

Virtualization Technology

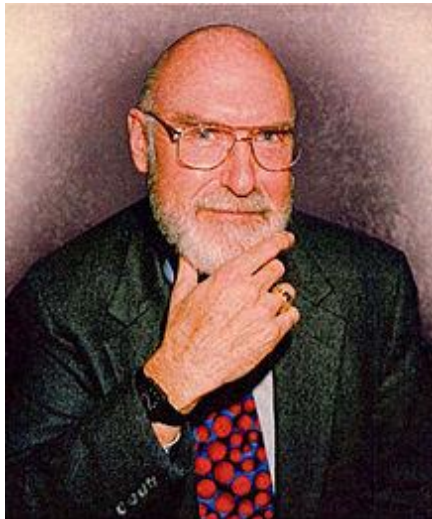
Zhiming Shen

Virtualization: rejuvenation

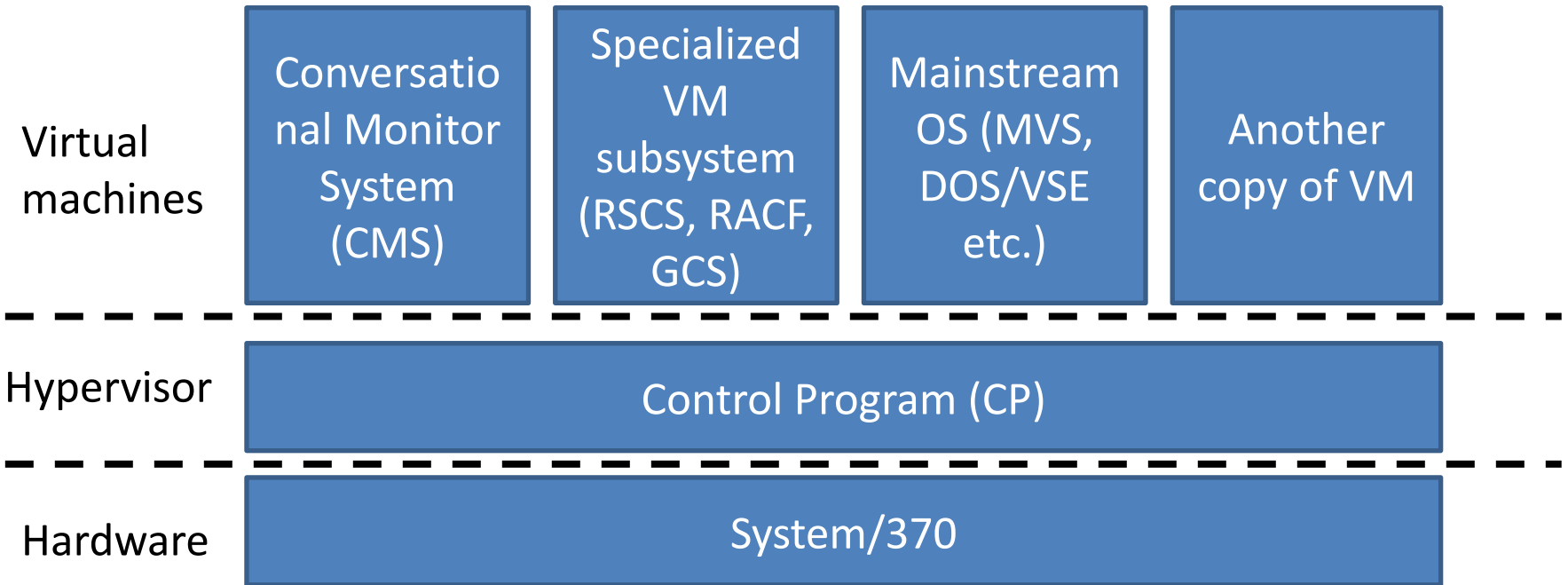
- 1960's: first track of virtualization
 - Time and resource sharing on expensive mainframes
 - IBM VM/370
- Late 1970's and early 1980's: became unpopular
 - Cheap hardware and multiprocessing OS
- Late 1990's: became popular again
 - Wide variety of OS and hardware configurations
 - VMWare
- Since 2000: hot and important
 - Cloud computing

IBM VM/370

- Robert Jay Creasy (1939-2005)
 - Project leader of the first full virtualization hypervisor: IBM CP-40, a core component in the VM system
 - The first VM system: VM/370

