Based on slides by Harsha V. Madhyastha

EECS 482 Introduction to Operating Systems

Spring/Summer 2020

Lecture 25: Virtual machine monitors

Nicole Hamilton
https://web.eecs.umich.edu/~nham/
nham@umich.edu

Agenda

1. Announcements.
2. Final exam.
3. Project 4.
4. Student evals.
5. Virtual machine monitors.
6. The big ideas in 482.

Agenda

1. Announcements.
2. Final exam.
3. Project 4.
4. Student evals.
5. Virtual machine monitors.
6. The big ideas in 482.

Announcements

Labs ended last Friday.

Today is the last lecture.

Style grading on P3 is 95% complete.

P4 due Mon Aug 17.

Final exam review session Tue Aug 18, 3:00 pm to 5:00 pm.

Final exam Thu Aug 20, 3:00 pm to 5:00 pm.

Agenda

1. Announcements.
2. Final exam.
3. Project 4.
4. Student evals.
5. Virtual machine monitors.
6. The big ideas in 482.

Final exam

Just like the midterm.

Online using *Crabster.org* Thu Aug 20 3:00 to 5:00 pm EDT.

We will post a link via Canvas and Piazza once the exam is live.

We will monitor Piazza for questions.

Material for the final:

1. Everything except my bonus lecture.
2. Focus will be on P3 and P4 and lecture topics since the midterm.
3. Review session Tue Aug 18 3:00 pm to 5:00 pm.

Final exam policy

The exam will be *"open everything"* except collaboration.

You can use any existing resource, including lecture notes, the book, your project solutions, you can even use Google, and your IDE.

The only thing you can't do is collaborate with others, including using social media to solicit help. If you can find an existing answer on stackexchange that's helpful, that's fair game. But you can't post a question.

Also, parts of the exam ask for short answers, which must be *in your own words*. Cutting and pasting word-for-word from an existing source and "close copying" will be treated as plagiarism and reported to the Honor Council.

Agenda

1. Announcements.
2. Final exam.
3. Project 4.
4. Student evals.
5. Virtual machine monitors.
6. The big ideas in 482.

Project 4 Testing

Verifying concurrency

- Test with a pair of requests
- Consider every combination of request types
- Vary commonality in pathnames
- Block around “slow” operations

Example:

- FS_CREATE for */a/b/c* blocks holding write lock for *b*
- FS_CREATE for */a/b/d* cannot proceed
- FS_CREATE for */a/f/g* should be able to complete

Agenda

1. Announcements.
2. Final exam.
3. Project 4.
4. Student evals.
5. Virtual machine monitors.
6. The big ideas in 482.

Student evals

Please get them done.

They are super important.

