfer
  - Dirty logging costs VM ~2-3%
  - CPU and network usage closely linked
- E.g. first copy iteration at 100Mb/s, then increase based on observed dirtying rate
  - Minimize impact of relocation on server while minimizing down-time

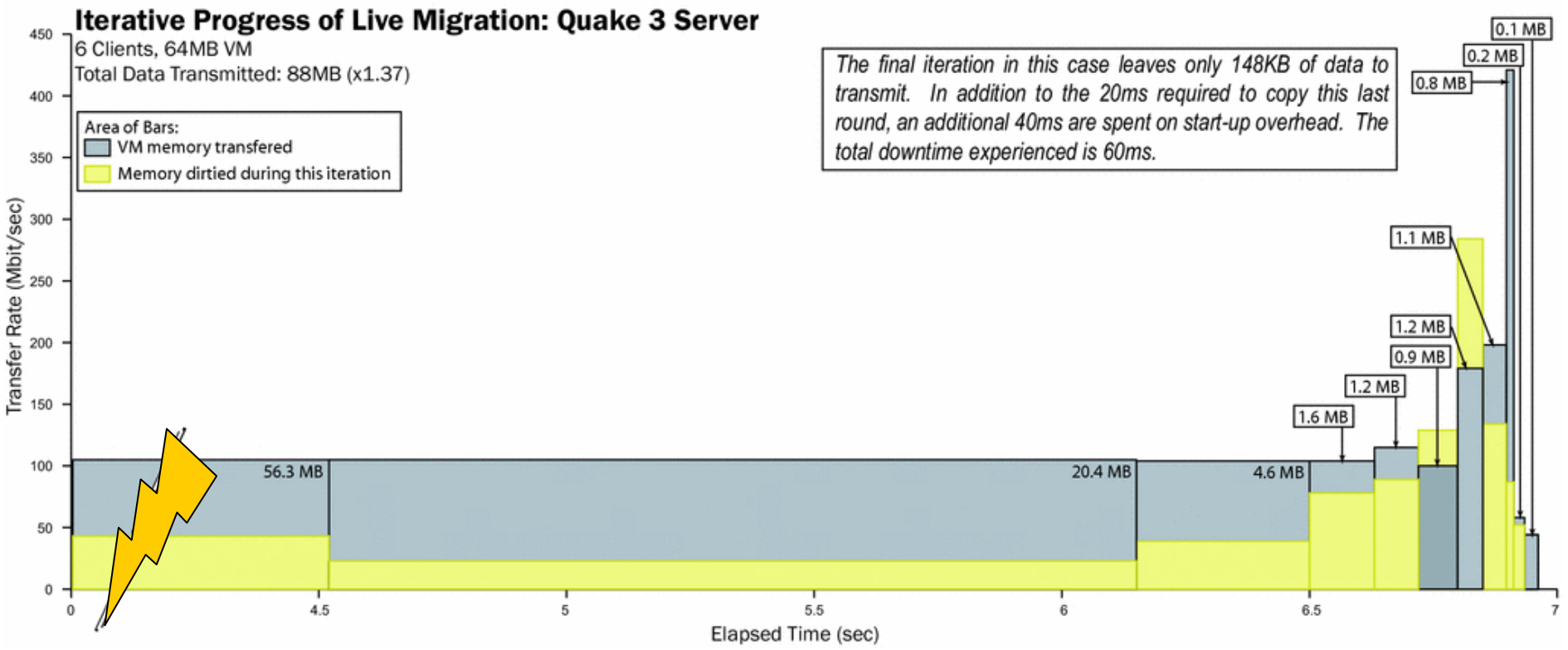
Web Server Relocation



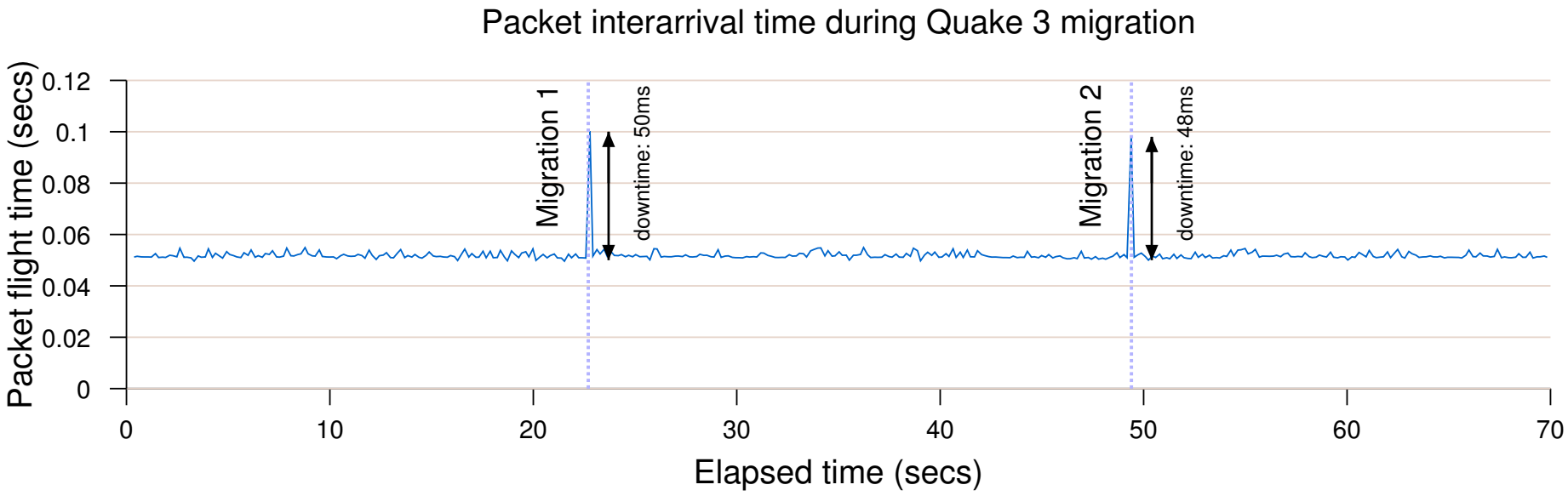