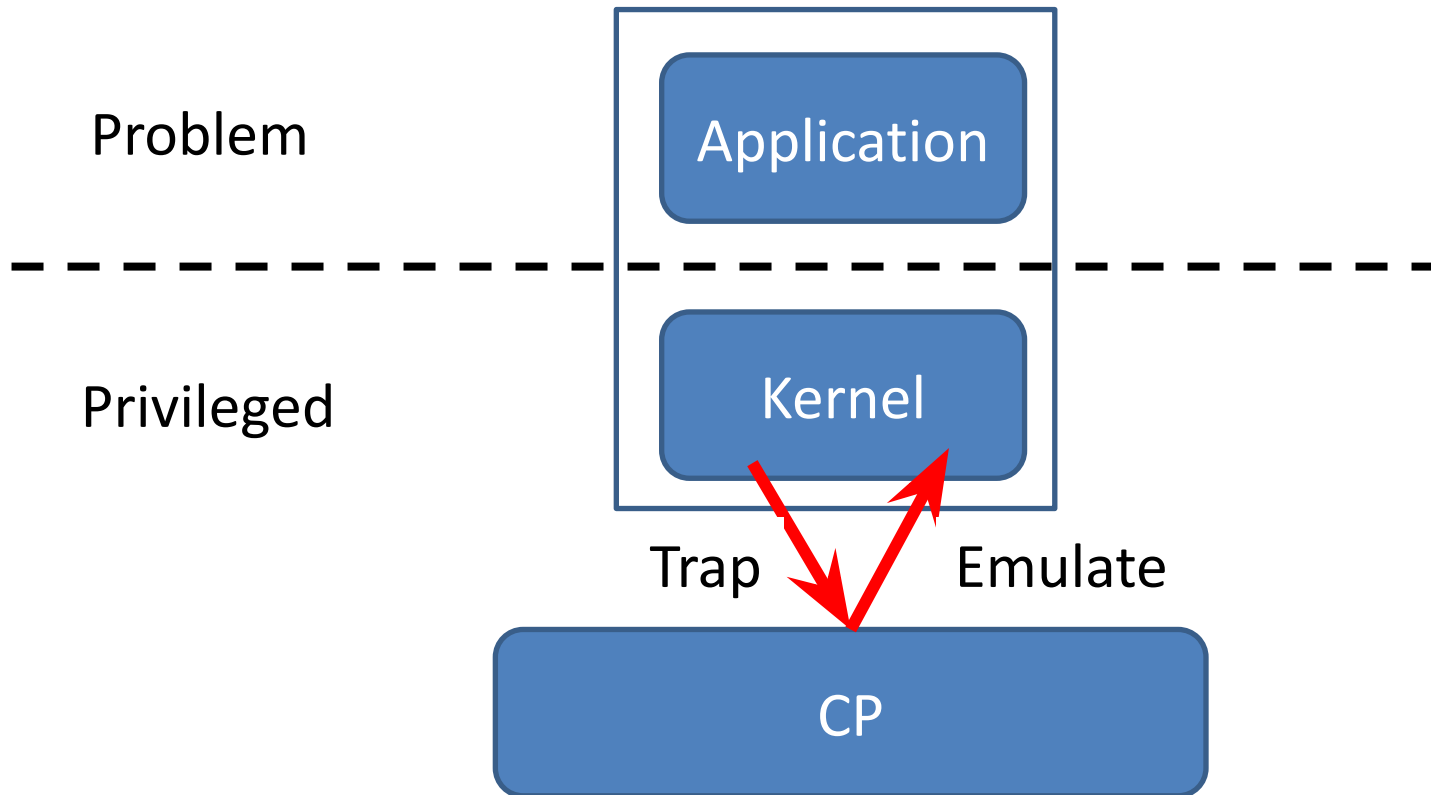


IBM VM/370



IBM VM/370

- Technology: trap-and-emulate



Virtualization on x86 architecture

- Challenges
 - Correctness: not all privileged instructions produce traps!
 - Example: popf
 - Performance:
 - System calls: traps in both enter and exit (10X)
 - I/O performance: high CPU overhead
 - Virtual memory: no software-controlled TLB

Virtualization on x86 architecture

- Solutions:
 - Dynamic binary translation & shadow page table
 - Hardware extension
 - Para-virtualization (Xen)