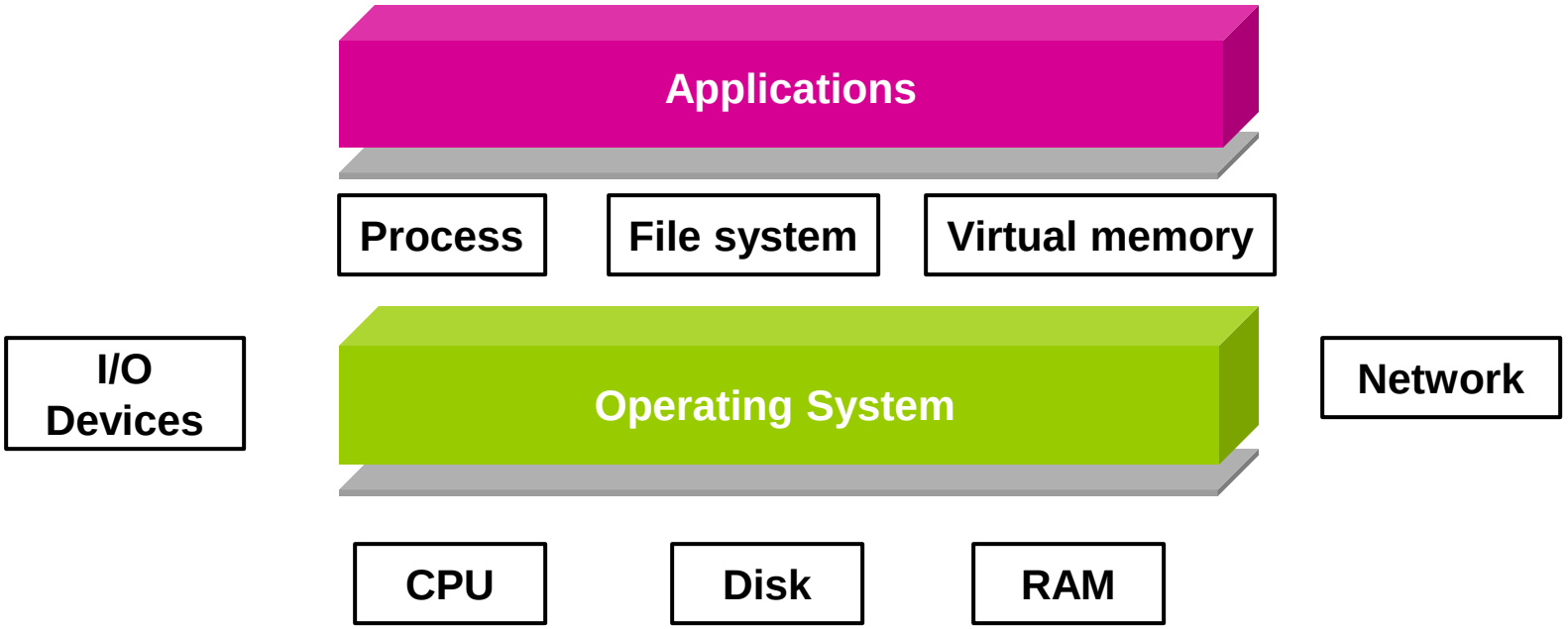
The 370 and other questions …

If you wonder if they matter …

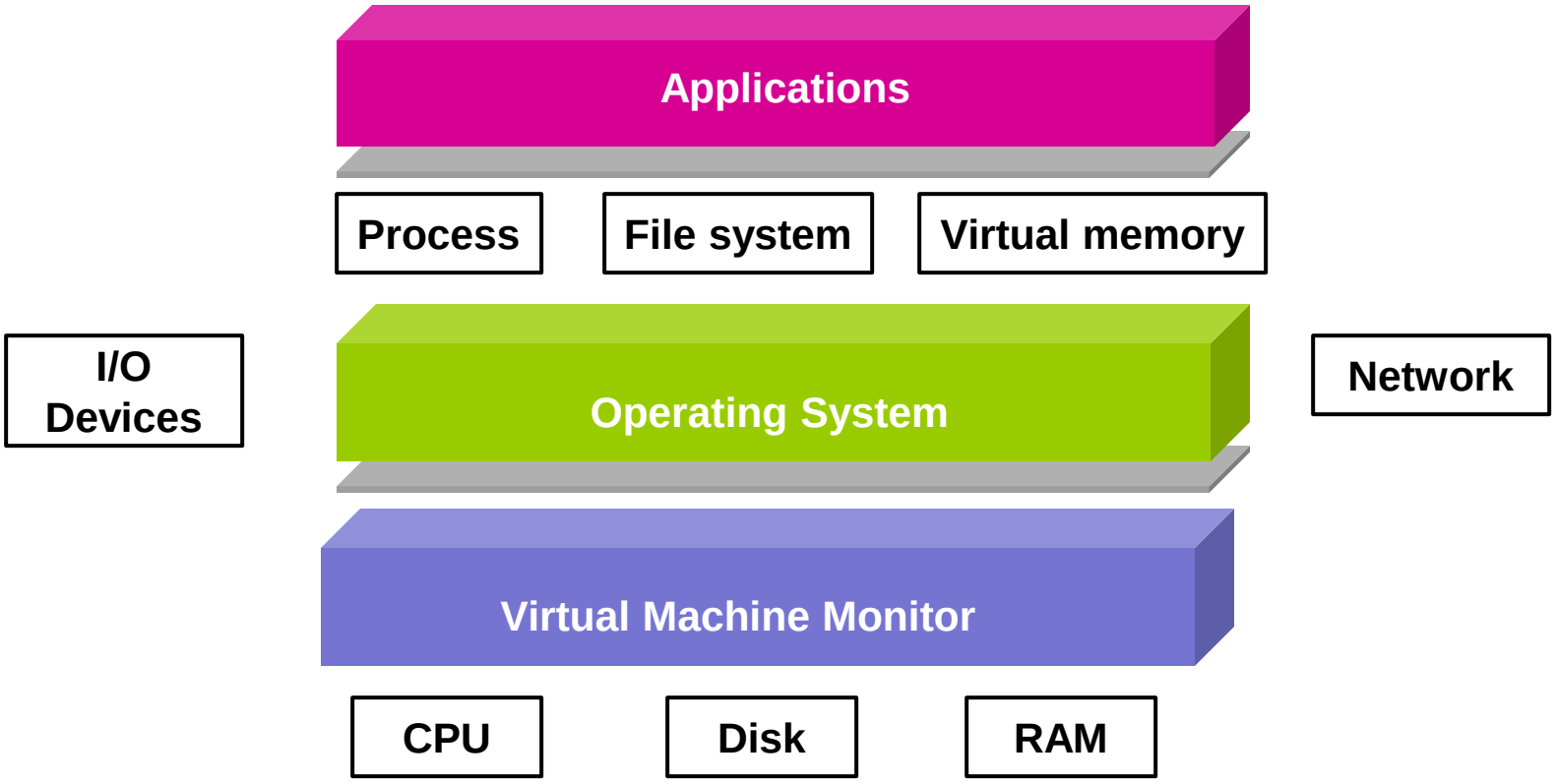
Agenda

1. Announcements.
2. Final exam.
3. Project 4.
4. Student evals.
5. Virtual machine monitors.
6. The big ideas in 482.

OS Abstractions



Virtual Machine Monitor



What is a VMM?

OS enables co-existence of multiple processes.

Offers illusion that each process is on own computer.

A VMM enables multiple OS instances to run simultaneously on a machine.

What interface should VMM export?

A VMM virtualizes an entire physical machine.

Offers illusion that OS has full control over hardware.

VMM “applications” (OSes) run in virtual machines.

Why run multiple OSes?

1. Some applications only run on certain operating systems and you only have one machine.
2. Developing and testing an OS.
3. Running possibly malicious software.
4. Cloud services like Amazon's AWS that rent out virtual machines to customers, all completely firewalled from each other.

Why run multiple OSes?

Resource utilization.

- Machines today are powerful; would like to share them, e.g., with Cloud services.
