# Paravirtualization

*ENGR 689 (Sprint)*

TEXAS A&M UNIVERSITY

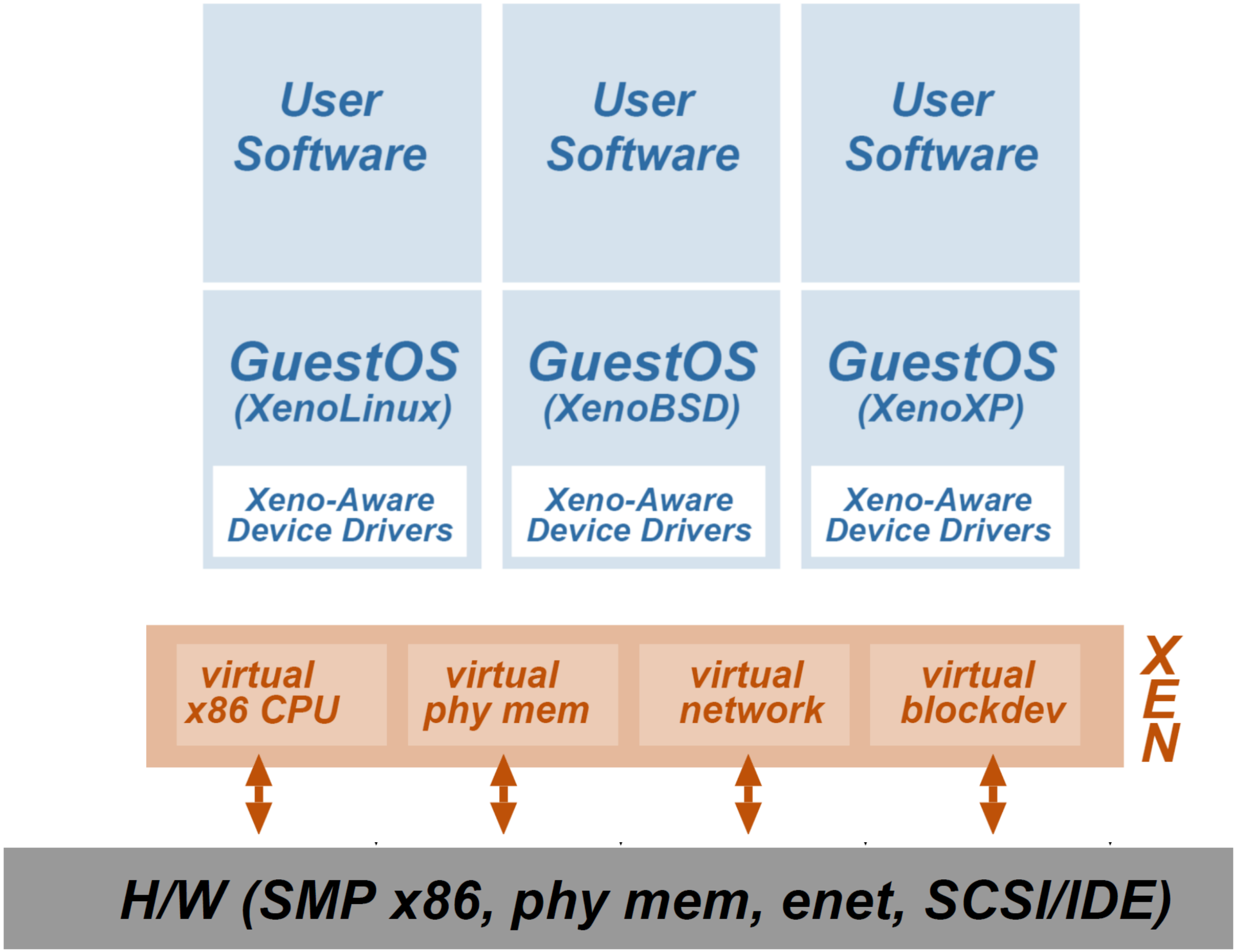
# Why Not Classic Virtualization

- Classic Virtualization (Popek & Goldberg) requires **<u>identical emulation</u>**
  - No change allowed to the guest OS & applications
  - Trap & emulate all privileged operations
  - Require hardware virtualization to be efficient
- However, most cloud users do not care (why?)

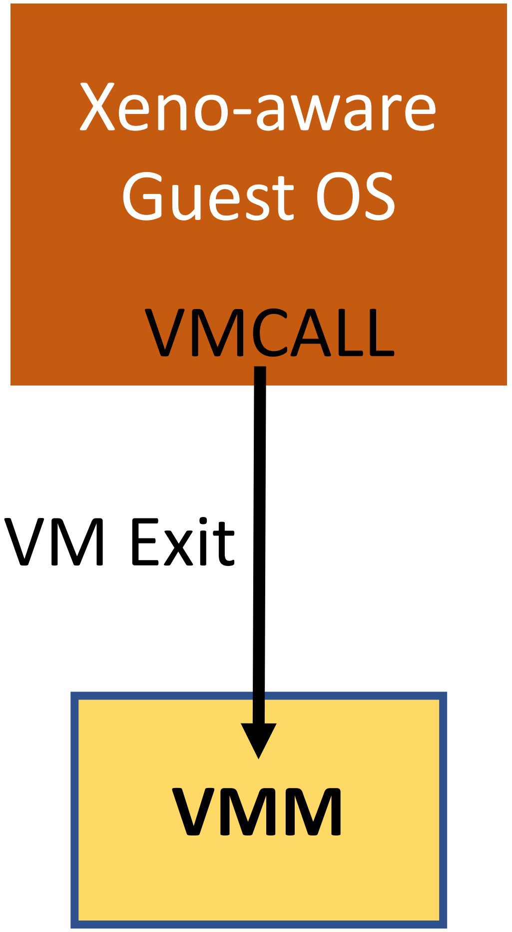
# Xen: A Paravirtualized VMM

- VMM that collaborates with the guest OSes, instead of deceiving them.
- Pro: avoids expensive trap & emulate
- Con: requires minor modification in guest OSes