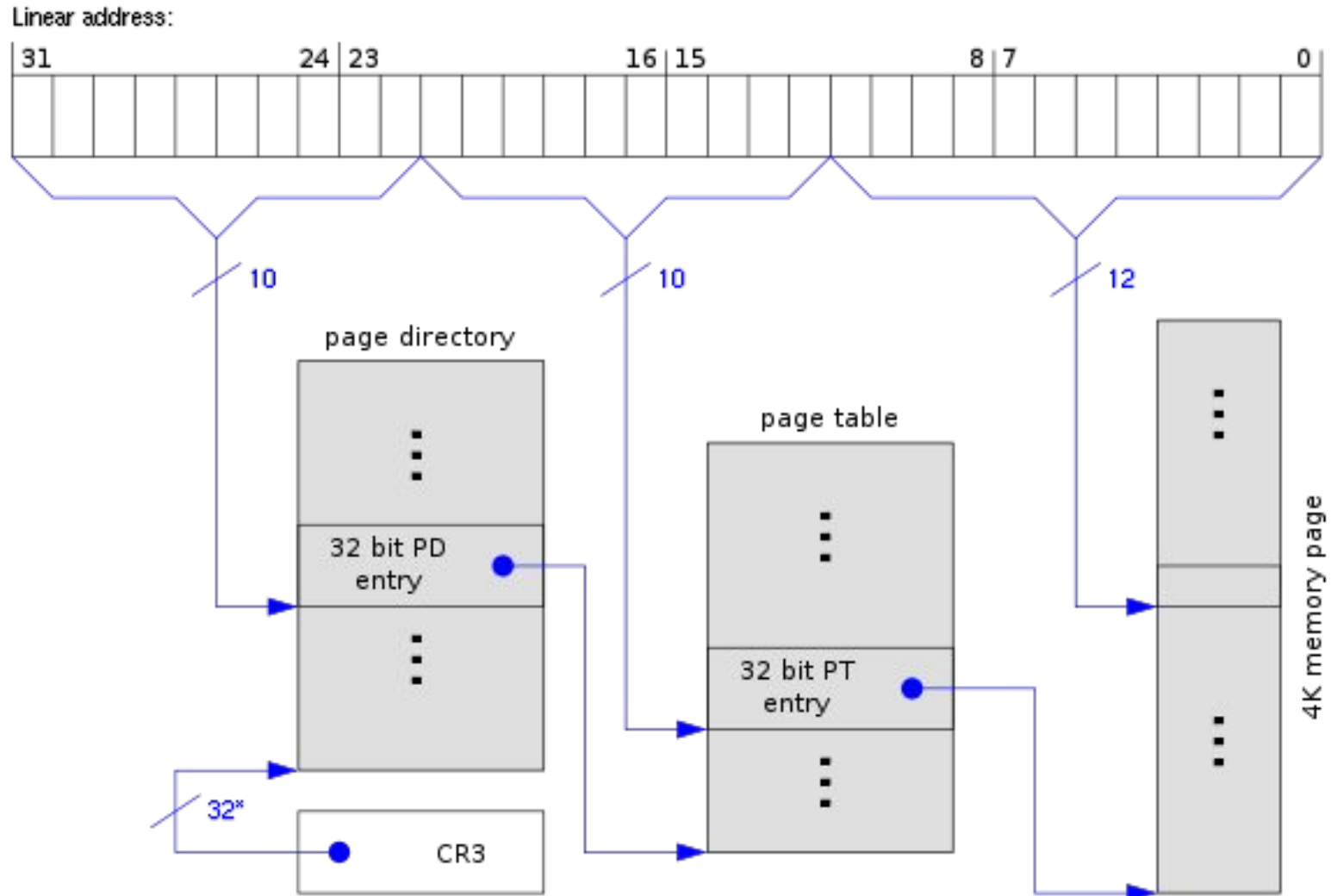
Dynamic binary translation

- Idea: intercept privileged instructions by changing the binary
- Cannot patch the guest kernel directly (would be visible to guests)
- Solution: make a copy, change it, and execute it from there
 - Use a cache to improve the performance

Dynamic binary translation

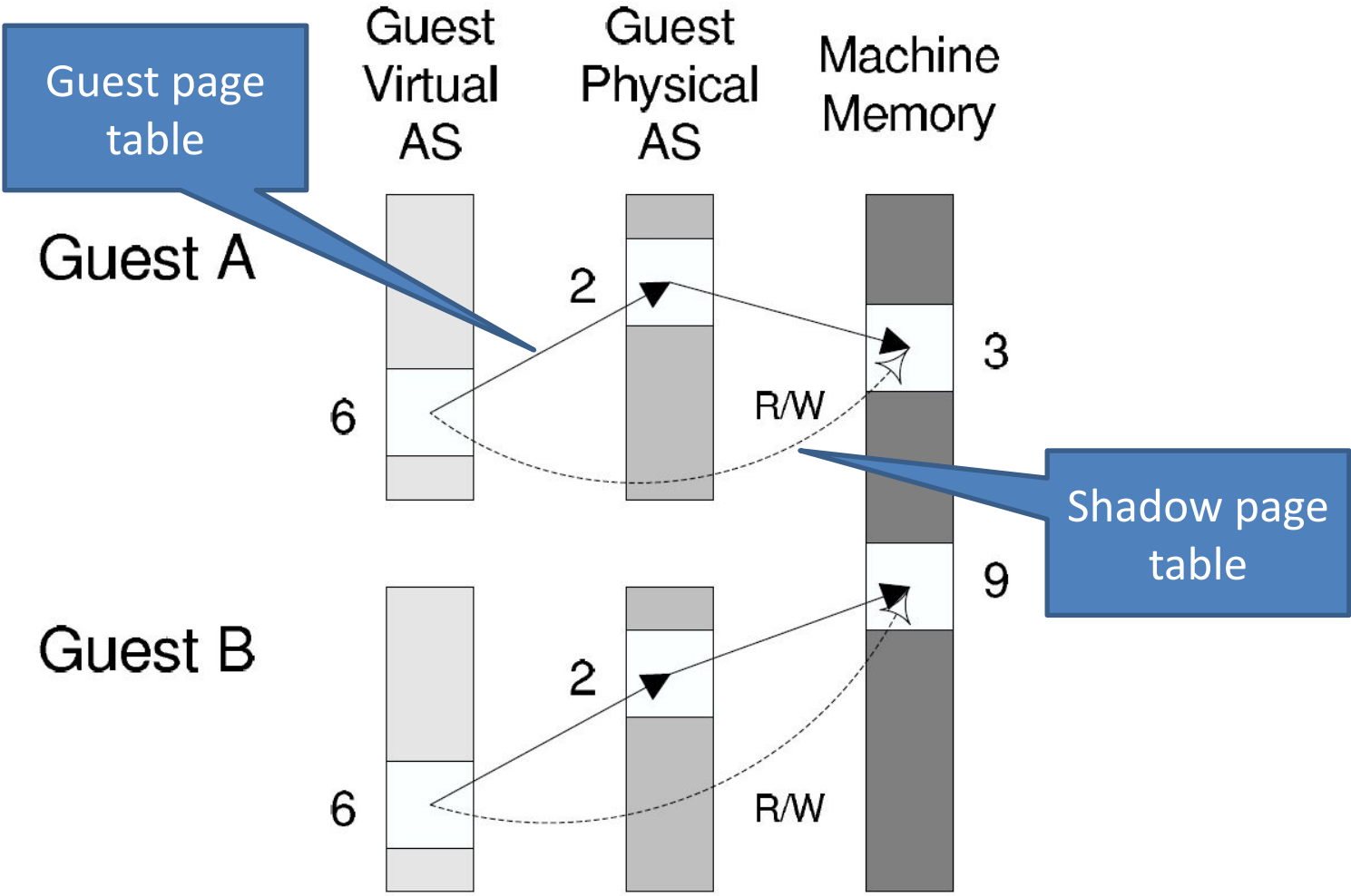
- Pros:
 - Make x86 virtualizable
 - Can reduce traps
- Cons:
 - Overhead
 - Hard to improve system calls, I/O operations
 - Hard to handle complex code