- Migrate VMs across machines without shutdown.

Software use and development.

- Can run multiple OSes simultaneously; no need to dual boot.
- Allows OS system development at user-level.

Many other cool applications

- Debugging, emulation, security, fault tolerance, …

Example of Cool VMM Tricks

How to experiment with apps, protocols, and systems on future hardware?

- Example: How to experiment with 100 Gbps network?

Time dilation

- VMM slows timer interrupt to make hardware (CPU, disk, network) appear faster to OS and apps.
- Example:
  - OS reads 10 Gb of data from network in 1 second, but thinks only 0.1 second has elapsed.
  - But, applications run 10x slower.

Example of Cool VMM Tricks

VMM Requirements

Fidelity

OSes and applications work without modification.

(But we may modify the OS a bit.)

Isolation

VMM protects resources and VMs from each other.

Performance

VMM is another layer of software → overhead.

As with OS, want to minimize this overhead.

VMware ESXi Server

Most popular VM at cloud services.

VMware ESX server uses a hypervisor model.

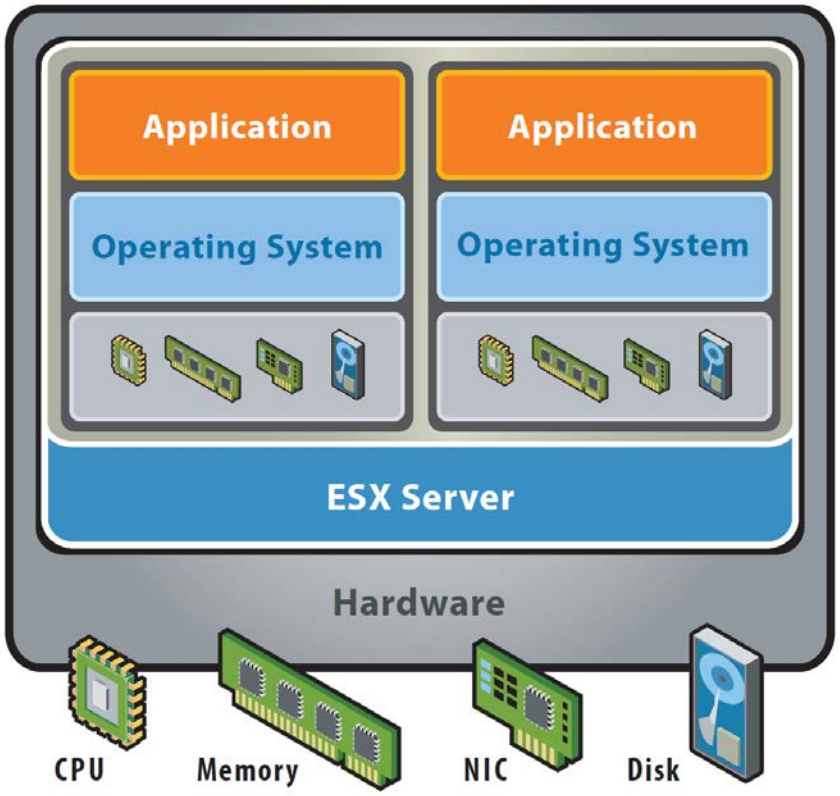
VMM runs directly on hardware.

