# CS 202
# Advanced Operating Systems

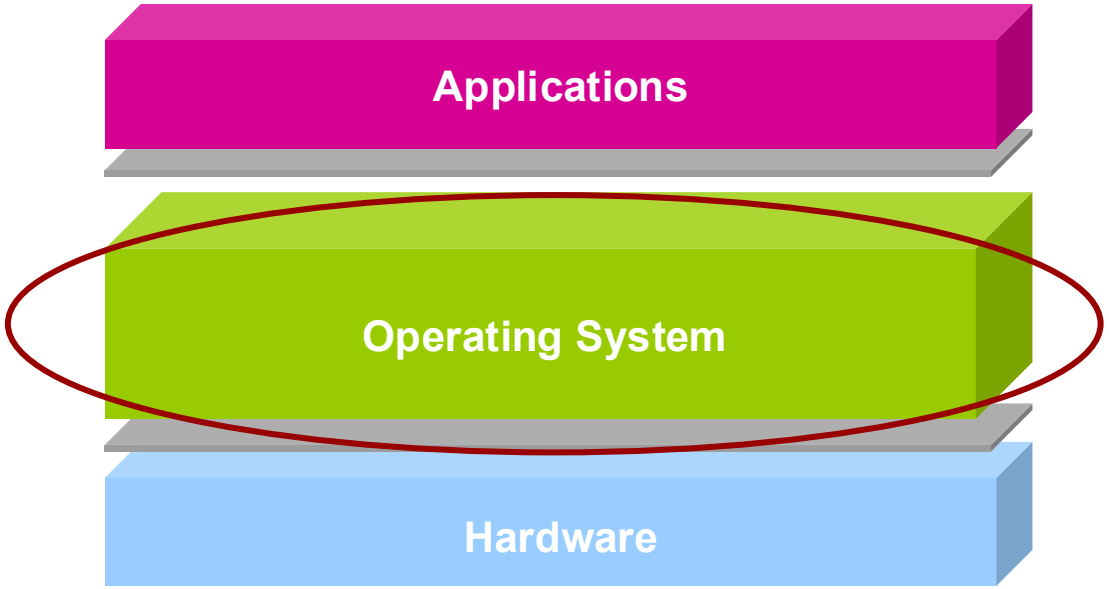
## Winter 26

Lecture 3: OS Architecture

Instructor: Chengyu Song

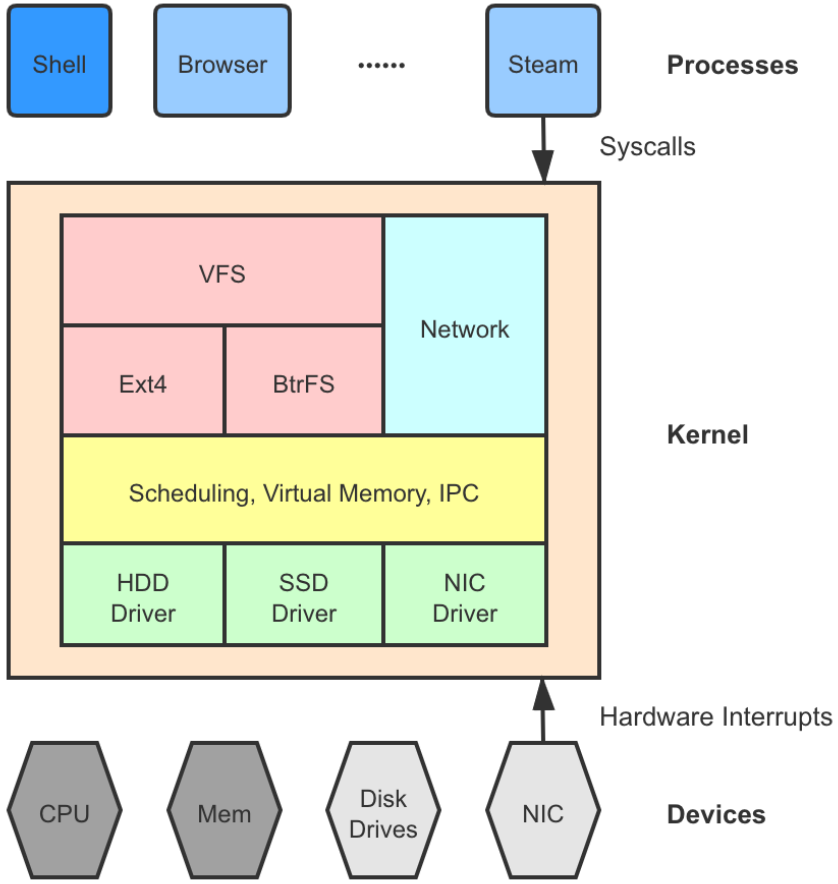
# What is OS architecture?

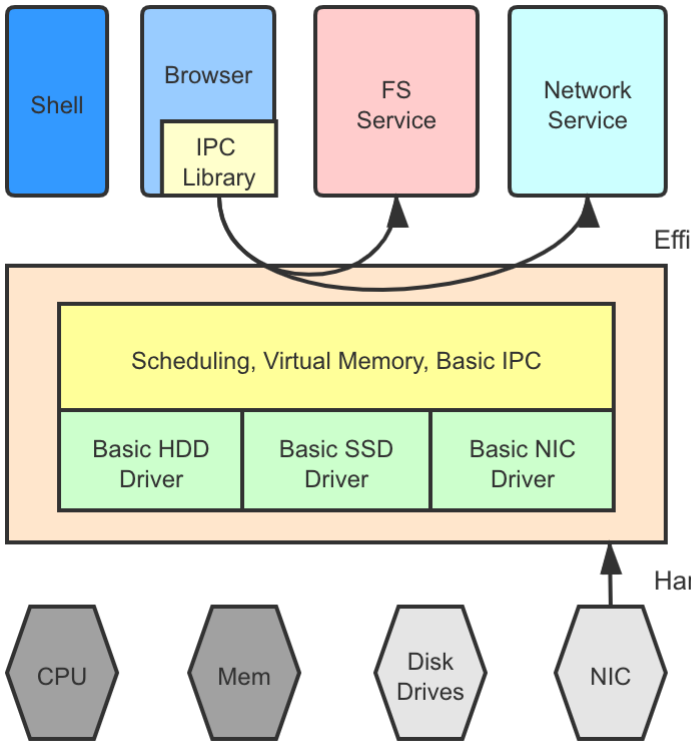
- The design and organization of the software layer that defines the interface between applications and physical resources