# Xen Architecture

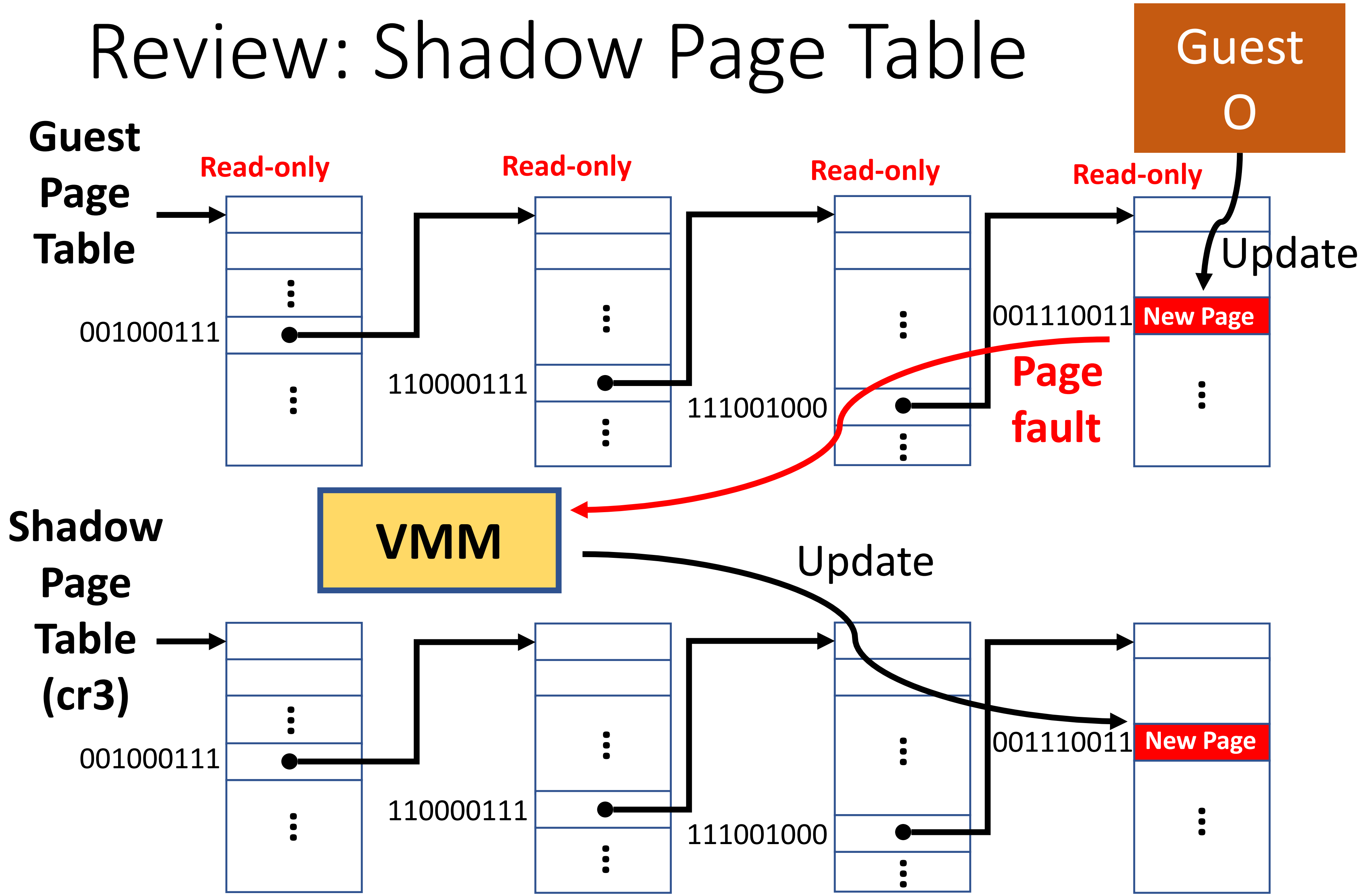


# Hypercalls



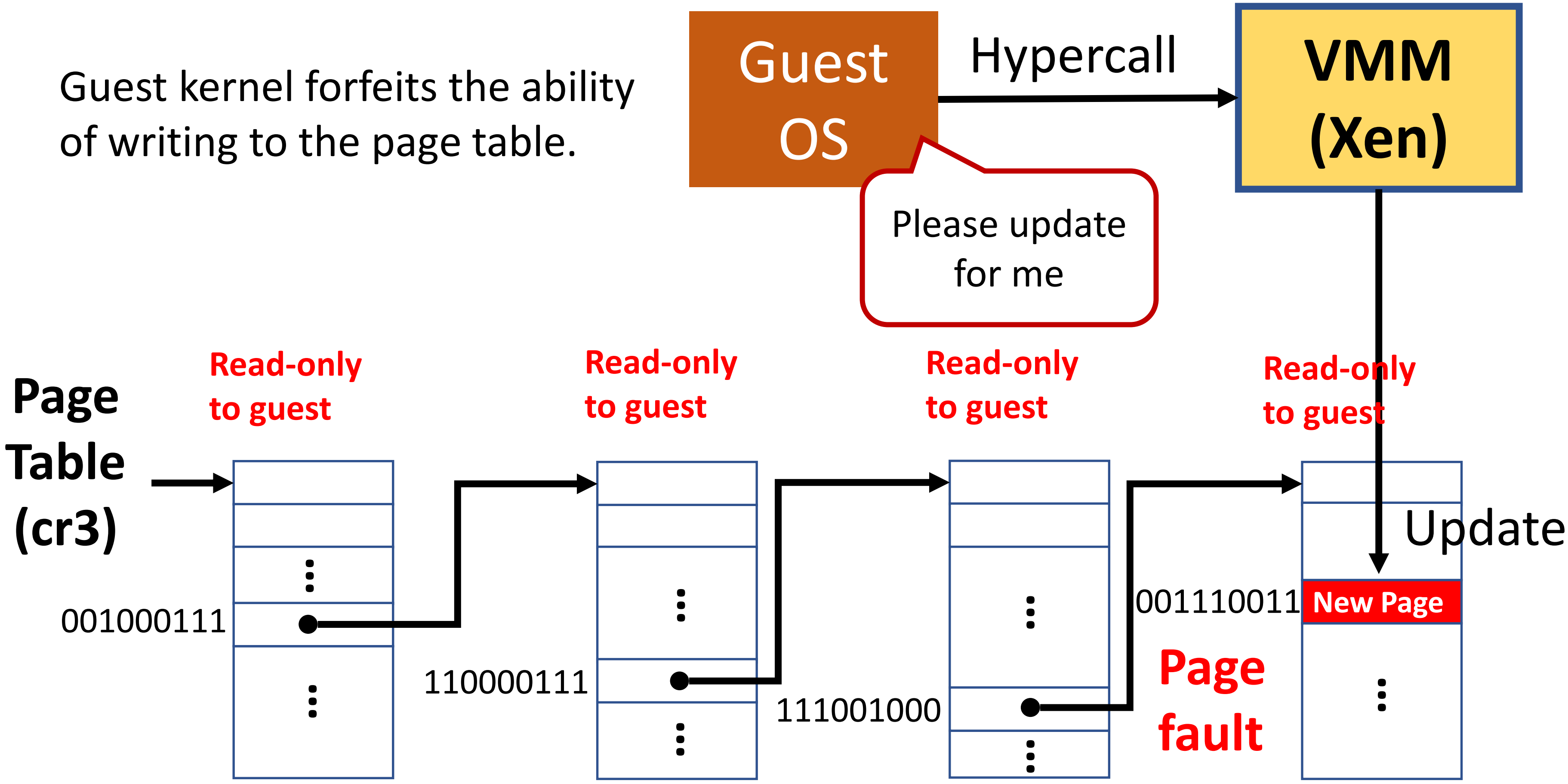

- Software traps to the VMM ➔ less expensive than handling hardware traps
- Pre-defined interface
- Can batch multiple privileged instructions at a time

# Review: Shadow Page Table



# Example: Paging with Hypercalls

Guest kernel forfeits the ability of writing to the page table.