Iterative Progress: SPECWeb



Iterative Progress: Quake3



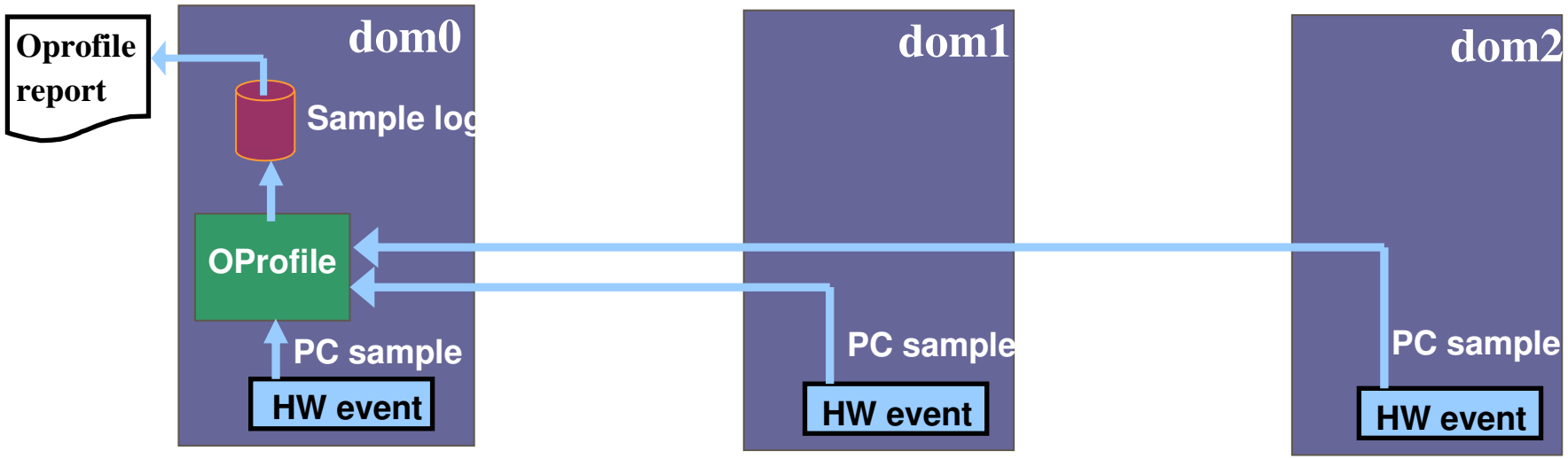
Quake 3 Server relocation



System Debugging on Xen

- Guest crash dump support
  - Post-mortem analysis
- gdbserver
  - No need for a kernel debugger
- Xentrace
  - Fine-grained event tracing
  - xenmon/xentop
- Xen oprofile
  - Sample-based profiling