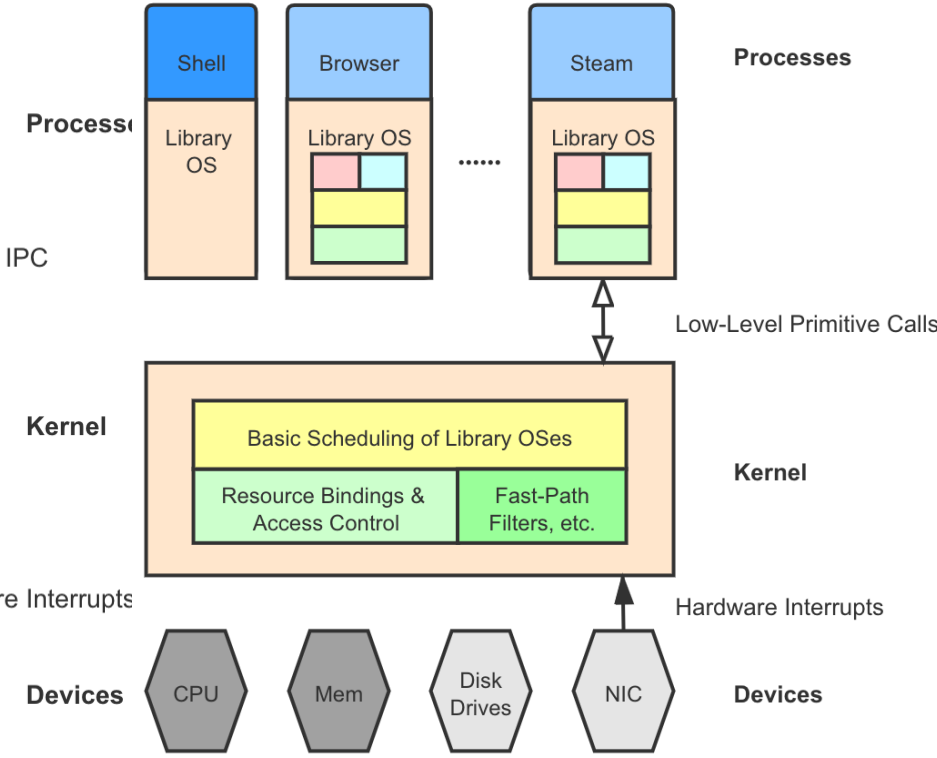


# Typical OS architectures

- Monolithic
  - UNIX, Linux, Windows
- Micro-kernel
  - L4, Mach, QNX
- LibraryOS
  - exokernel, graphene