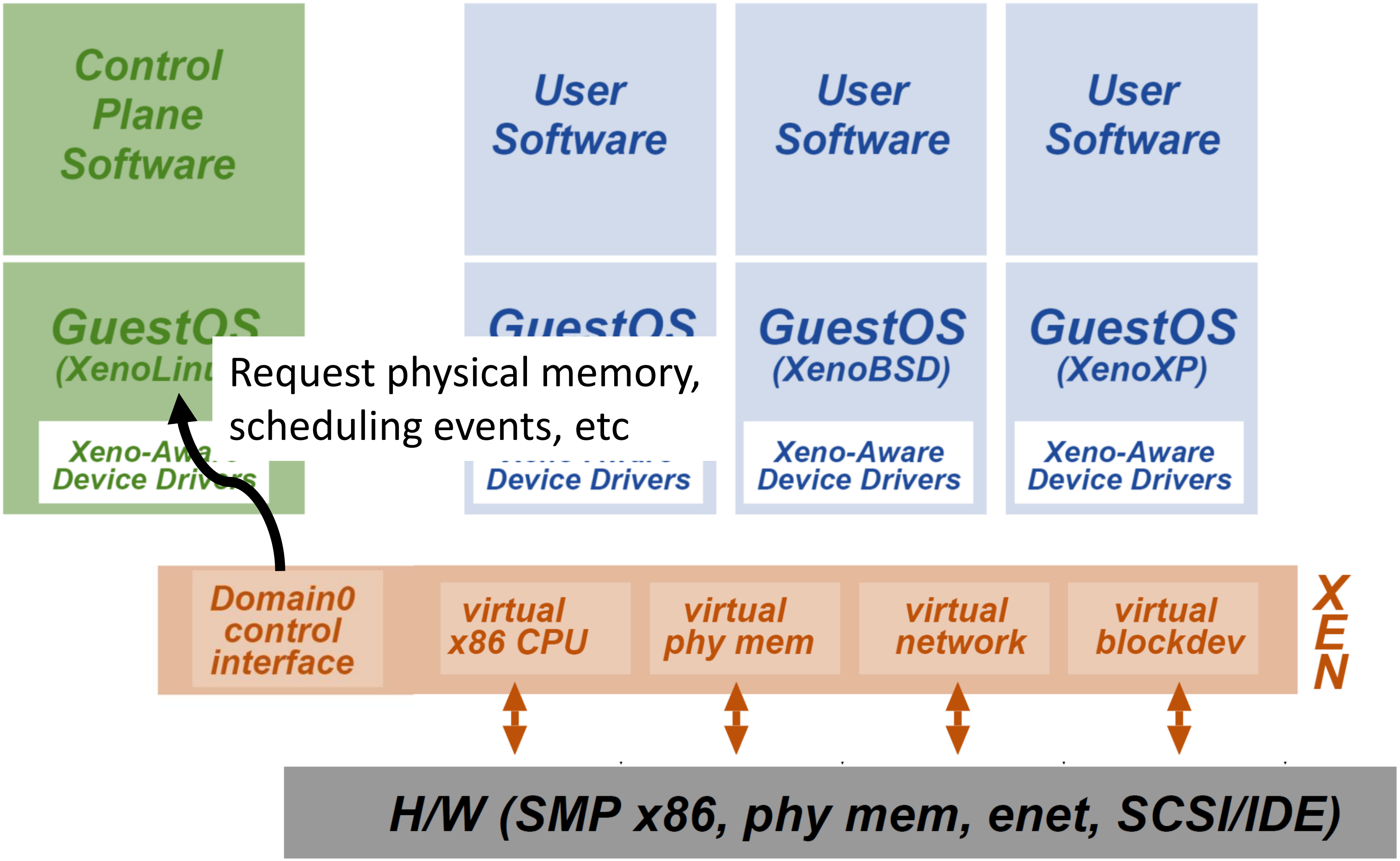
# VMM Complexity

- Paravirtualization makes VMM significantly smaller
  - Only implement a small amount of hypercall interfaces (~40 in current Xen VMM)
  - Doesn't have to "guess" what guest is doing
  - Just validate updates from the guest and apply them
- The only complexity is modifying guest OSes
  - 2,995 lines in Linux (1.36%)
  - 4,620 lines in Windows (0.04%)

# Controlling Virtual Machines

- Many control operations are too complex to be part of the VMM
    - Initializing hardware at startup
    - Allocating and reclaiming physical memory
    - Making CPU scheduling decisions
    - Creating virtual interfaces (VIFs) and virtual block devices (VBDs)
    - Creating and destroying VMs