Xen Oprofile



- Sample-based profiling for the whole system

Xen Oprofile example

CPU: P4 / Xeon with 2 hyper-threads, speed 2794.57 MHz (estimated)
Counted GLOBAL_POWER_EVENTS events with a unit mask of 0x01 (mandatory) count 1000000

samples	%	image name	app name	symbol name
30353	12.0434	domain1-kernel	domain1-kernel	__copy_to_user_ll
7174	2.8465	xen-syms-3.0-unstable	xen-syms-3.0-unstable	do_grant_table_op
6040	2.3965	vmlinux-syms-2.6.16.13-xen0	vmlinux-syms-2.6.16.13-xen0	net_tx_action
5508	2.1854	xen-syms-3.0-unstable	xen-syms-3.0-unstable	find_domain_by_id
4944	1.9617	xen-syms-3.0-unstable	xen-syms-3.0-unstable	gnttab_transfer
4848	1.9236	domain1-xen	domain1-xen	evtchn_set_pending
4631	1.8375	vmlinux-syms-2.6.16.13-xen0	vmlinux-syms-2.6.16.13-xen0	net_rx_action
4322	1.7149	domain1-kernel	domain1-kernel	tcp_v4_rcv
4145	1.6446	xen-syms-3.0-unstable	xen-syms-3.0-unstable	hypercall
4005	1.5891	domain1-xen	domain1-xen	guest_remove_page
3644	1.4459	vmlinux-syms-2.6.16.13-xen0	vmlinux-syms-2.6.16.13-xen0	hypercall_page
3589	1.4240	vmlinux-syms-2.6.16.13-xen0	vmlinux-syms-2.6.16.13-xen0	eth_type_trans
3425	1.3590	domain1-xen	domain1-xen	get_page_from_l1e
2846	1.1292	domain1-kernel	domain1-kernel	eth_type_trans
2770	1.0991	vmlinux-syms-2.6.16.13-xen0	vmlinux-syms-2.6.16.13-xen0	e1000_intr

…

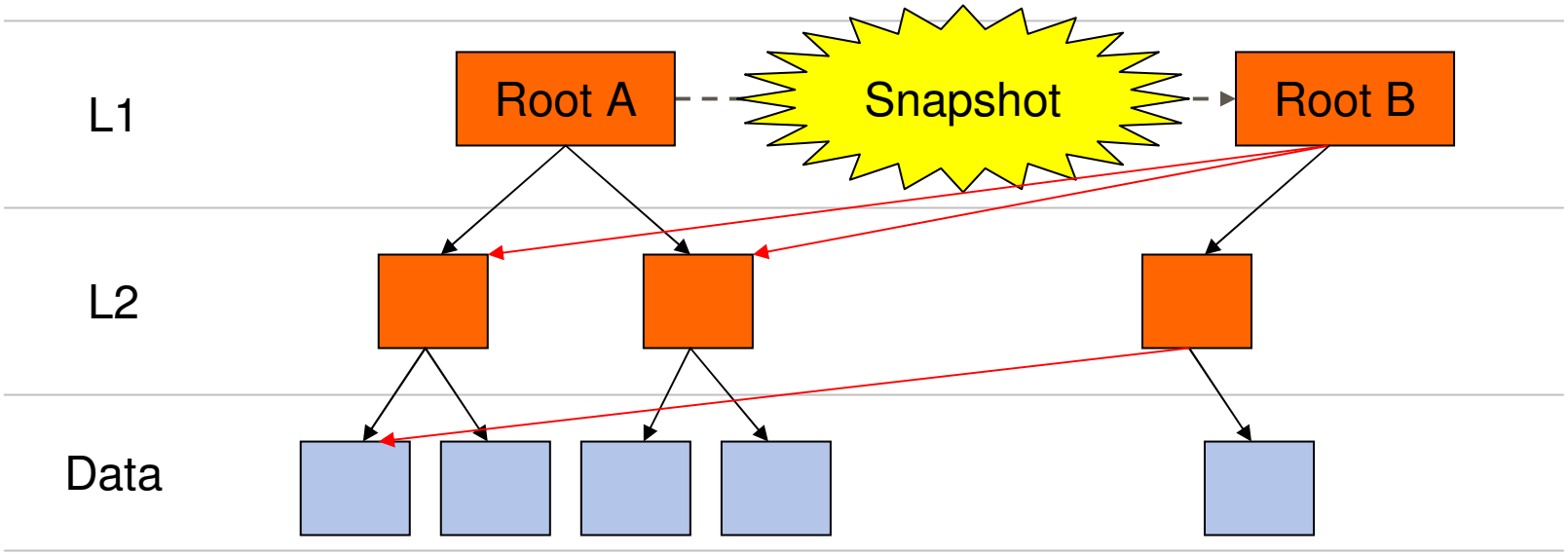
75	0.0298	domain1-apps	domain1-apps	(no symbols)
69	0.0274	domain1-kernel	domain1-kernel	ns_to_timespec
69	0.0274	vmlinux-syms-2.6.16.13-xen0	vmlinux-syms-2.6.16.13-xen0	delay_tsc
68	0.0270	domain1-kernel	domain1-kernel	do_IRQ
66	0.0262	oprofiled	oprofiled	odb_insert