Shadow page table



*) 32 bits aligned to a 4-KByte boundary

Shadow page table



Shadow page table

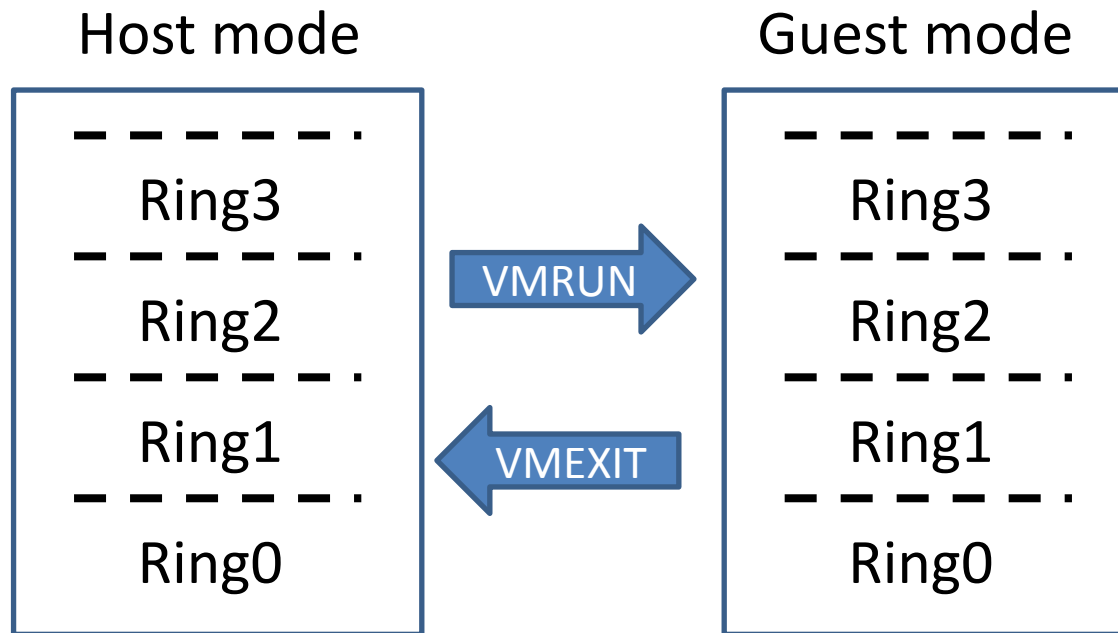
- Pros:
 - Transparent to guest VMs
 - Good performance when working set fit into shadow page table
- Cons:
 - Big overhead of keeping two page tables consistent
 - Introducing more issues: hidden fault, double paging ...

Hardware support

- First generation - processor
- Second generation - memory
- Third generation – I/O device

First generation: Intel VT-x & AMD SVM

- Eliminating the need of binary translation



Second generation: Intel EPT & AMD NPT


- Eliminating the need to shadow page table

Future Extensions: EPT

EPT: Overview

The diagram illustrates the EPT translation process. It starts with a 'Guest Linear Address' on the left. An arrow points from this address to a box labeled 'Intel® 64 Page Tables'. A 'CR3' label is positioned above this box, with a line connecting it to the top of the box. From the 'Intel® 64 Page Tables' box, an arrow labeled 'Guest Physical Address' points to a second box labeled 'EPT Page Tables'. Above this second box is the label 'EPT Base Pointer', with a line connecting it to the top of the box. Finally, an arrow points from the 'EPT Page Tables' box to the 'Host Physical Address' on the right.

- Intel® 64 page tables
 - Map **guest-linear** to **guest-physical** (translated again)
 - Can be read and written by guest
- New EPT page tables under VMM control
 - Map **guest-physical** to **host-physical** (accesses memory)
 - Referenced by new **EPT base pointer**
- No VM exits due to **page faults**, **INVLPG**, or **CR3** accesses



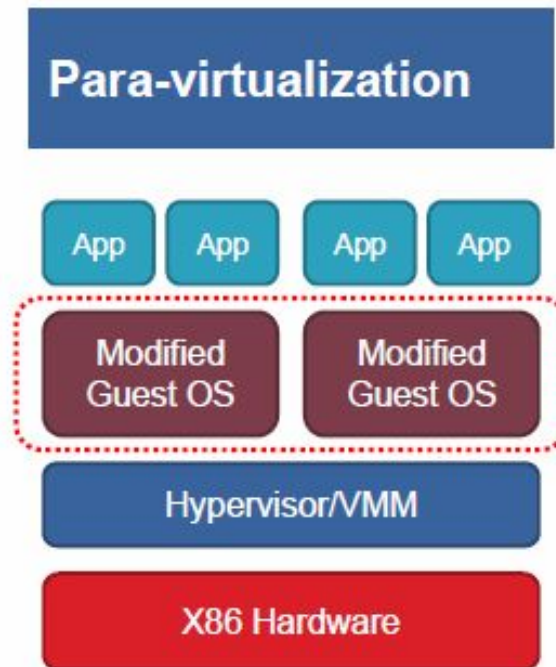
8

Third generation: Intel VT-d & AMD IOMMU

- I/O device assignment
 - VM owns real device
- DMA remapping
 - Support address translation for DMA
- Interrupt remapping
 - Routing device interrupt