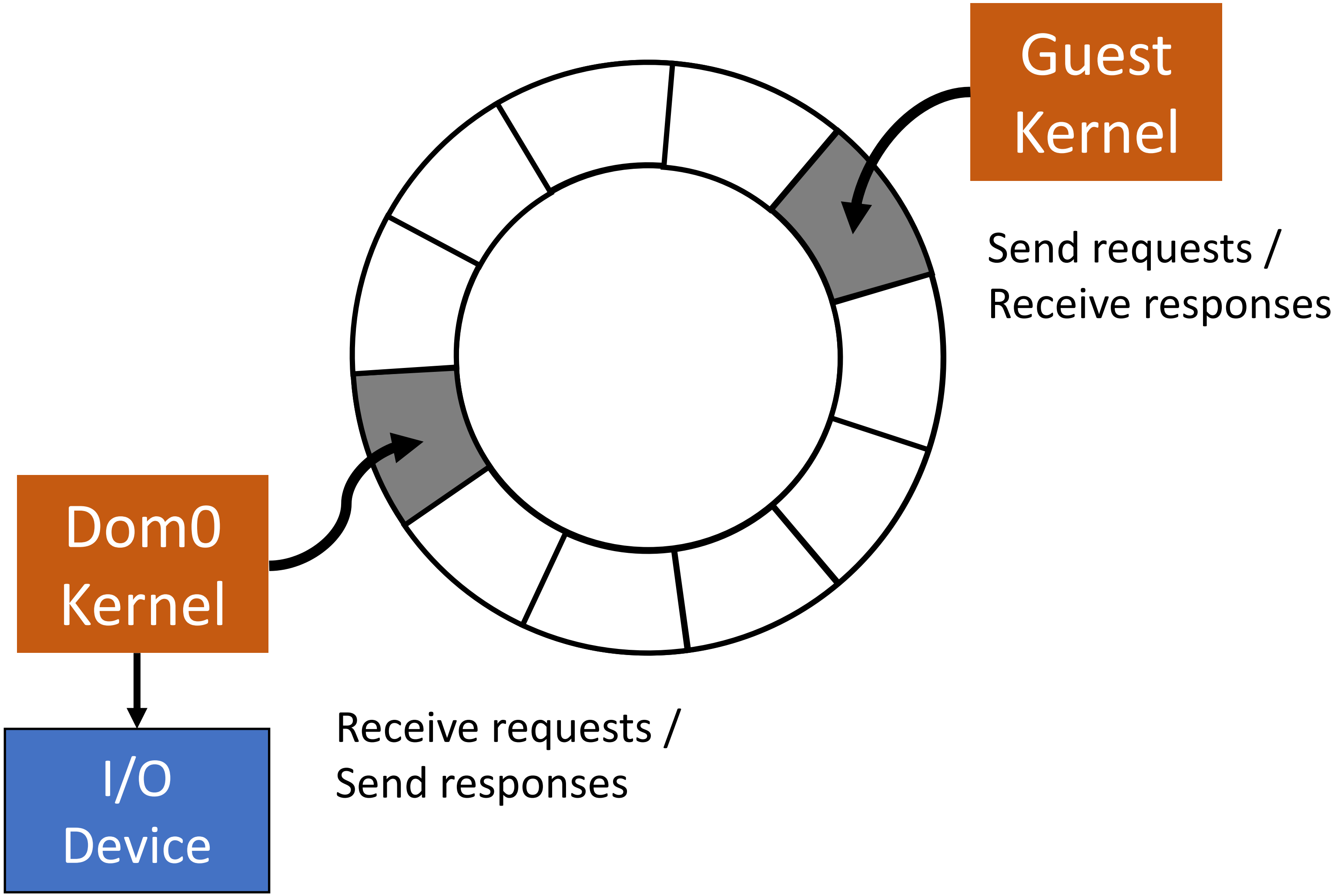
# Domain 0 (Dom0) VM



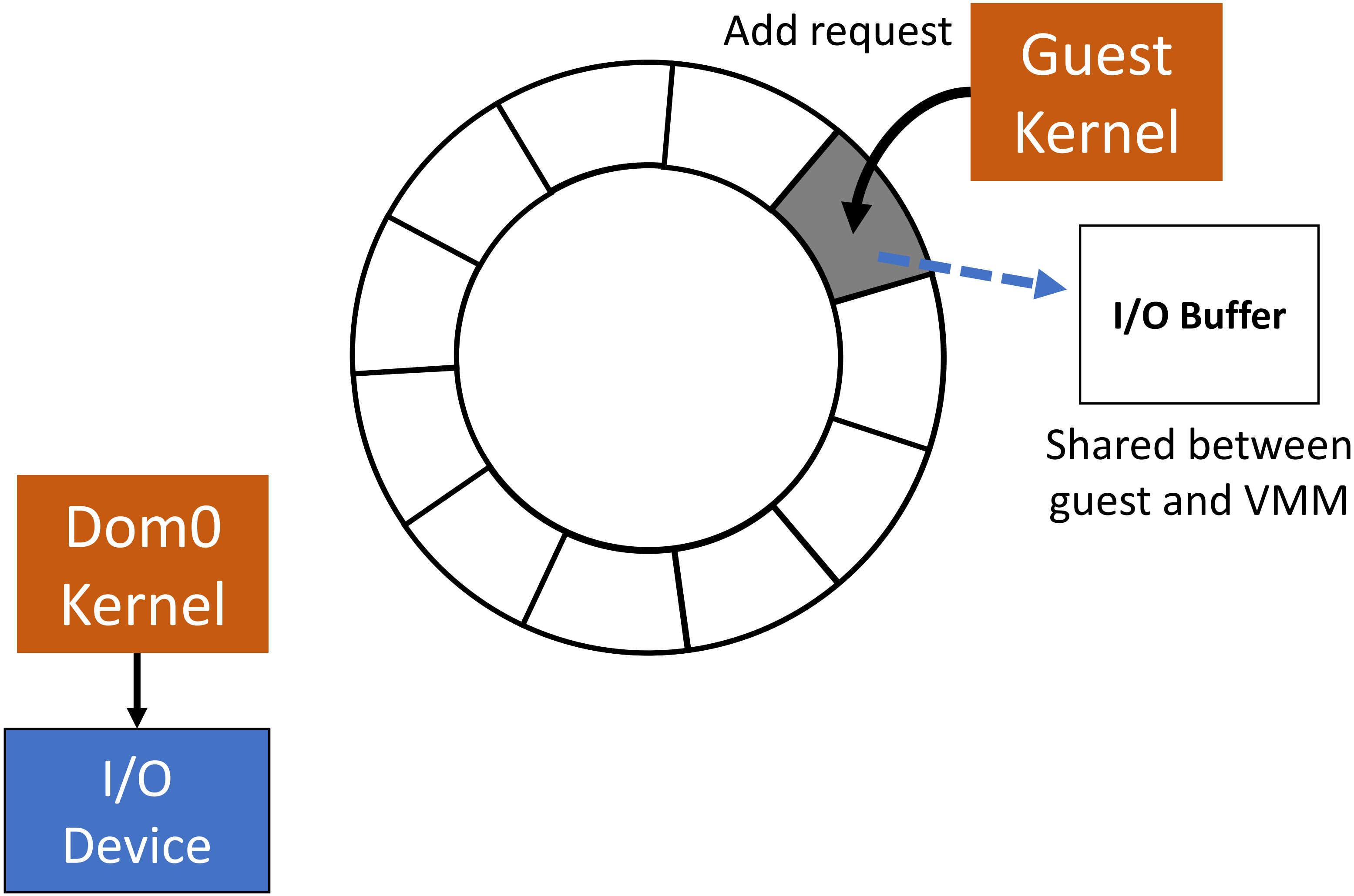
# I/O Paravirtualization

- How to efficiently collaborate on I/O
  - No frequent context switches between guest and VMM
  - Batching I/O operations
  - Favoring **throughput** over **latency**

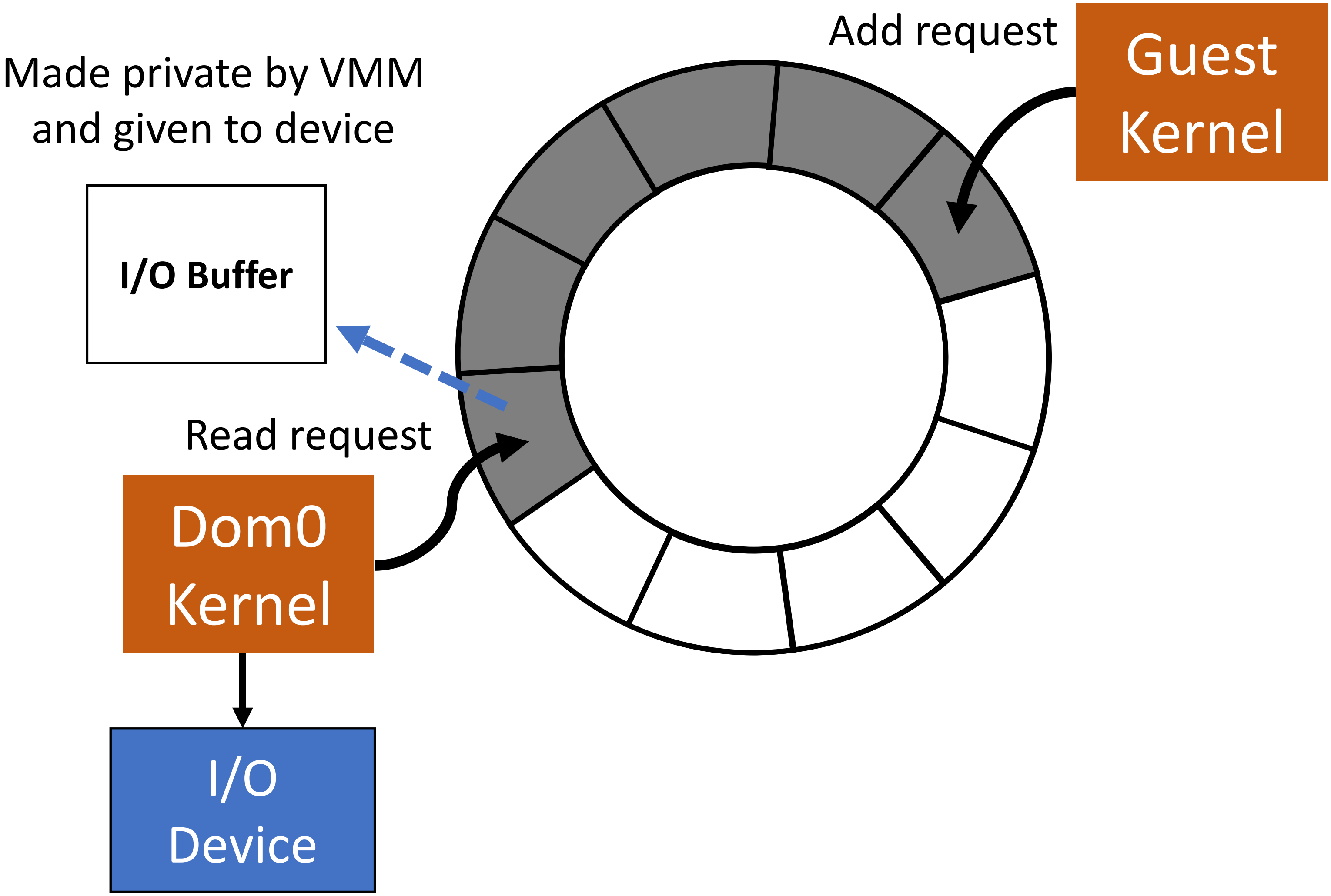
# I/O Ring Buffer (1/5)