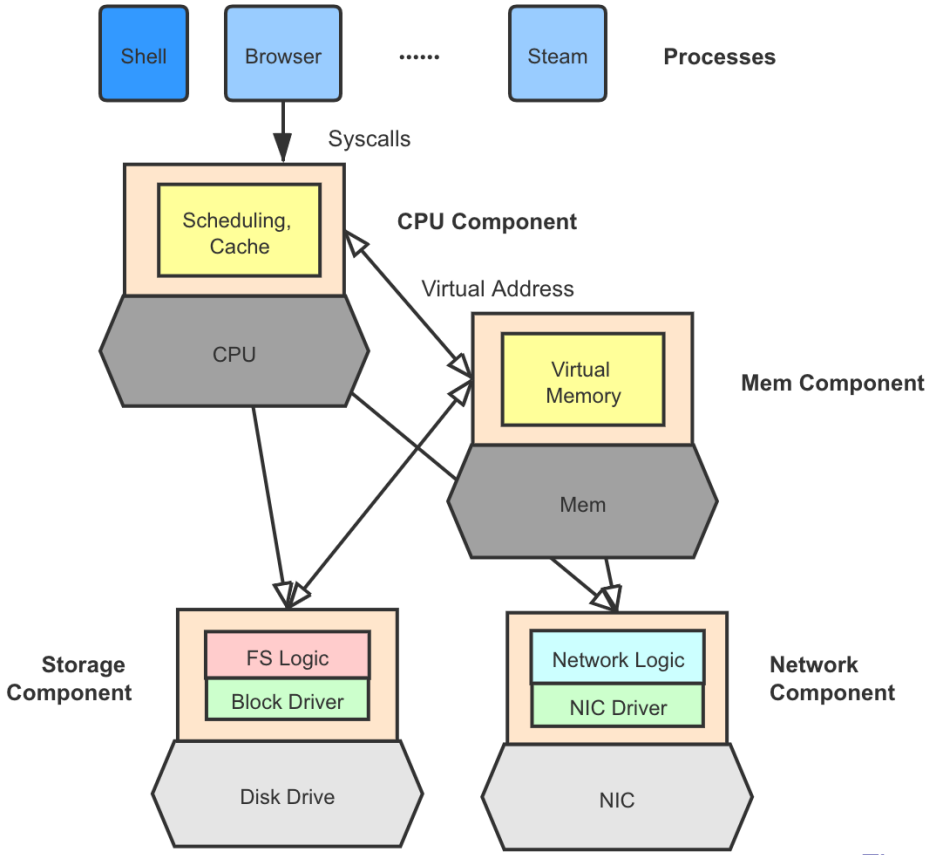



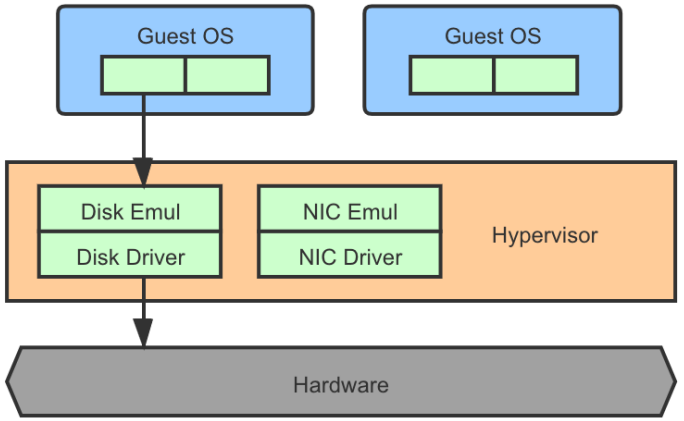

Figures from https://www.josehu.com/technical/2021/05/24/os-kernel-models.html

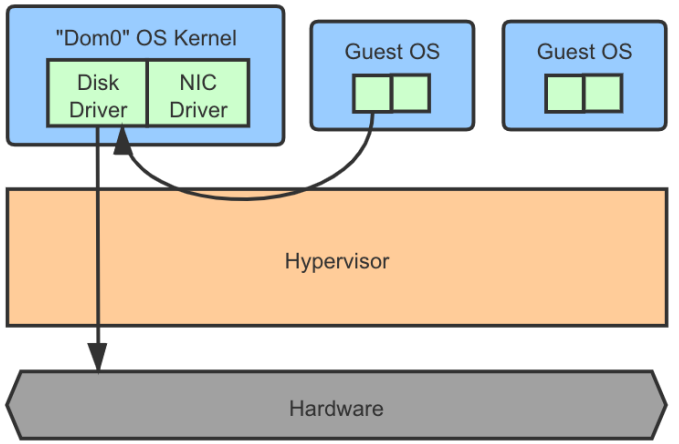
# Typical OS architectures

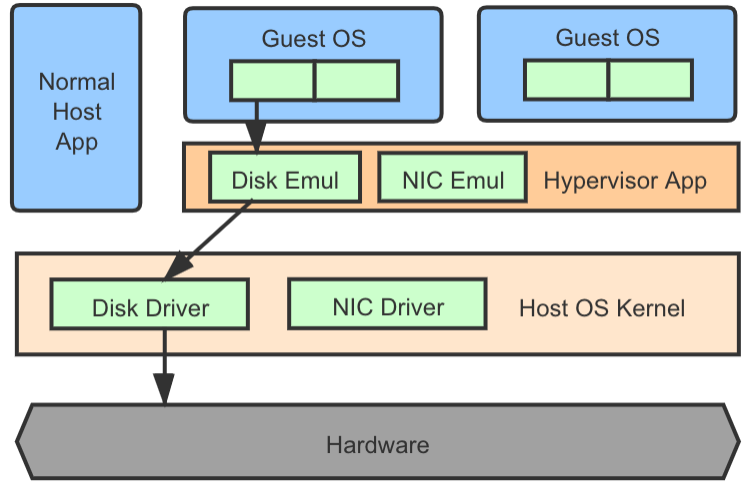
- Multi-kernels
  - Barrelfish, LegoOS




- Hypervisors
  - VMware ESX, Xen, KVM



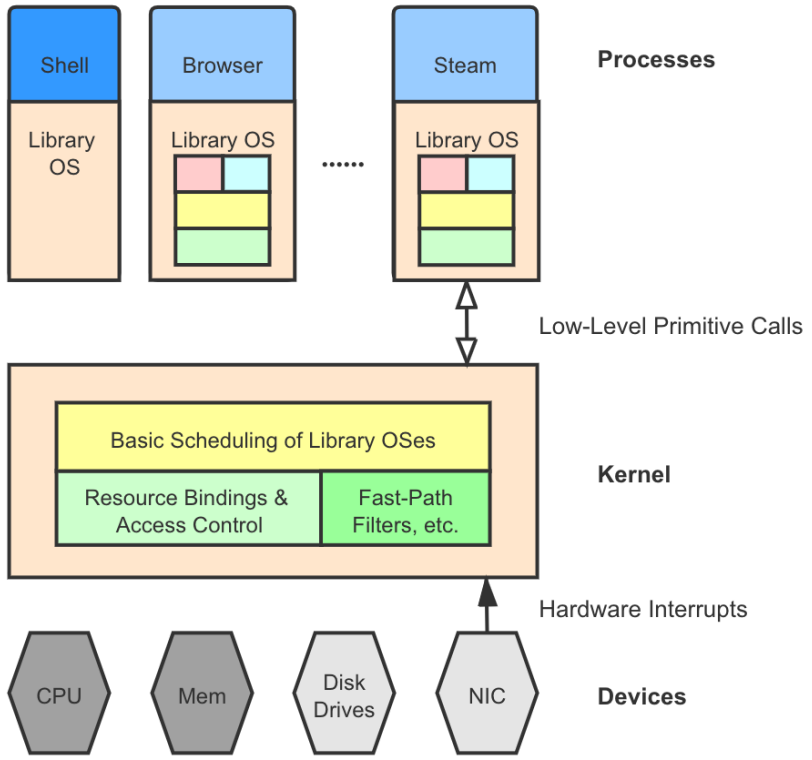





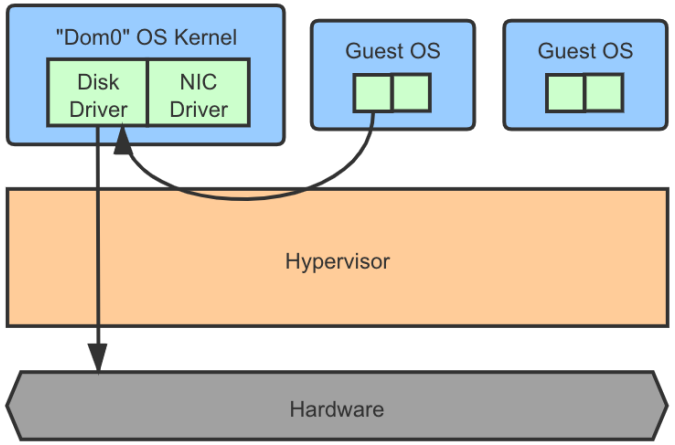

Figures from https://www.josehu.com/technical/2021/05/24/os-kernel-models.html

# Xen: paravirtualization

- Context
  - SOSP'03
  - x86
  - libOS?
  - μ-kernel?
  - hypervisor
- LibraryOS
  - exokernel, graphene




- Hypervisors
  - Xen



# Xen: paravritualization

- Motivations?
  - “Commodity” operations systems
  - Performance
  - Compatibility

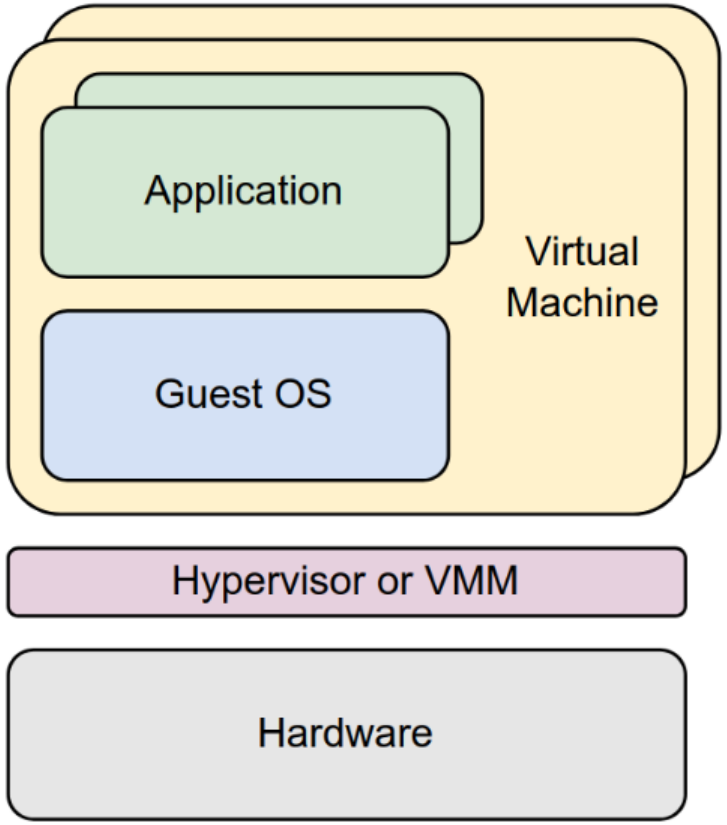
- Core idea?
  - Is it possible to convert commodity OS into a libOS without much modifications?
  - Is it possible to realize a thin layer of hardware multiplexing hypervisor on x86?