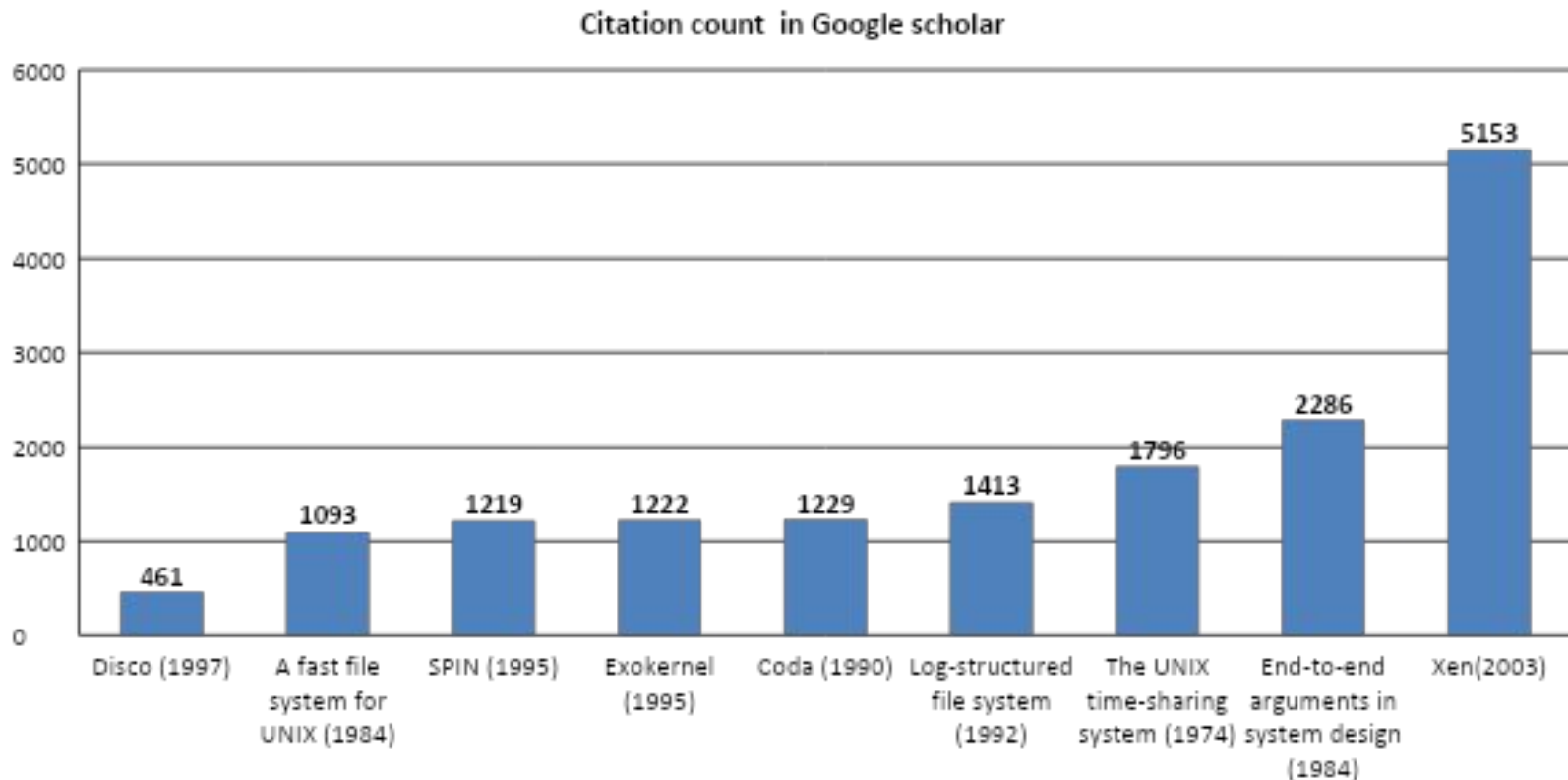
Para-virtualization

- Full vs. para virtualization



Xen and the art of virtualization

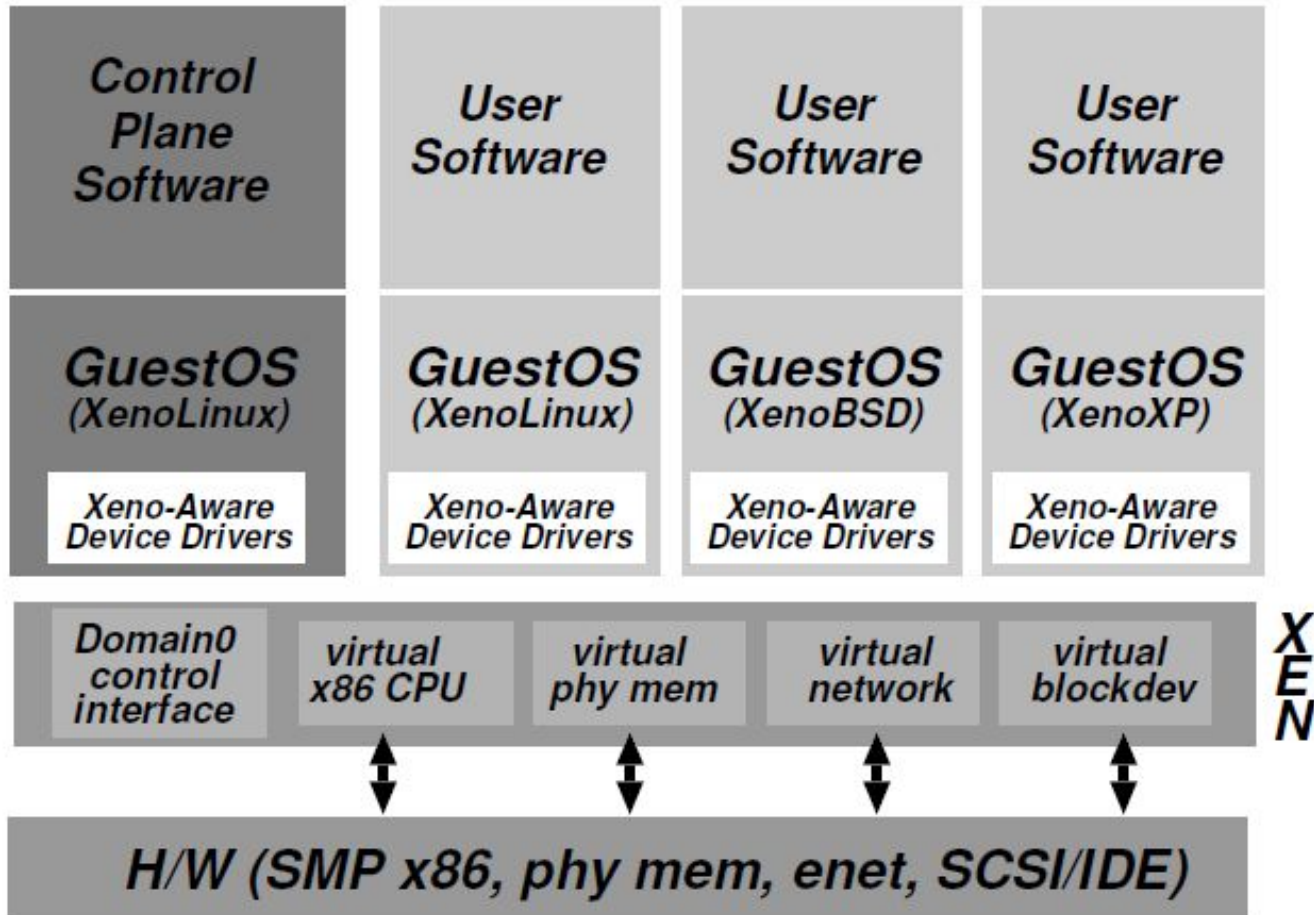
- SOSP'03
- Very high impact



Overview of the Xen approach

- Support for unmodified application binaries (but not OS)
 - Keep Application Binary Interface (ABI)
- Modify guest OS to be aware of virtualization
 - Get around issues of x86 architecture
 - Better performance
- Keep hypervisor as small as possible
 - Device driver is in Dom0

Xen architecture



Virtualization on x86 architecture

- Challenges
 - Correctness: not all privileged instructions produce traps!
 - Example: popf
 - Performance:
 - System calls: traps in both enter and exit (10X)
 - I/O performance: high CPU overhead