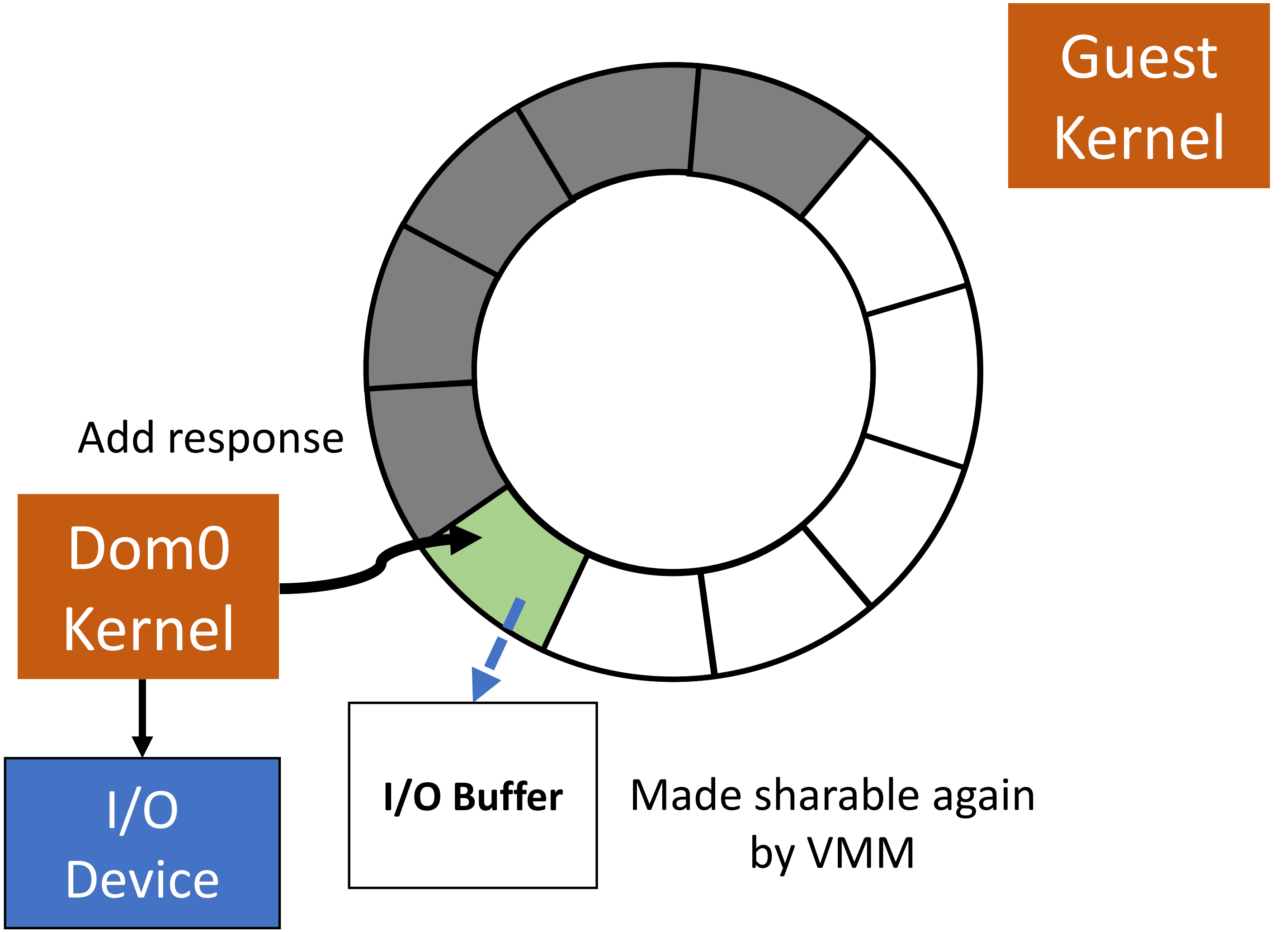
# I/O Ring Buffer (2/5)



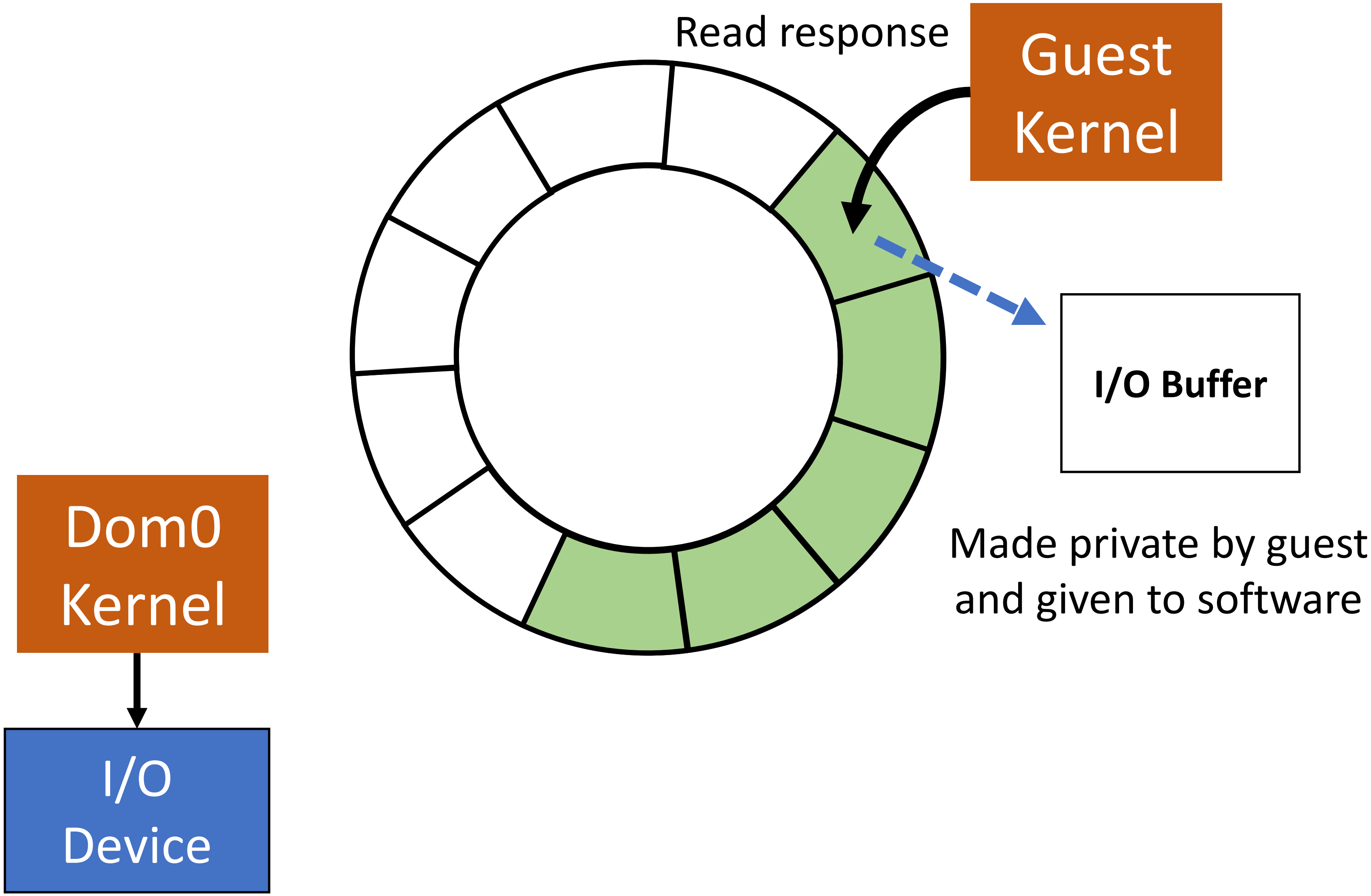
# I/O Ring Buffer (3/5)