# Xen: x86 is a problem

- “Un-cooperative” design for full virtualization
  - Insufficient permission on supervisor instructions on VMM fail silently
  - Hardware-managed TLB, no tagging (high cost of flushing)
  - Complicated privilege model
    - 4 privilege levels (rings)
    - Ring 0: kernel mode
    - Ring 3: user mode
    - No clean way for the hypervisor to sit “below” the guest OS while maintaining the illusion that it’s running in ring 0.
    - Complex workarounds for full virtualization (binary translation, device emulation)

# Xen: cost of full virtualization

- Full virtualization: no modification to the guest OS
  - Full ABI compatibility
- Architectures like x86 are not natively virtualizable and require costly techniques, causing **huge performance overhead**
  - **Binary translation**: Hypervisor traps and emulates privileged instructions
  - Each VM gets a fully virtualized hardware environment
  - Needs hardware support or **heavy device emulation**



# Xen: paravirtualization

- Let's corporate, like in exokernel
  - Guest OS knows it's running in a virtualized environment
  - Guest OS asks hypervisor for help (vs. trap and emulation)
  - Guest OS uses virtual device drivers

- Trade off (hopefully) small changes to the guest OS for big improvements in performance and simplicity

# Xen: dicussion1

- What are the differences between virtualization and exokernels?
- Why was virtualization much more successful than exokernels?