 - Virtual memory: no software-controlled TLB

CPU virtualization

- Protection
 - Xen in ring0, guest kernel in ring1
 - Privileged instructions are replaced with hypercalls
- Exception and system calls
 - Guest OS registers handles validated by Xen
 - Allowing direct system call from app into guest OS
 - Page fault: redirected by Xen

CPU virtualization (cont.)

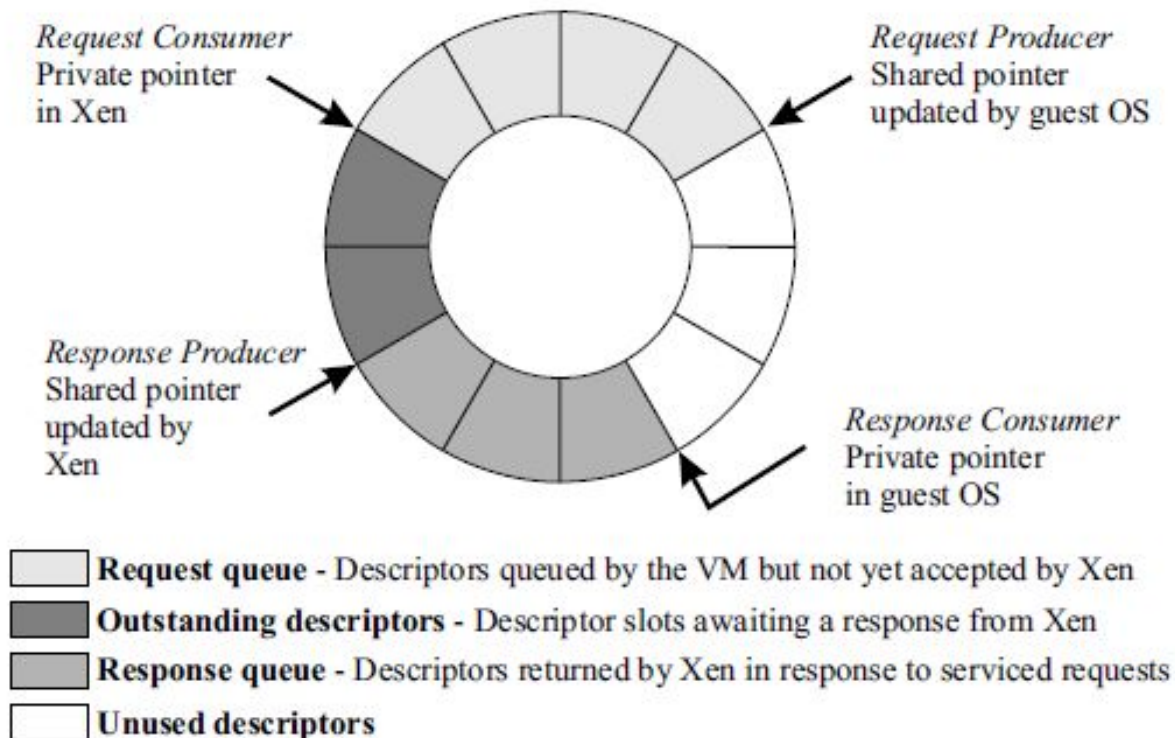
- Interrupts:
 - Lightweight event system
- Time:
 - Interfaces for both real and virtual time

Memory virtualization

- Xen exists in a 64MB section at the top of every address space
- Guest sees real physical address
- Guest kernels are responsible for allocating and managing the hardware page tables.
- After registering the page table to Xen, all subsequent updates must be validated.