Xen Research Projects

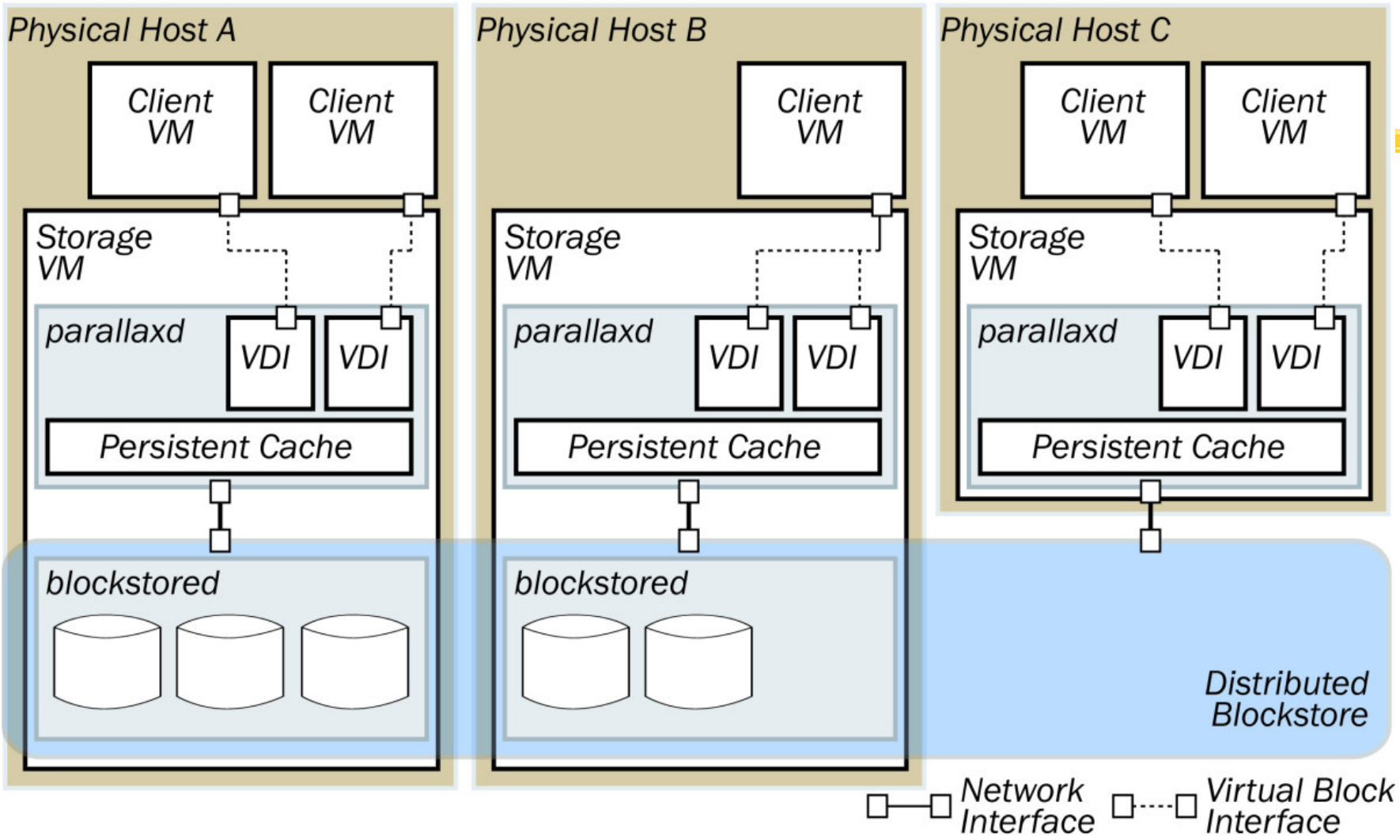
- Whole-system pervasive debugging
  - Lightweight checkpointing and replay
  - Cluster/distributed system debugging
- Software implemented h/w fault tolerance
  - Exploit deterministic replay
  - Explore possibilities for replay on SMP systems
- Multi-level secure systems with Xen
  - XenSE/OpenTC : Cambridge, Intel, GCHQ, HP, …
- VM forking
  - Lightweight service replication, isolation
  - UCSD Potemkin honeyfarm project