Everything goes through the hypervisor.

Main downside is performance.

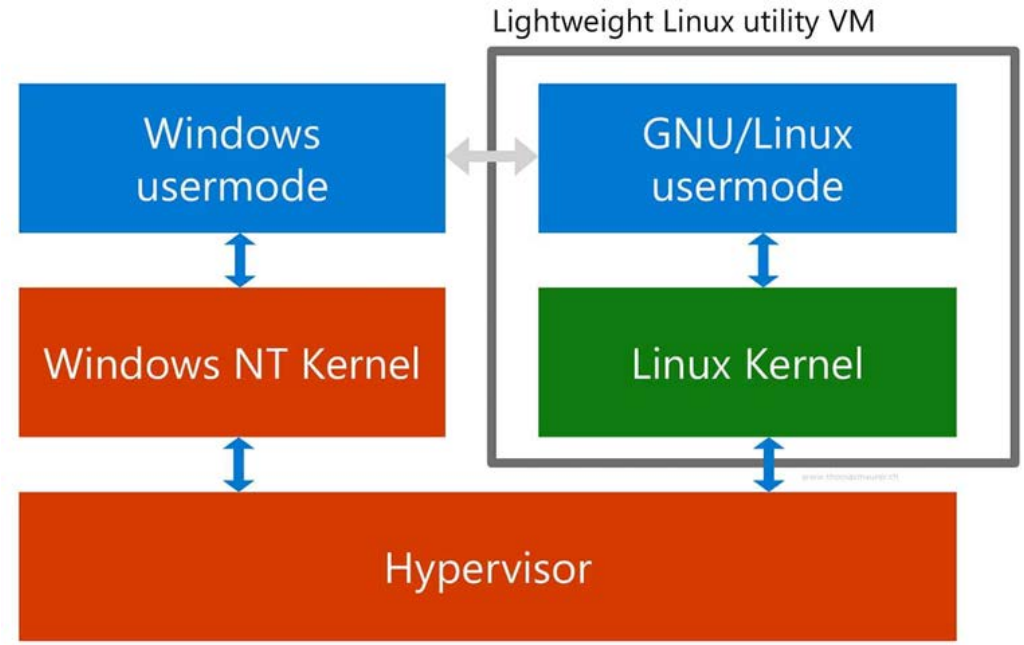
Software virtualization allows dynamic rewriting of the code executed in VM.

Rewrites only privileged instructions to reduce overhead.



Microsoft WSL 2

Microsoft's Windows Subsystem for Linux also uses a hypervisor architecture.



Xen

University of Cambridge open source research project.

Used by Amazon AWS.

Xen hypervisor runs at privilege, VMs (domains) run unprivileged.

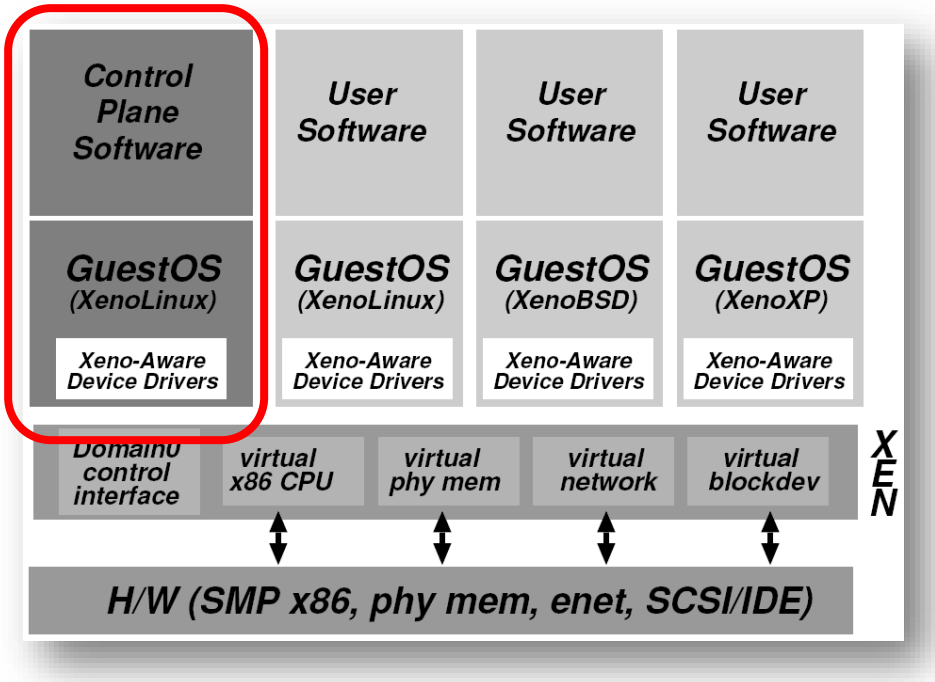
Trusted OS (Linux) runs in own domain (Domain0), manages privileged operations and has direct access to the hardware.