I/O virtualization

- Shared-memory, asynchronous buffer descriptor rings



Porting effort

OS subsection	# lines	
	Linux	XP
Architecture-independent	78	1299
Virtual network driver	484	–
Virtual block-device driver	1070	–
Xen-specific (non-driver)	1363	3321
Total	2995	4620
(Portion of total x86 code base	1.36%	0.04%)

Table 2: The simplicity of porting commodity OSes to Xen. The cost metric is the number of lines of reasonably commented and formatted code which are modified or added compared with the original x86 code base (excluding device drivers).

Evaluation

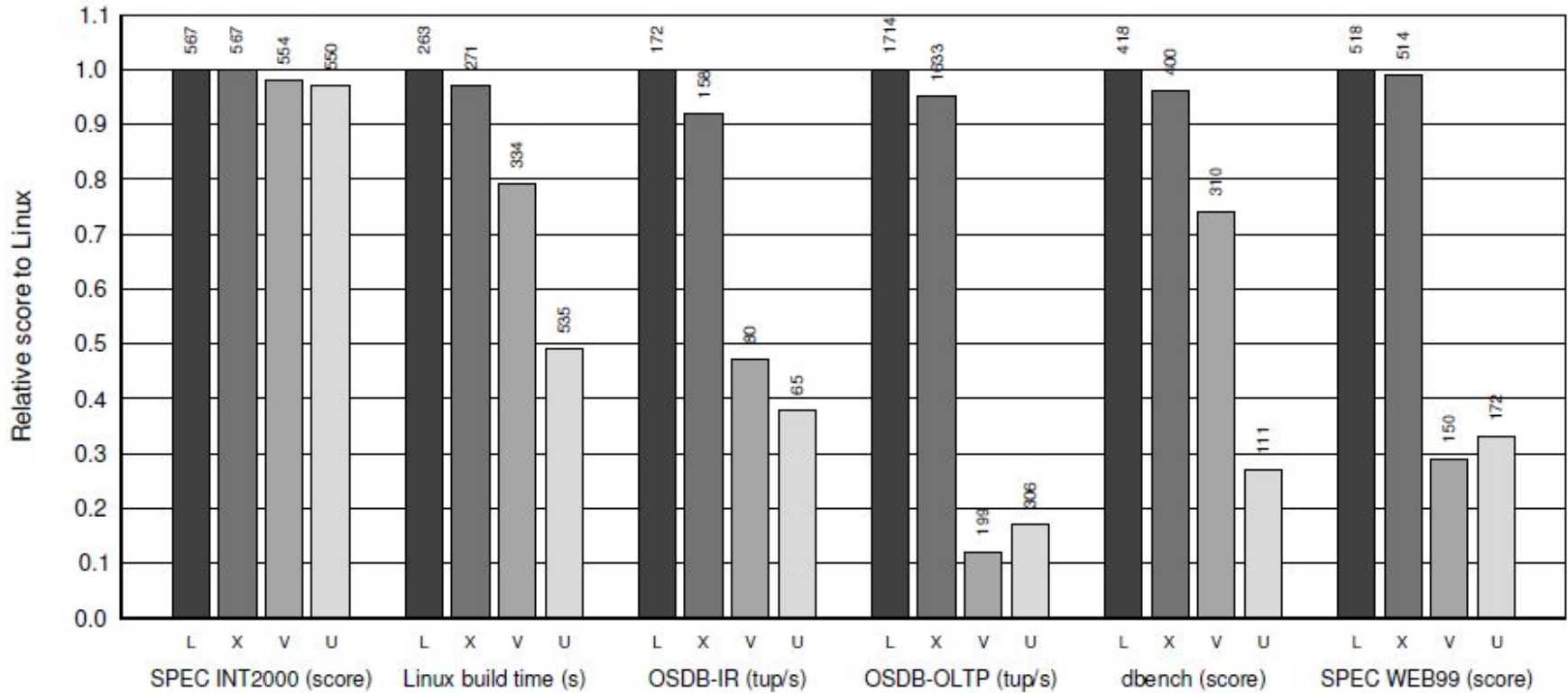


Figure 3: Relative performance of native Linux (L), XenLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