# Xen: architecture

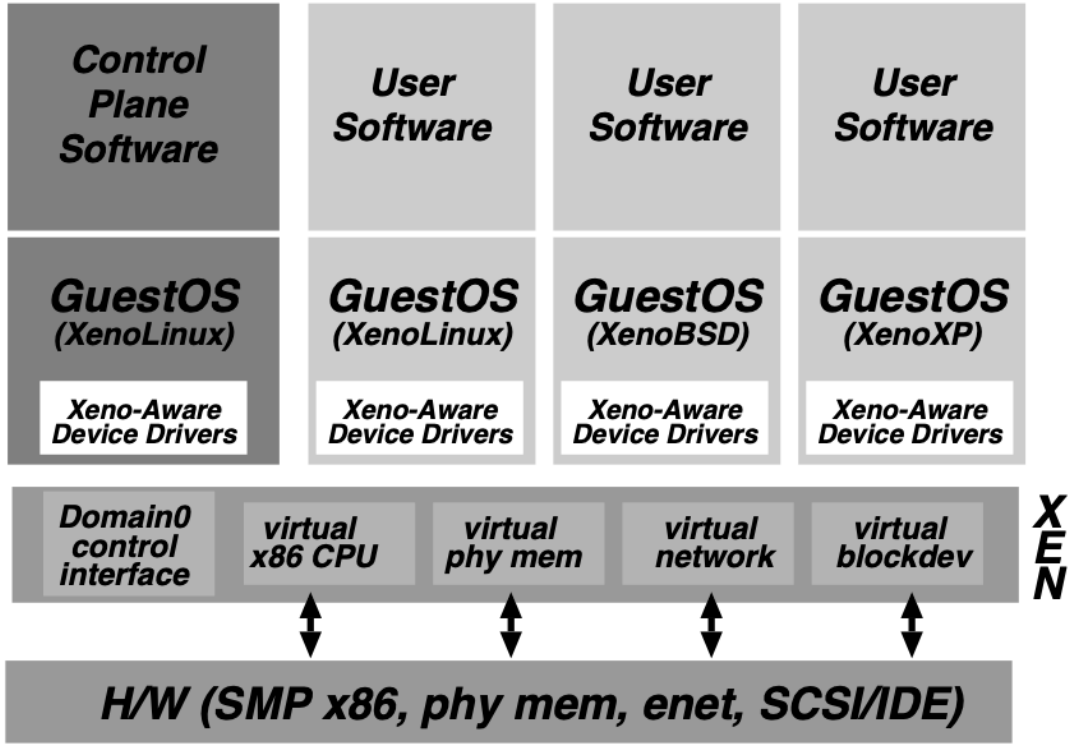


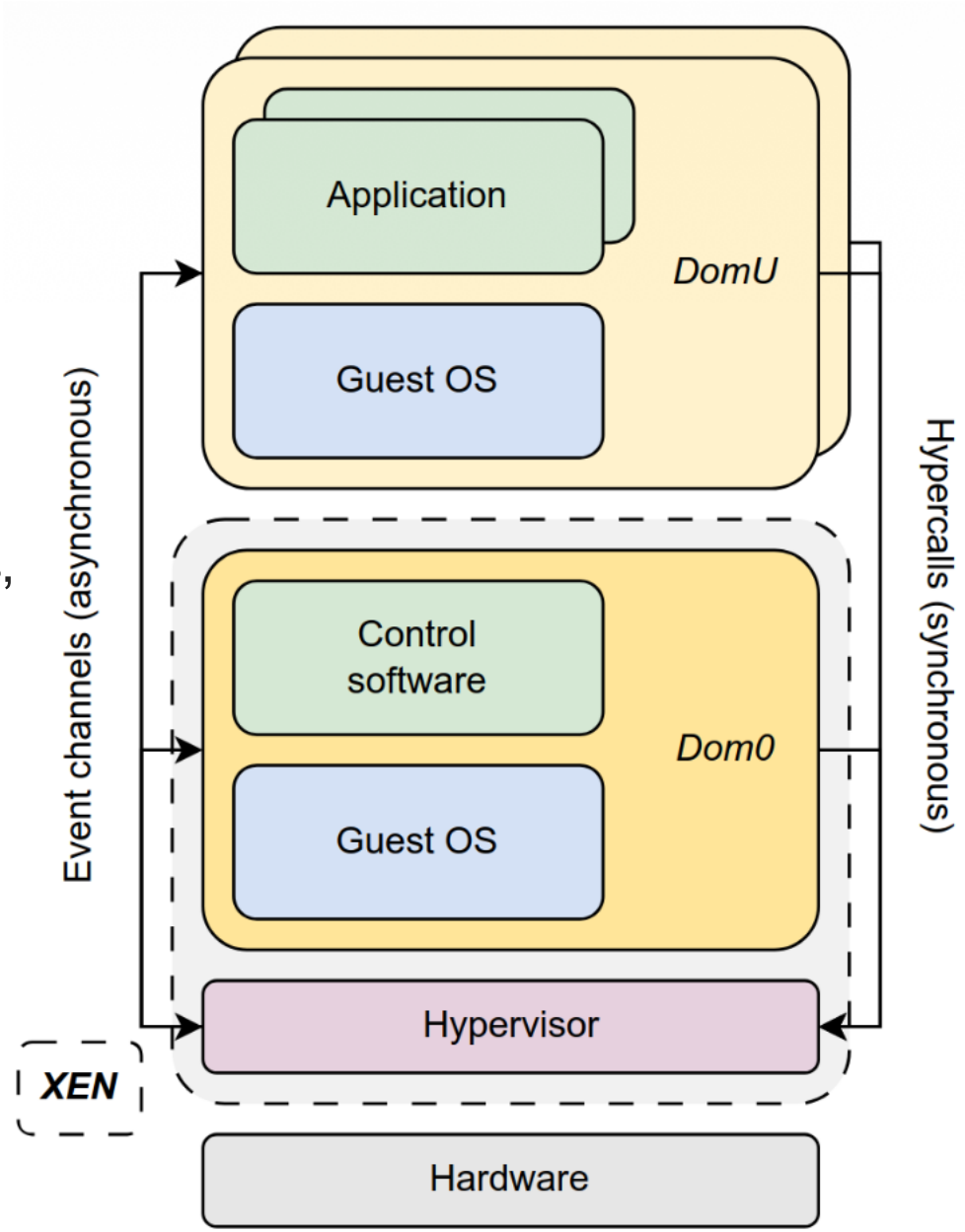

**Figure 1: The structure of a machine running the Xen hypervisor, hosting a number of different guest operating systems, including *Domain0* running control software in a XenoLinux environment.**

# Xen: dom0

- Automatically started at boot time
- Special management privileges: only domain with direct access to physical devices and the control interface to the hypervisor
- Responsibilities
  - Creating, destroying, and managing DomUs
  - Device management (network, disk, etc.)
  - I/O request mediation for DomUs
  - Resource allocation for DomUs

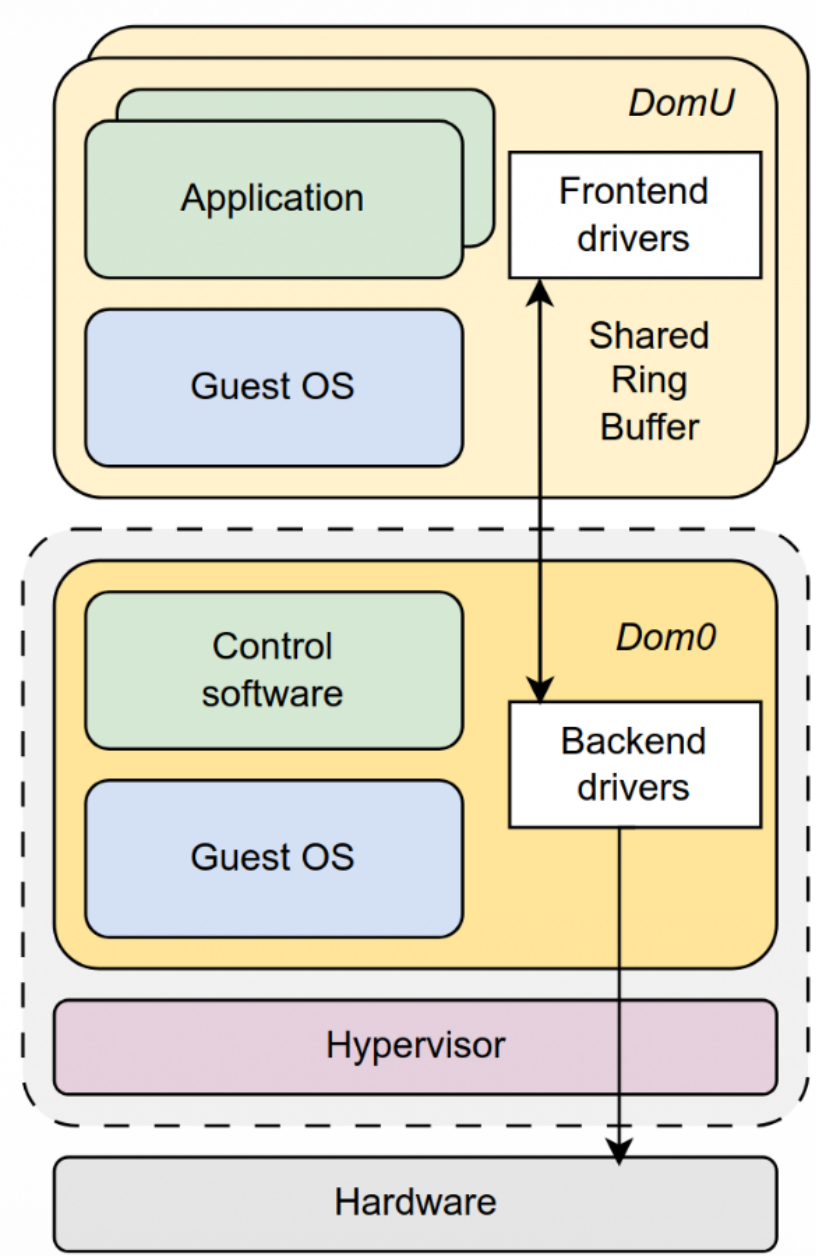
# Xen: control transfer

- Xen provides two mechanisms
  - **Hypercall** (sync): DomU /Dom0 requests a privileged service, allocating resources or managing memory
  - **Event channel (async)**: Signals, interrupts, notifications between domans and hypervisor.



# Xen: split drivers

- Multiple DomUs need access to hardware
  - Xen doesn't include drivers in the hypervisor itself, they are run in domains
  - Dom0 is the only domain with direct access to hardware
  - DomU domains must communicate with Dom0 to perform I/O



# Xen: I/O rings

- Shared-memory circular queue of descriptors allocated by a domain but accessible from within Xen
  - Descriptors do not hold the I/O data themselves; instead, they point to buffers allocated by the Guest OS.
  - Requests are processed asynchronously