# I/O Ring Buffer (4/5)



# I/O Ring Buffer (5/5)



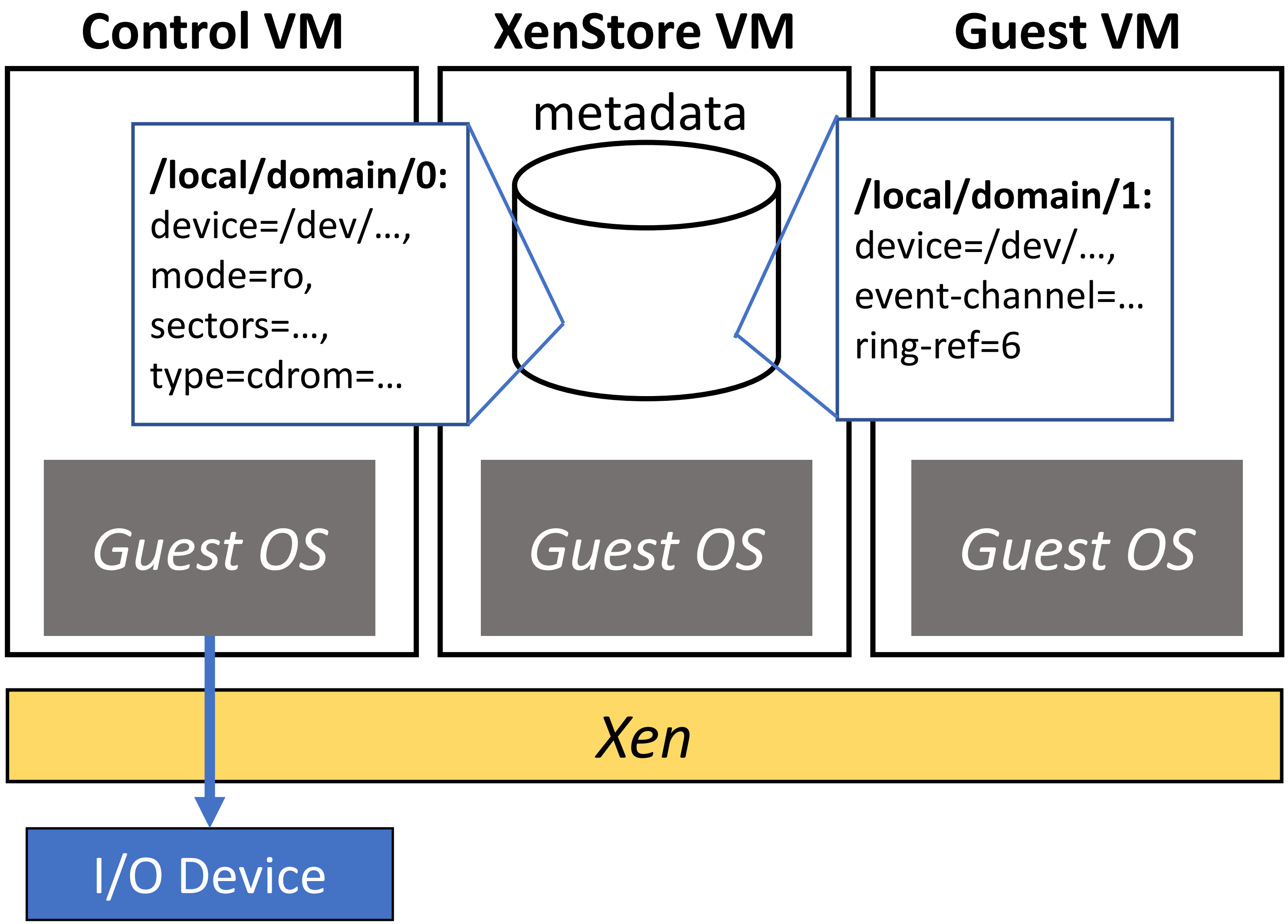
# XenStore

- VM metadata stored in a shared database
    - VM image and boot-up commands
    - Memory & CPU configuration
    - Device and ring buffer information
- Special APIs

```
# xenstore-list /vm
 00000000-0000-0000-0000-000000000000
 9b30841b-43bc-2af9-2ed3-5a649f466d79-1
```

```
# xenstore-read /vm/9b30841b-43bc-2af9-2ed3-5a649f466d79-1/vcpus
 1
```

# XenStore



Control VM
XenStore VM
Guest VM
metadata
/local/domain/0:
device=/dev/…,
mode=ro,
sectors=…,
type=cdrom=…
/local/domain/1:
device=/dev/…,
event-channel=…
ring-ref=6
Guest OS
Guest OS
Guest OS
Xen
I/O Device

# Xen + Hardware Virtualization

- With Hardware Virtualization (VT or AMD-V), Xen can run unmodified kernels now
- Paravirtualization is still used for I/O and a few hardware features (system time, random number generator, etc)
- Only need guest device drivers for hypercalls and accessing ring buffers