Early versions used "*paravirtualization*", a fancy word for *"we have to modify & recompile OS."*

Most recent version does not require OS mods.

Because of Intel/AMD hardware support

Commercialized via XenSource, but also open source



VMware Workstation

VMware Workstation uses hosted model.

Free version for MacOS.

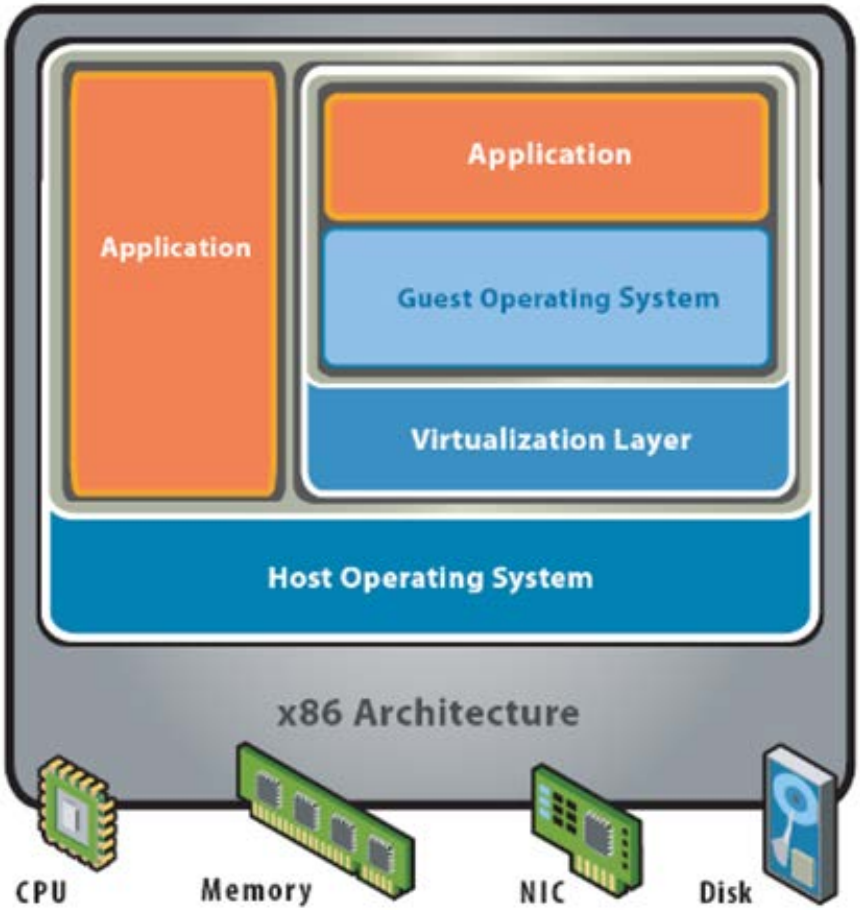
VMM runs unprivileged, installed on base OS and relies upon base OS for device functionality.

The entire virtualization runs as an application process.

Most flexible. Allows booting of whole operating system as an application.

Several levels of indirection.

Slow but better than dual-booting for a personal machine.



What needs to be virtualized?

Exactly what you would expect:

- CPU
- Events
- Memory
- I/O devices

Isn't this just duplicating OS functionality?

- Yes and no.
- Approaches will be similar to what OS does.
  - Simpler functionality (VMM much smaller than OS).
- But implements a different abstraction:
  - Hardware interface vs. OS interface.

Virtualizing Memory

OS assumes full control over memory.

But VMM partitions memory among VMs.

VMM needs to control mappings for isolation.

OS can only map to a physical page given to it by VMM.

Solution: Need MMU support to handle two-levels of page tables.