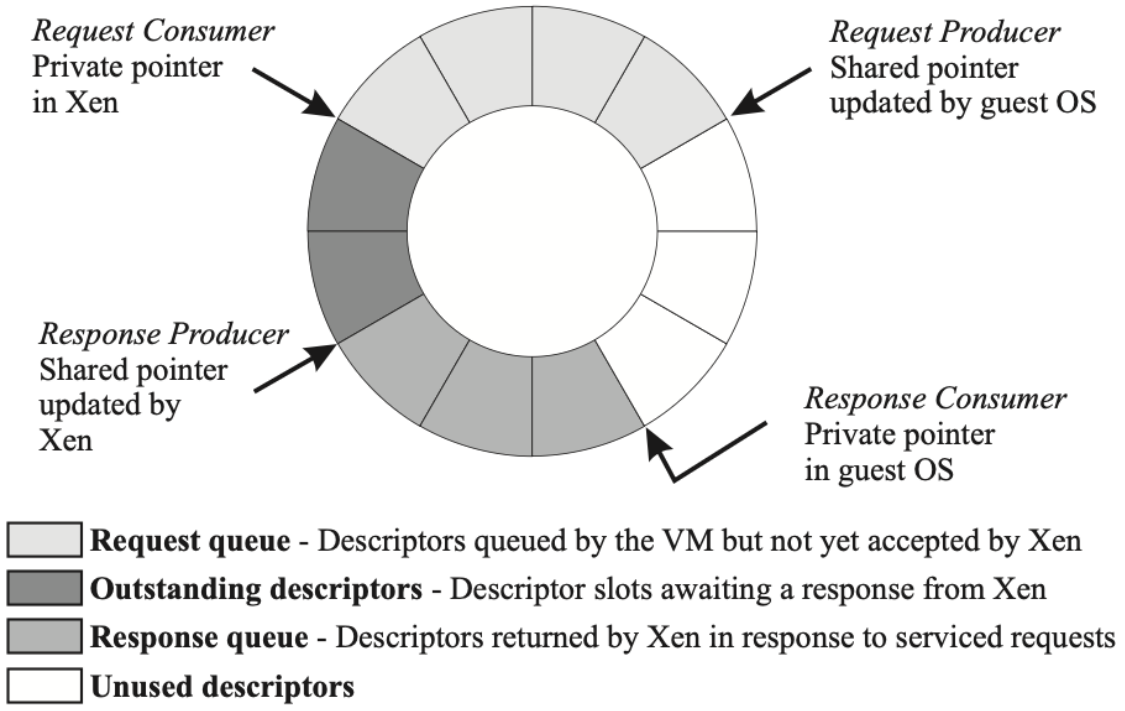



**Figure 2: The structure of asynchronous I/O rings, which are used for data transfer between Xen and guest OSes.**

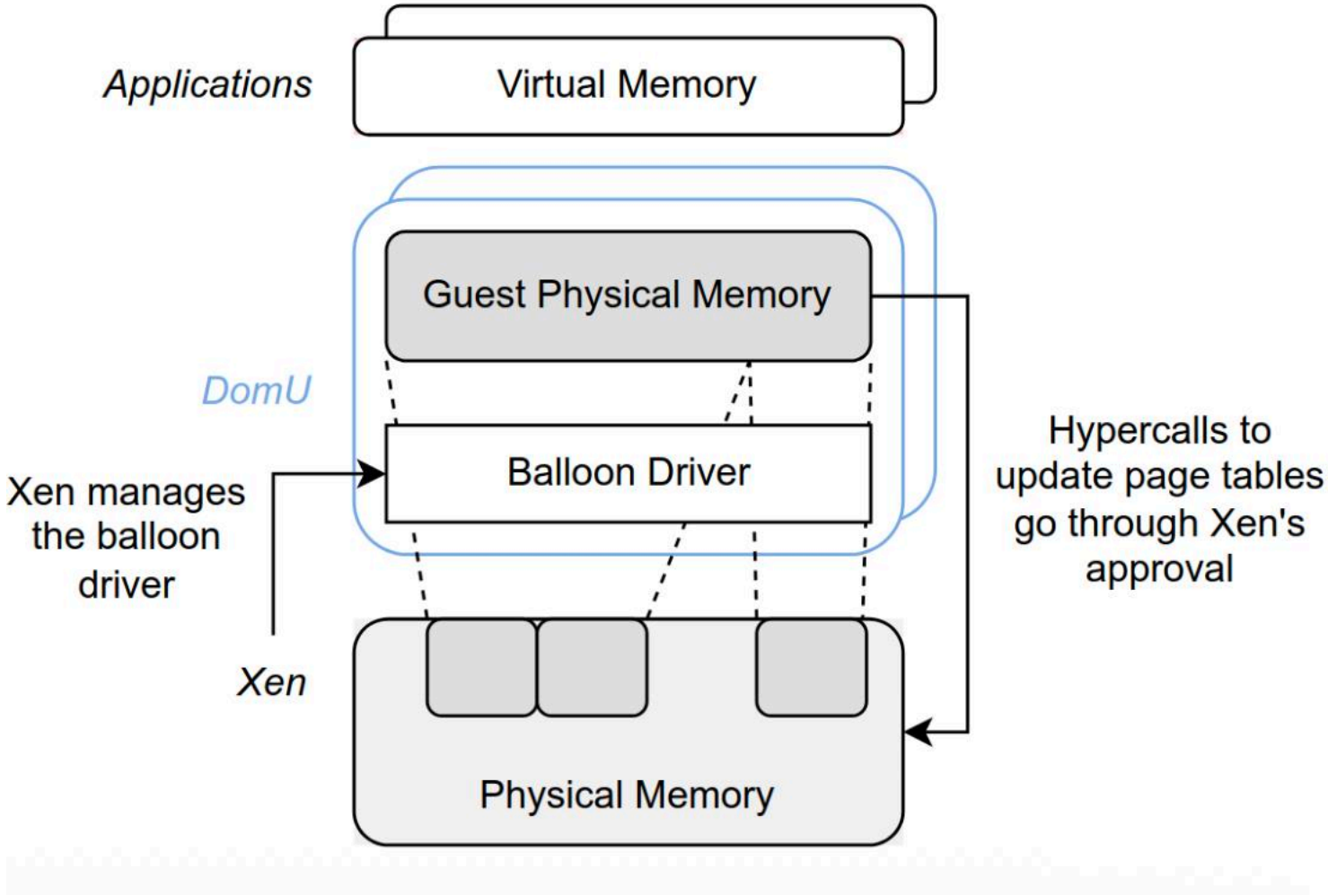
# Xen: paravirtualized x86 interface

| | |
|---|---|
| **Memory Management** | |
| Segmentation | Cannot install fully-privileged segment descriptors and cannot overlap with the top end of the linear address space. |
| Paging | Guest OS has direct read access to hardware page tables, but updates are batched and validated by the hypervisor. A domain may be allocated discontiguous machine pages. |
| **CPU** | |
| Protection | Guest OS must run at a lower privilege level than Xen. |
| Exceptions | Guest OS must register a descriptor table for exception handlers with Xen. Aside from page faults, the handlers remain the same. |
| System Calls | Guest OS may install a 'fast' handler for system calls, allowing direct calls from an application into its guest OS and avoiding indirecting through Xen on every call. |
| Interrupts | Hardware interrupts are replaced with a lightweight event system. |
| Time | Each guest OS has a timer interface and is aware of both 'real' and 'virtual' time. |
| **Device I/O** | |
| Network, Disk, etc. | Virtual devices are elegant and simple to access. Data is transferred using asynchronous I/O rings. An event mechanism replaces hardware interrupts for notifications. |