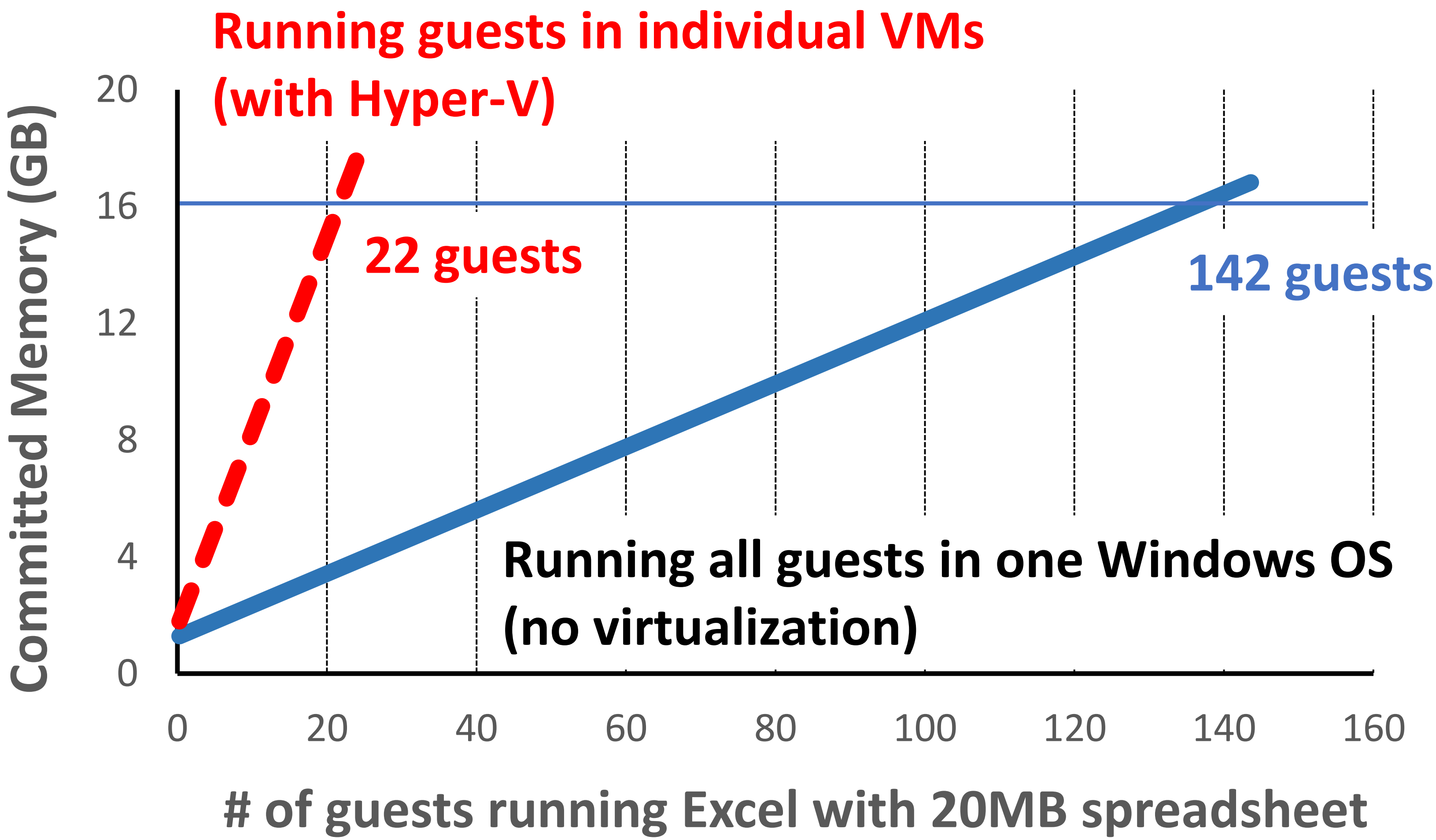
# VMs for Cloud: Good or Bad?

- Benefits:
  - **Backward compatibility:** unmodified OS and application
  - **Security isolation:** guest states governed by VMM
- Cons:
  - **Density:**

$$\#\text{ VMs per host} = \text{MIN}\left( \frac{\text{Total \# CPUs}}{\text{\# CPUs per VM}}, \frac{\text{Total RAM}}{\text{\# RAM per VM}} \right)$$

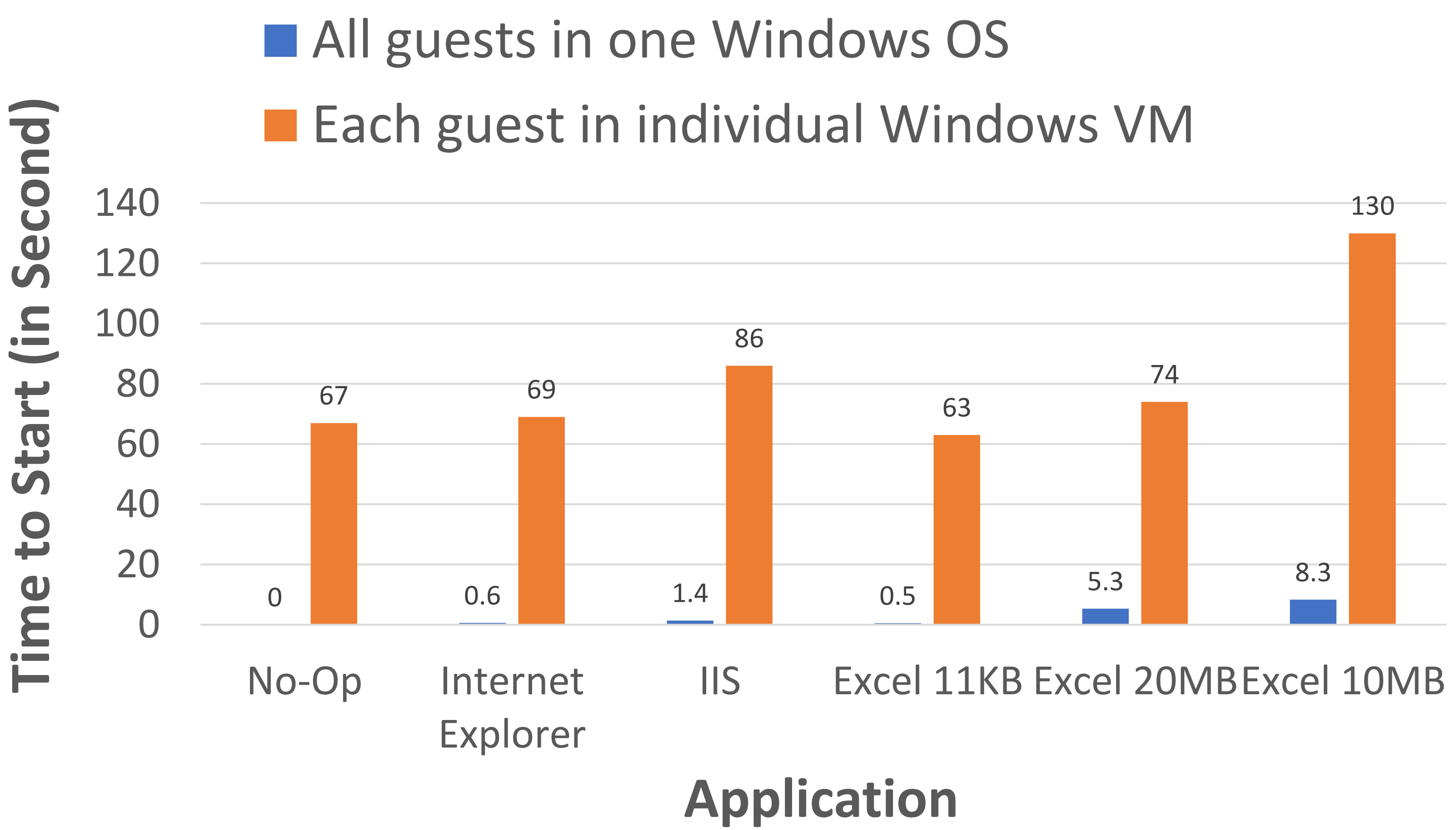
  - **Startup cost:** Cannot quickly spin up new VM or restore a paused idle VM