Parallax

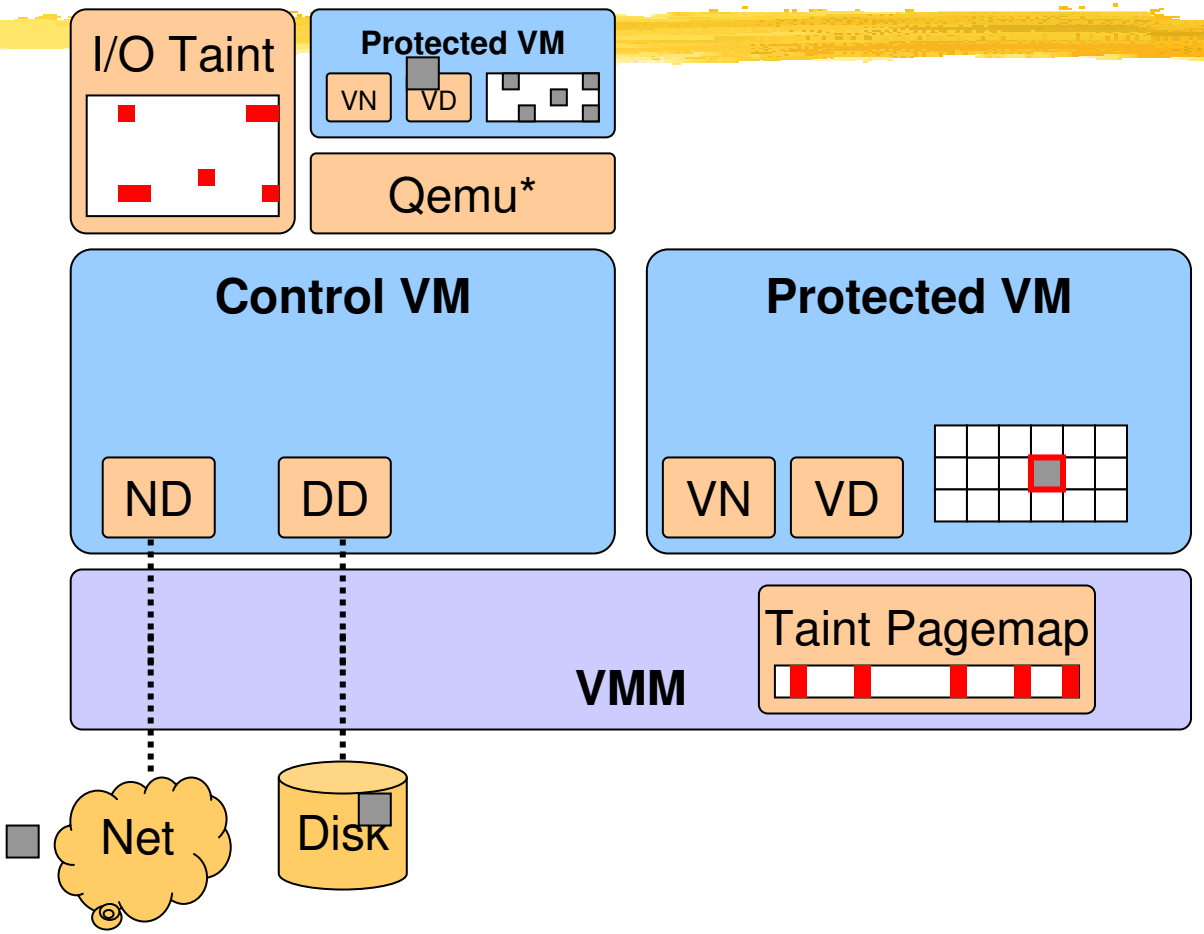
- *Managing storage in VM clusters.*
- Virtualizes storage, fast snapshots
- Access optimized storage