**Table 1: The paravirtualized x86 interface.**

# Xen: memory virtualization

- Reservations are specified at time of creation
- Balloon driver (dynamic adjustment)
- Shared translation array (between guest virtual memory and actual machine memory)



# Xen: memory virtualization

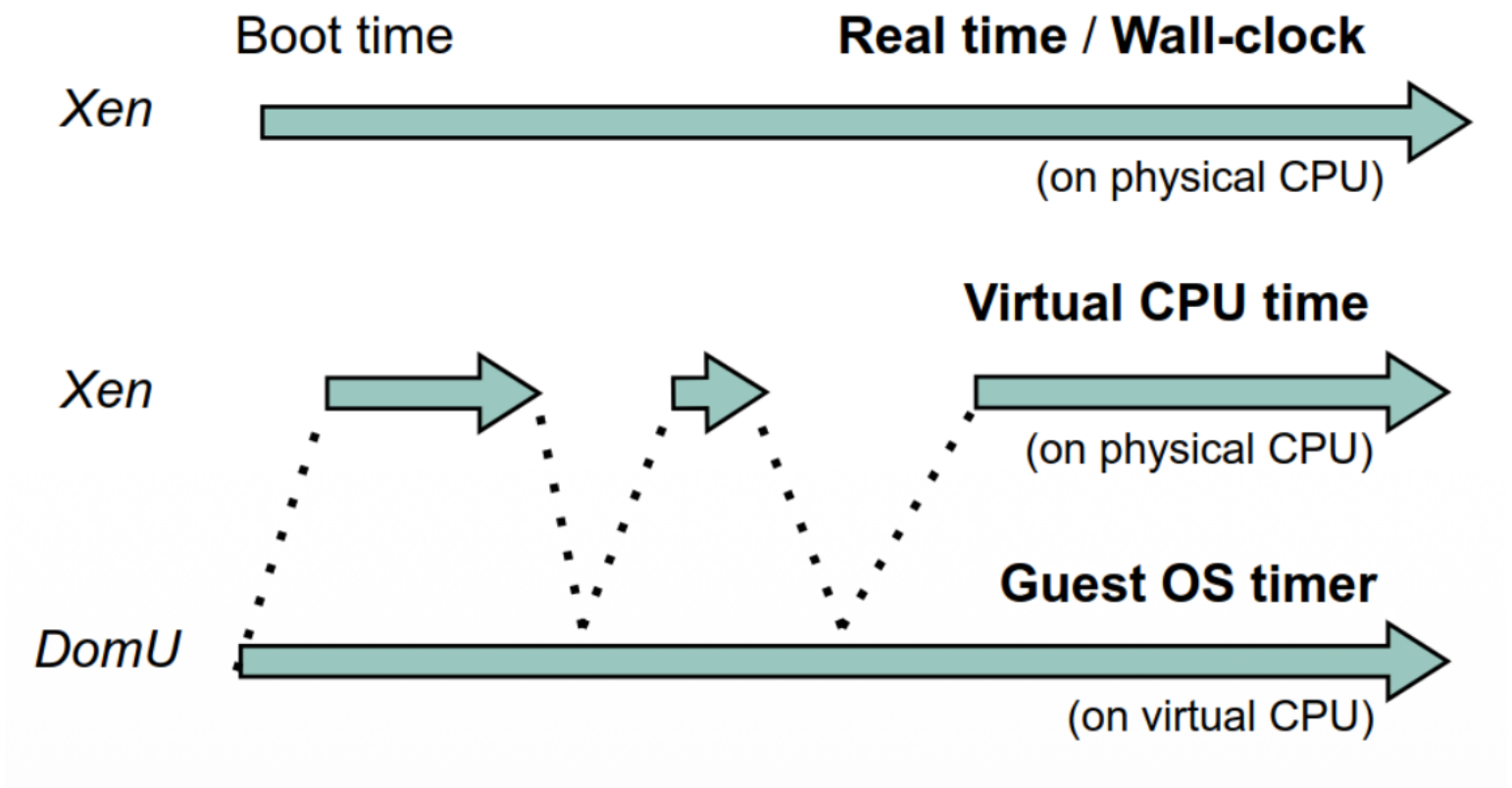
- x86 uses hardware page tables
- Xen exists in a 64MB section at the top of every address space
  - To avoid TLB flush
- DomUs allocate and manage their own page tables
  - DomU requires new page table → allocates and initializes from its own memory reservation and registers it to Xen
  - Xen is only involved in updates via hypercall
  - Updates must be validated before being applied
  - Can queue updates and apply in batch

# Xen: CPU virtualization

- Protection
  - Xen in ring 0, Guest kernel in ring 1
  - DomUs cannot execute privileged instructions directly.
  - Privileged instructions are replaced with hypercalls
- Xen uses Borrowed Virtual Time (BVT) to schedule virtual CPUs
  - Threads can ”borrow” virtual time by warping their virtual time to an earlier point, making them appear to have a higher priority
- Xen handles page fault exceptions on behalf of DomUs

# Xen: virtual timers

- How to handle timeout?
  - Network: When do I need to resend a packet? Did I receive something while I was off?



# Xen: performance

Xen performs almost as good as native Linux for all microbenchmarks!