# VM Density

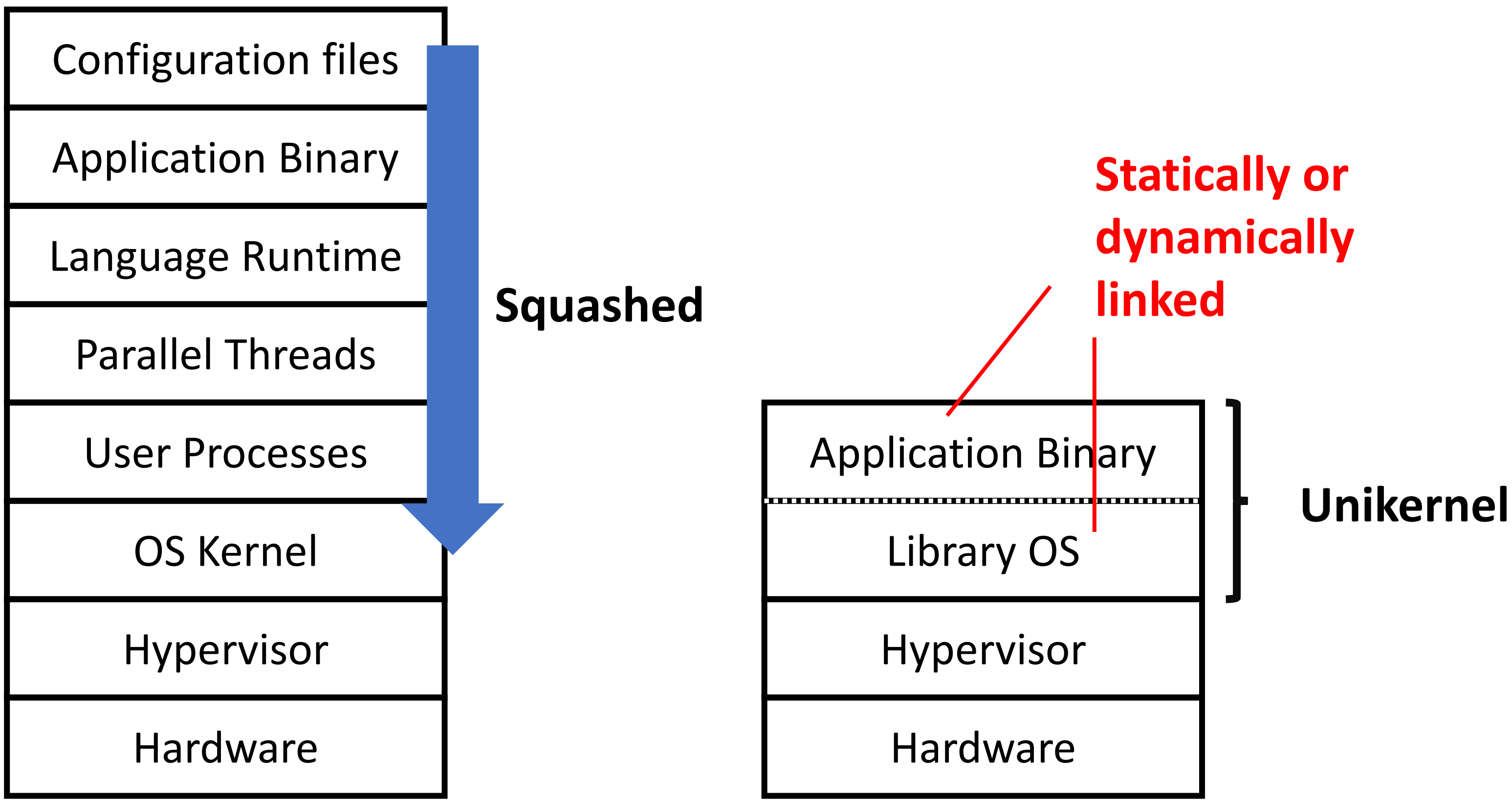


Running guests in individual VMs (with Hyper-V)
Committed Memory (GB)
20
16
12
8
4
0
22 guests
142 guests
Running all guests in one Windows OS (no virtualization)
0
20
40
60
80
100
120
140
160
# of guests running Excel with 20MB spreadsheet

# VM Startup Time

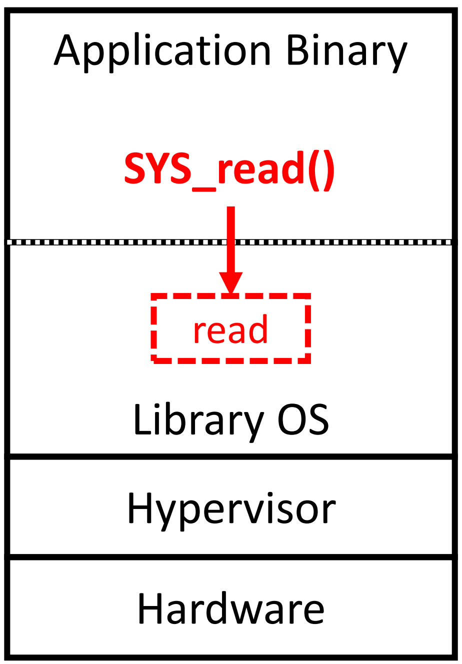


# Squashing the OS Stack



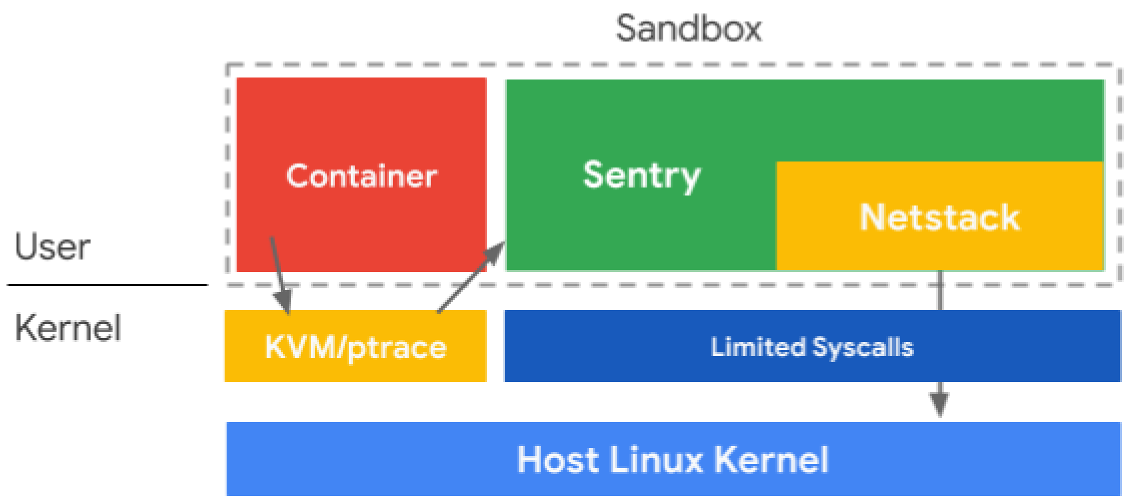
# No More System Calls

- Characteristic of library OSes:
  system calls ➔ function calls inside application




- Benefit 1:
  **No need for virtualization (i.e., Popek & Goldberg requirements)**
- Benefit 2:
  **No context switches**

# gVisor




- gVisor is widely used in Google Cloud Function, App Enginer, and other frameworks
- Sandboxing applications in “containers”
- A “Sentry” acting as the library OS to serve system calls from the containers

# MicroVMs (e.g., Firecracker)