Parallax Architecture

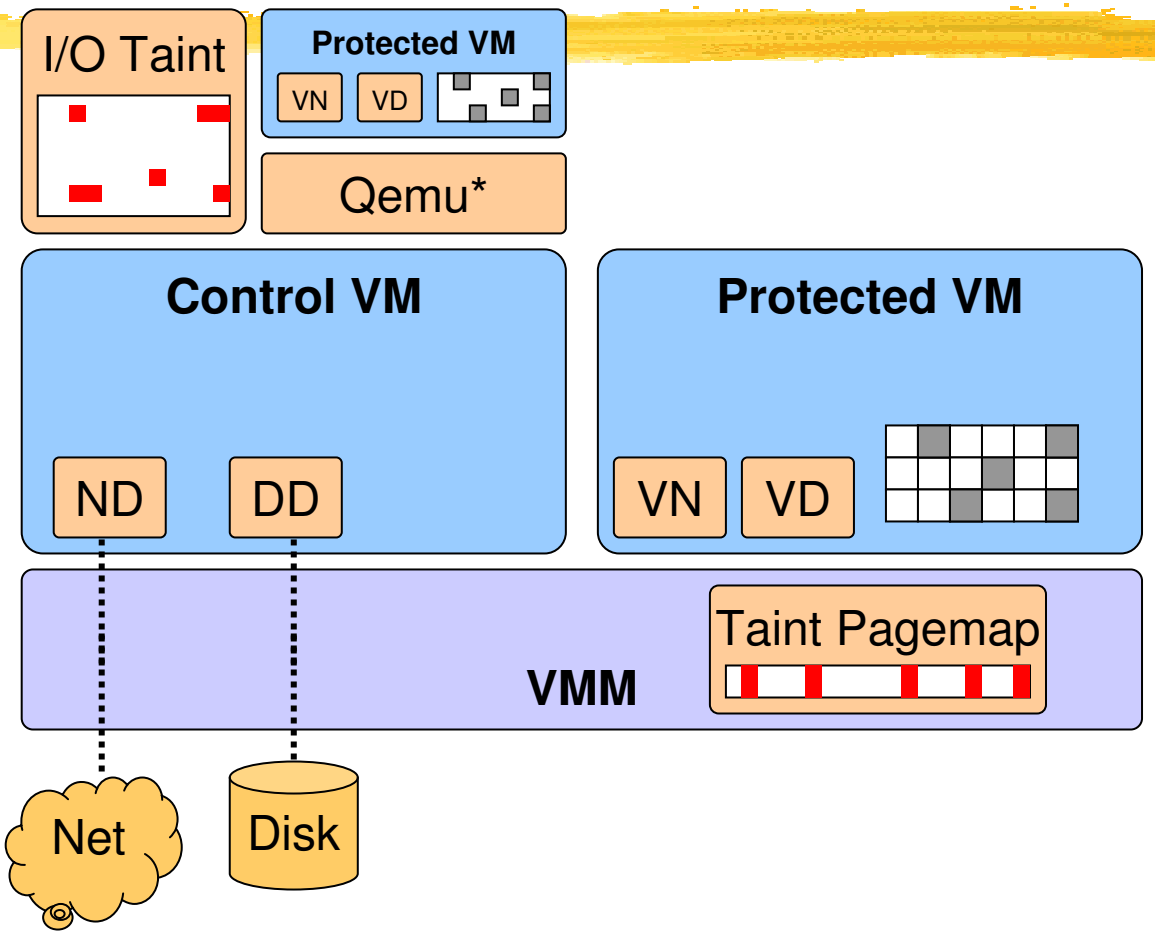


V2E : Taint tracking



4. Taint markings are propagated to disk. Disk extension marks tainted data, and re-taints memory on read.

V2E : Taint tracking



4. Taint markings are propagated to disk. Disk extension marks tainted data, and re-taints memory on read.

Current Xen Status

	x86_32	x86_32p	x86_64	IA64	Power
Privileged Domains					
Guest Domains					
SMP Guests					
Save/Restore/Migrate					
>4GB memory					
Progressive PV					
Driver Domains					

Post-3.0.0 Rough Code Stats

	aliases	checkins	insertions
xensource.com	16	1281	363449
ibm.com	30	271	40928
intel.com	26	290	29545
hp.com	8	126	19275
novell.com	8	78	17108
valinux.co.jp	3	156	12143
bull.net	1	145	11926
ncsc.mil	3	25	6048
fujitsu.com	13	119	6442
redhat.com	7	68	4822
amd.com	5	61	2671
virtualiron.com	5	23	1434
cam.ac.uk	1	9	1211
sun.com	2	9	826
unisys.com	3	7	857
other	30	189	48132

Stats since
3.0.0 Release

Xen Development Roadmap

- Performance tuning and optimization
  - Particularly for HVM and x86_64
- Enhanced control stack
- More automated system tuning
- Scalability and NUMA optimizations
- Better laptop/desktop support
  - OpenGL virtualization, power management
- Network optimziations

Conclusions

- Xen is a complete and robust hypervisor
- Outstanding performance and scalability
- Excellent resource control and protection
- Vibrant development community
- Strong vendor support

- Try the Xen demo CD to find out more!
  (or Fedora Core 6, Suse 10.x)