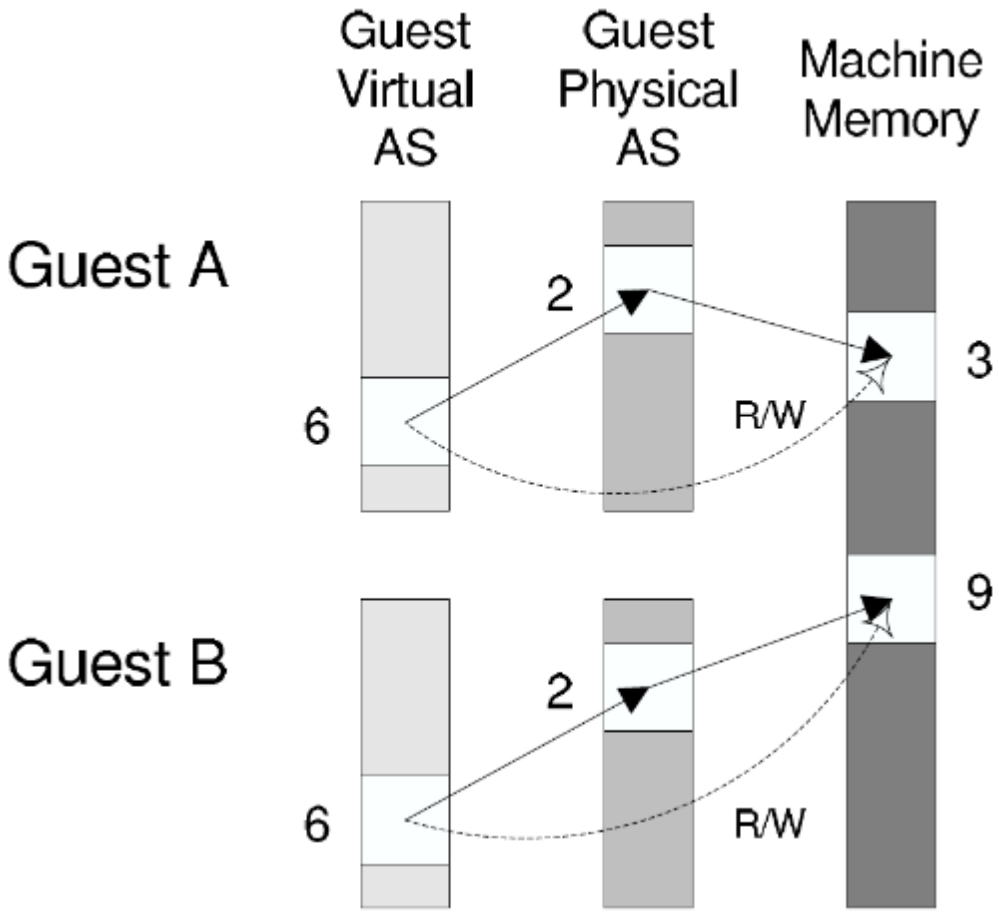
Shadow Page Tables

Three abstractions of memory.

Machine: actual hardware memory, e.g., 2 GB of DRAM.

Physical: The memory given to each guest operating system to be managed by it.

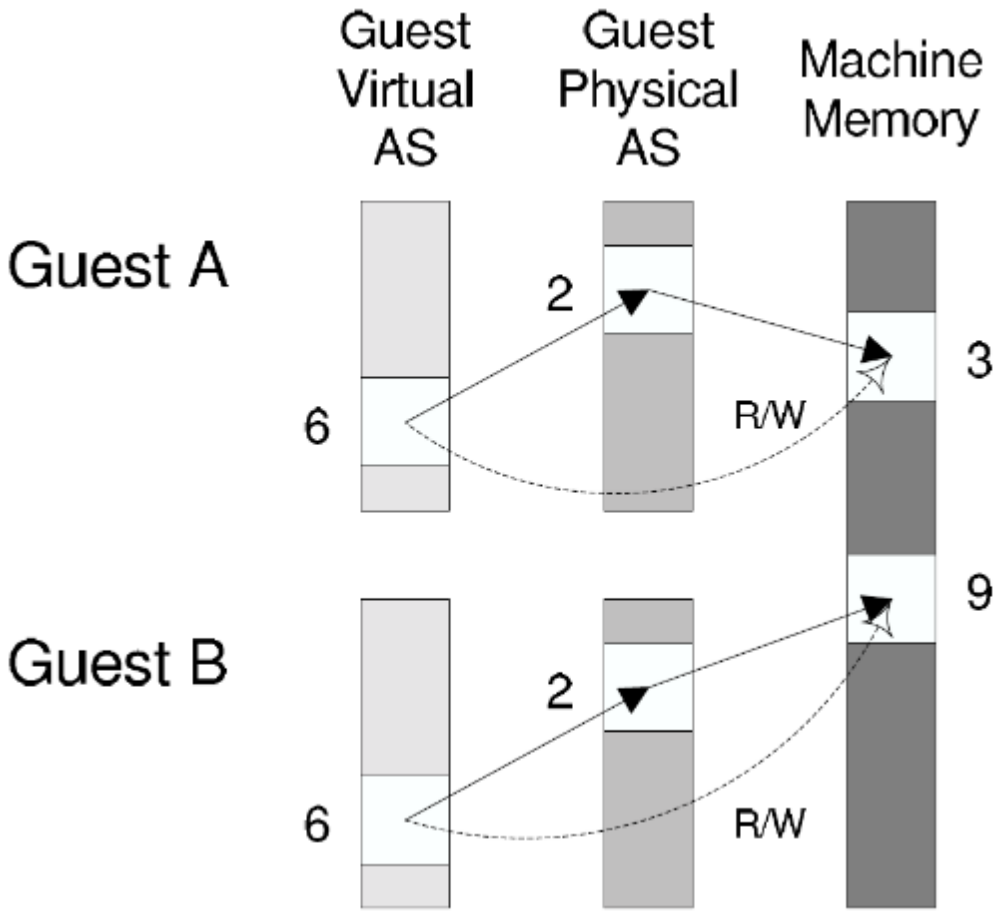
Virtual: virtual address space per process.