Evaluation

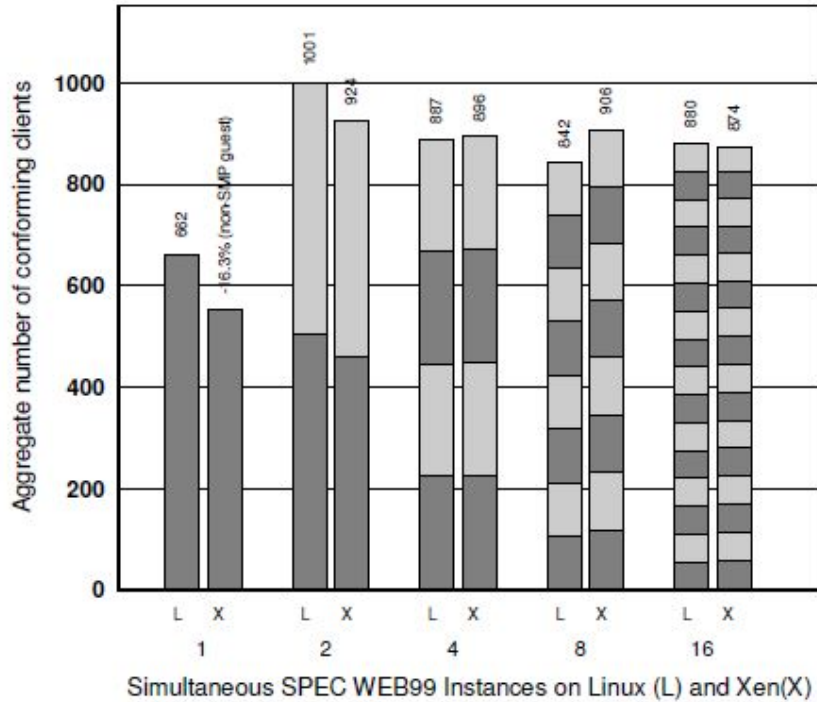


Figure 4: SPEC WEB99 for 1, 2, 4, 8 and 16 concurrent Apache servers: higher values are better.

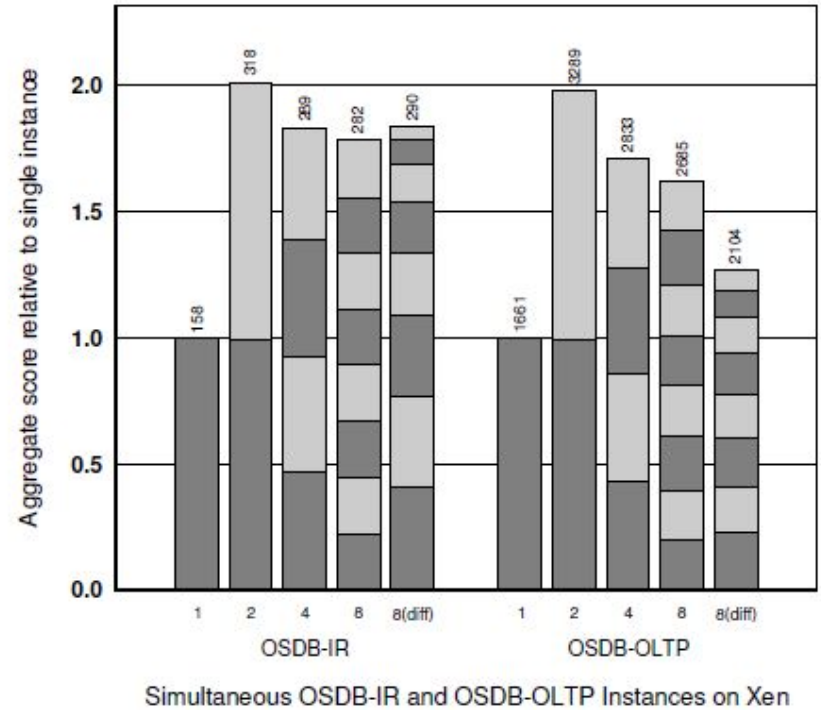


Figure 5: Performance of multiple instances of PostgreSQL running OSDB in separate Xen domains. 8(diff) bars show performance variation with different scheduler weights.

Evaluation

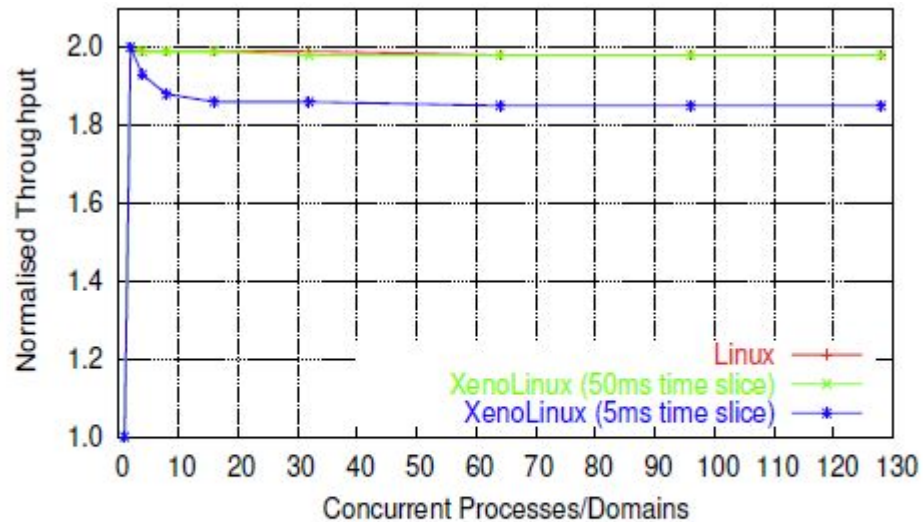


Figure 6: Normalized aggregate performance of a subset of SPEC CINT2000 running concurrently on 1-128 domains

Conclusion

- x86 architecture makes virtualization challenging
- Full virtualization
 - unmodified guest OS; good isolation
 - Performance issue (especially I/O)
- Para virtualization:
 - Better performance (potentially)
 - Need to update guest kernel
- Full and para virtualization will keep evolving together