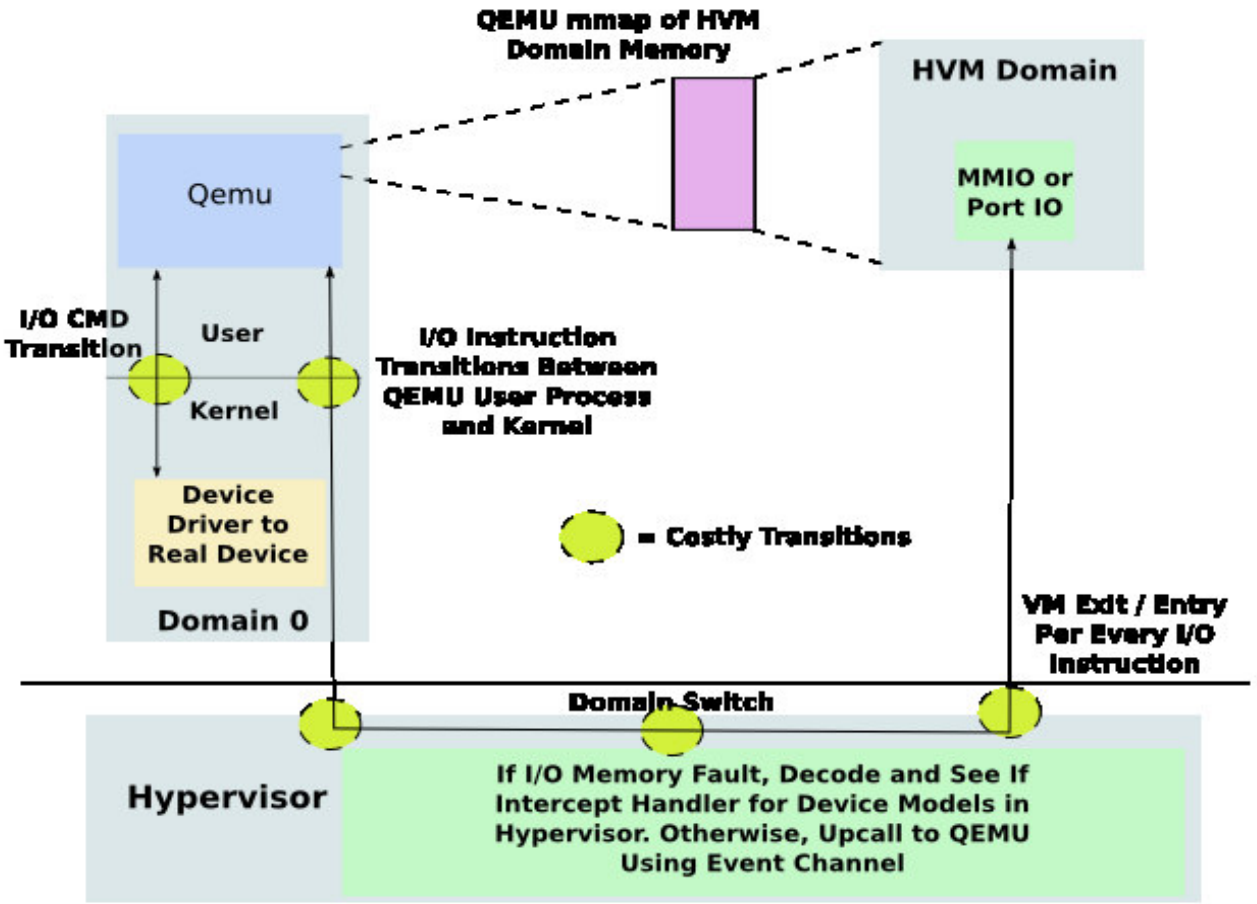
- http://xensource.com/community

Thanks!

- If you're interested in working on Xen we're looking for Research Assistants and Associates in the Lab, and also folk to work in the XenSource Cambridge office.