Shadow Page Tables

If a VMM allocates 512 MB to a VM, the OS thinks the computer has 512 MB of contiguous physical memory.

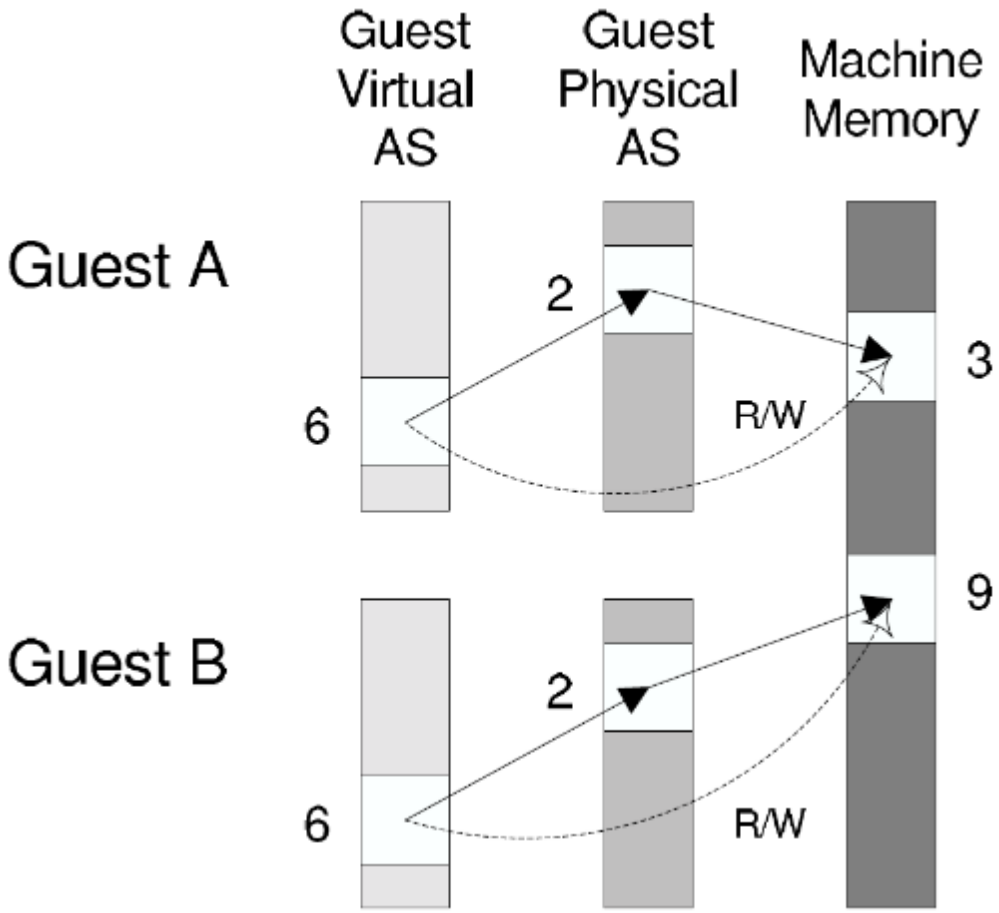
Underlying machine memory may be discontiguous.