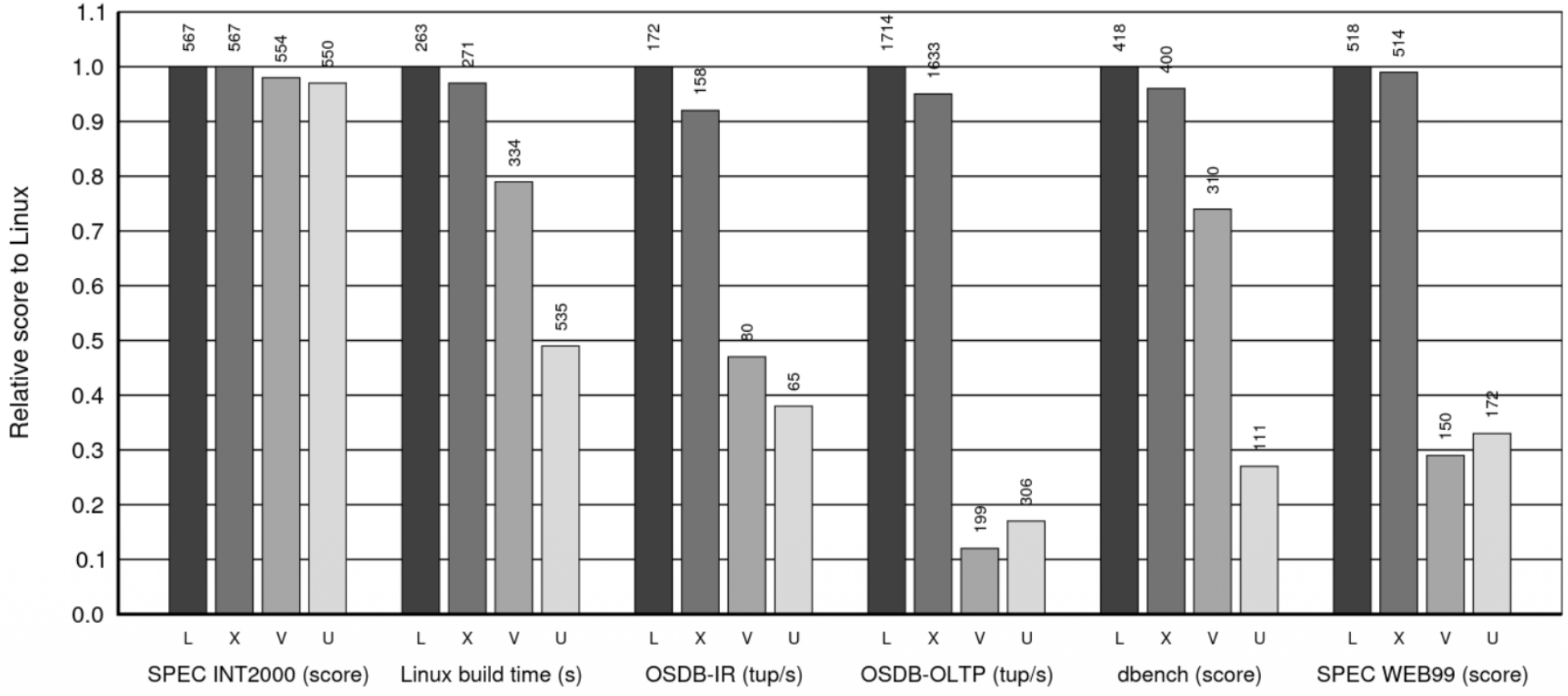



**Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).**

# Xen: scalability

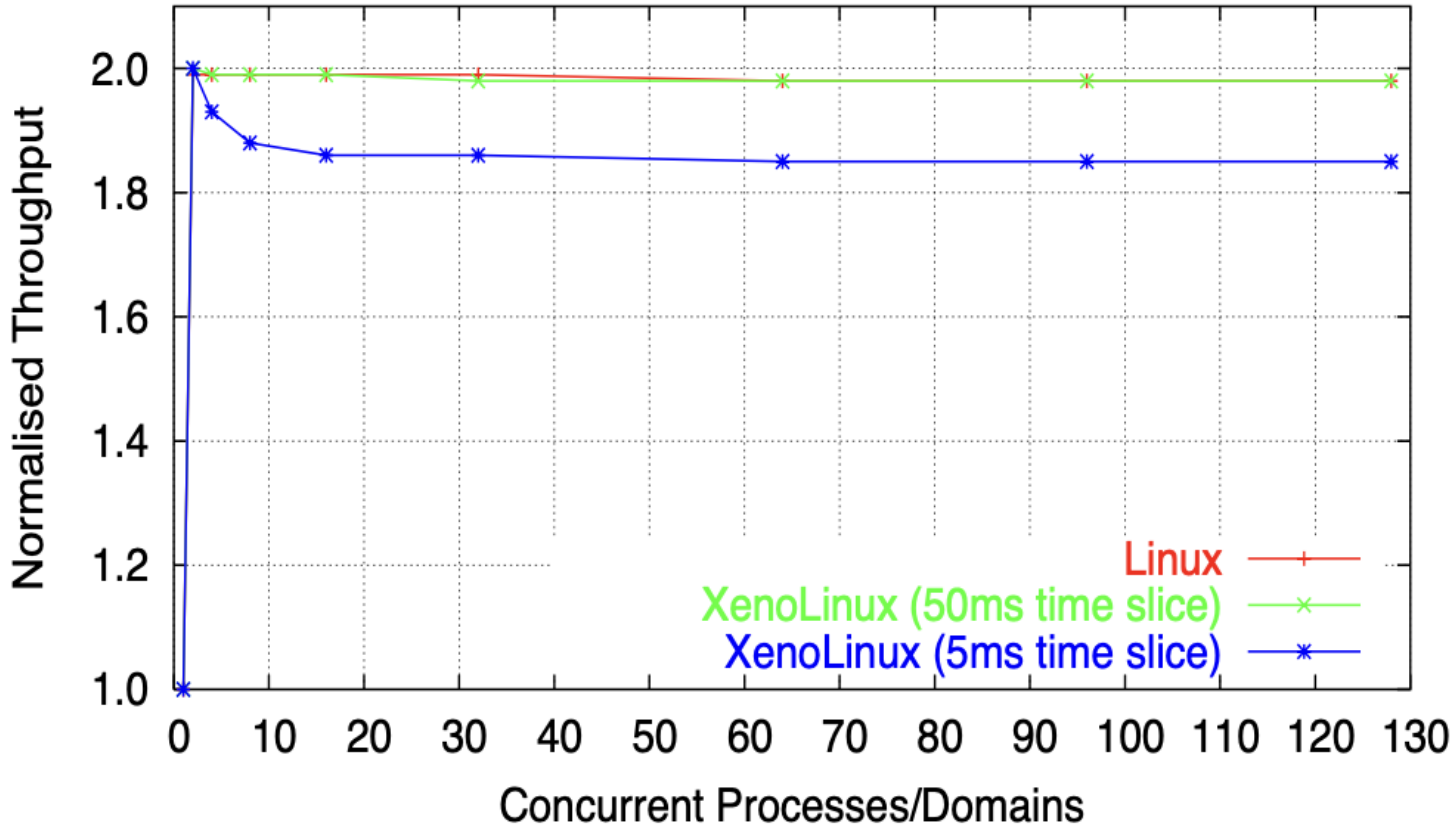



**Figure 6: Normalized aggregate performance of a subset of SPEC CINT2000 running concurrently on 1-128 domains**

# Xen: modifications

| 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).**

# Xen: impacts

- Industry adoption: Basis for commercial XenSource, then bought by Citrix ($500 million)
- Backbone of Amazon's AWS EC2 in early days, used Xen hypervisor until late 2010s → Cloud ran on Xen!
- Set stage for hardware virtualization extensions (Intel VT-x, AMD-V): paravirtualization influenced Intel/AMD to add new hardware instructions to make virtualization easier