- ian.pratt@cl.cam.ac.uk

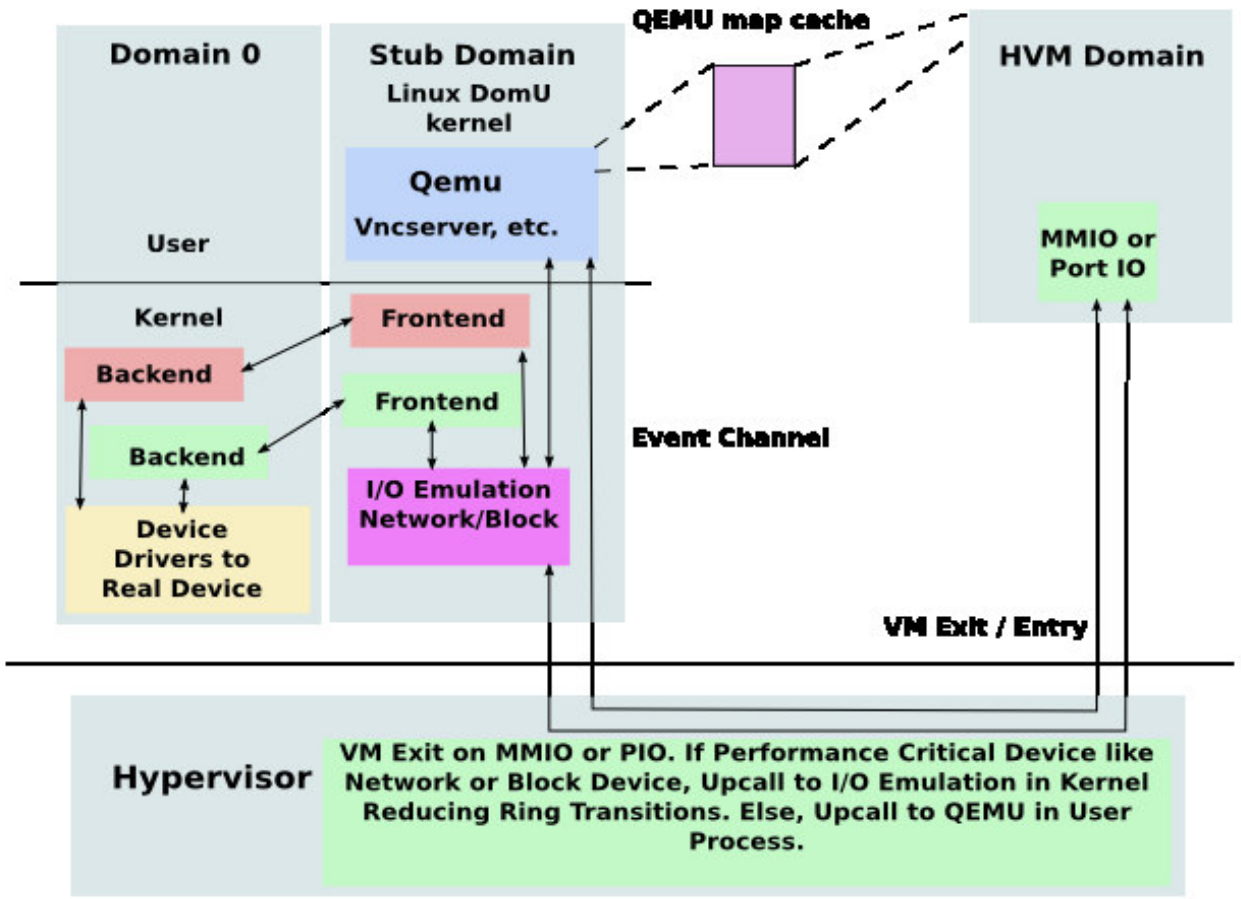
Current Device Model



Current Device Model

- QEMU Device Emulation
  - Runs in Domain0 as User Process
  - No Isolation to Properly Accounting for HVM I/O
  - Doesn't Scale – Launch New QEMU-DM For Every HVM Domain, Currently Maps All HVM Memory
  - Many Costly Ring Transitions
    - On Every I/O Instruction Event Between QEMU and Kernel
    - On Every I/O Command Between QEMU and Real Device Drivers
  - Many Costly VM Exits for Every I/O

Stub Domain


QEMU map cache
Domain 0
Stub Domain
Linux DomU
kernel
HVM Domain
Qemu
Vncserver, etc.
User
MMIO or
Port IO
Kernel
Frontend
Backend
Frontend
Backend
Event Channel
I/O Emulation
Network/Block
Device
Drivers to
Real Device
VM Exit / Entry
Hypervisor
VM Exit on MMIO or PIO. If Performance Critical Device like
Network or Block Device, Upcall to I/O Emulation in Kernel
Reducing Ring Transitions. Else, Upcall to QEMU in User
Process.

Stub Domain Requirements

- Requirements
  - Need User Space Context for QEMU
    - Some Devices Require User Process - vncserver
  - Need Library Support for QEMU
  - Need SMP Support
  - Need to Run IO Emulation in Kernel Space
  - Need to Run Frontend Drivers
  - Need to Limit Impact to Build
  - Need to Limit QEMU Maintenance Impact