Shadow Page Tables

In each VM, OS creates and manages page tables for its virtual address spaces without modification

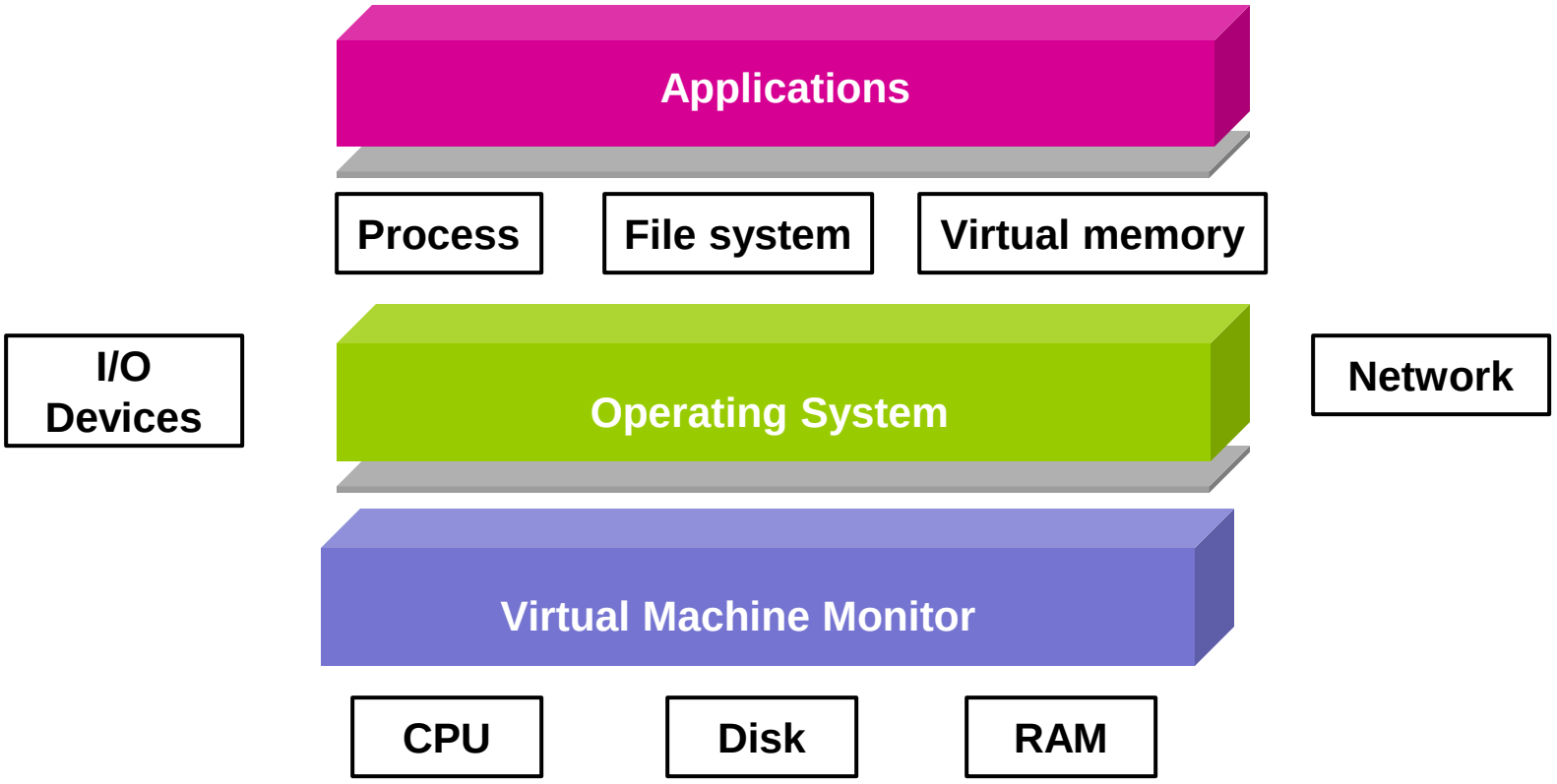
But these page tables *are not used* by the MMU hardware.



Agenda

1. Announcements.
2. Final exam.
3. Project 4.
4. Student evals.
5. Virtual machine monitors.
6. The big ideas in 482.

482 – The Big Ideas



482 – The Big Ideas

Abstraction: Virtualizing a resource.

- CPU → Thread
- Physical memory → Address space
- Disk → File system

Concurrency and consistency.

- Ordering, atomicity, and transactions.

482 – The Big Ideas

Caching and exploiting locality.

Memory as a cache for disk + LRU eviction.

Indirection.

Gains power, hurts performance.

Recover performance via caching.

Multi-level paging + TLB, inode map in the Linux filesystem.

Tolerating faults through redundancy.

RAID, replication.

The end.

Thank you.

It has been my honor to be your instructor.

I could not have done it without my wonderful staff, Austin, Brandon and Celine.

Special thanks to Harsha Madhyastha for his slides and to Ben Reeves for help setting up the course.

I know this course has been an enormous amount of work for all you. I hope you've enjoyed it and that it helps you in your career objectives.

I hope to see some of you again, perhaps in my EECS 440 W21 System Design of a Search Engine class.

I wish you all great success.