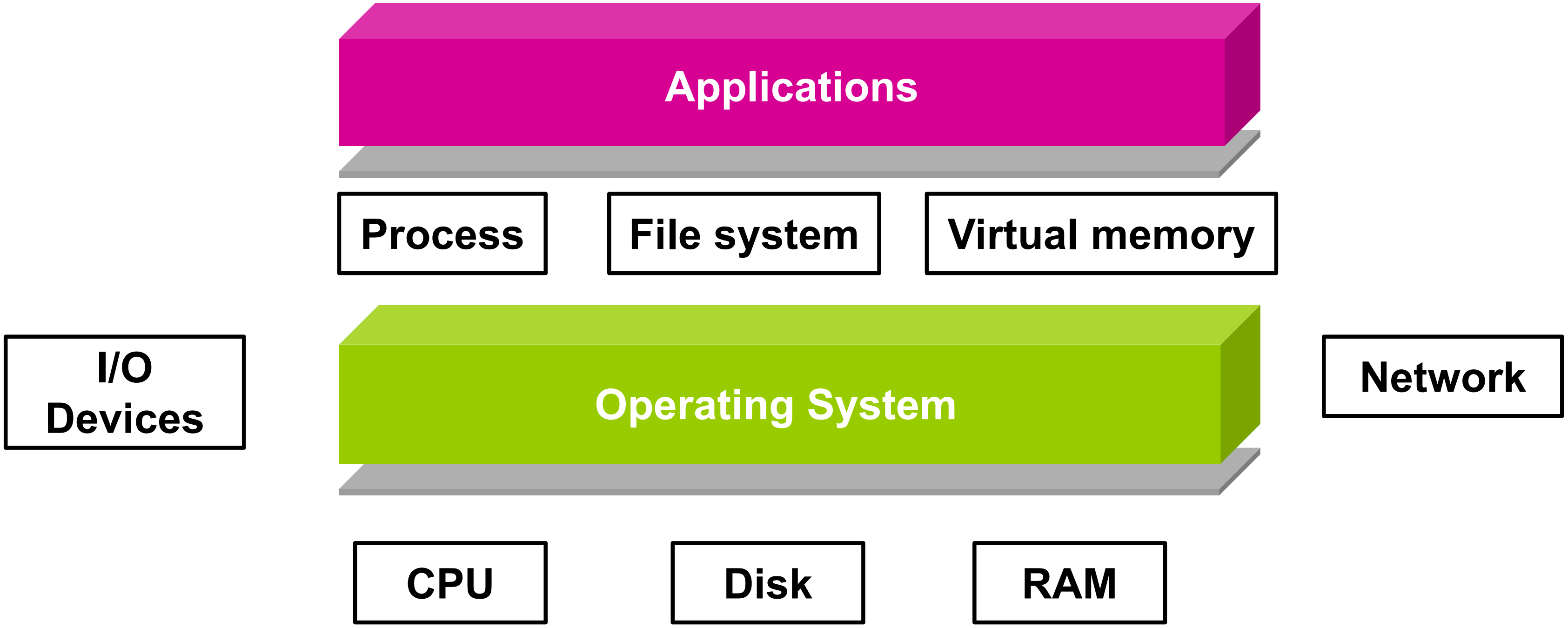
EECS 482
Introduction to Operating Systems

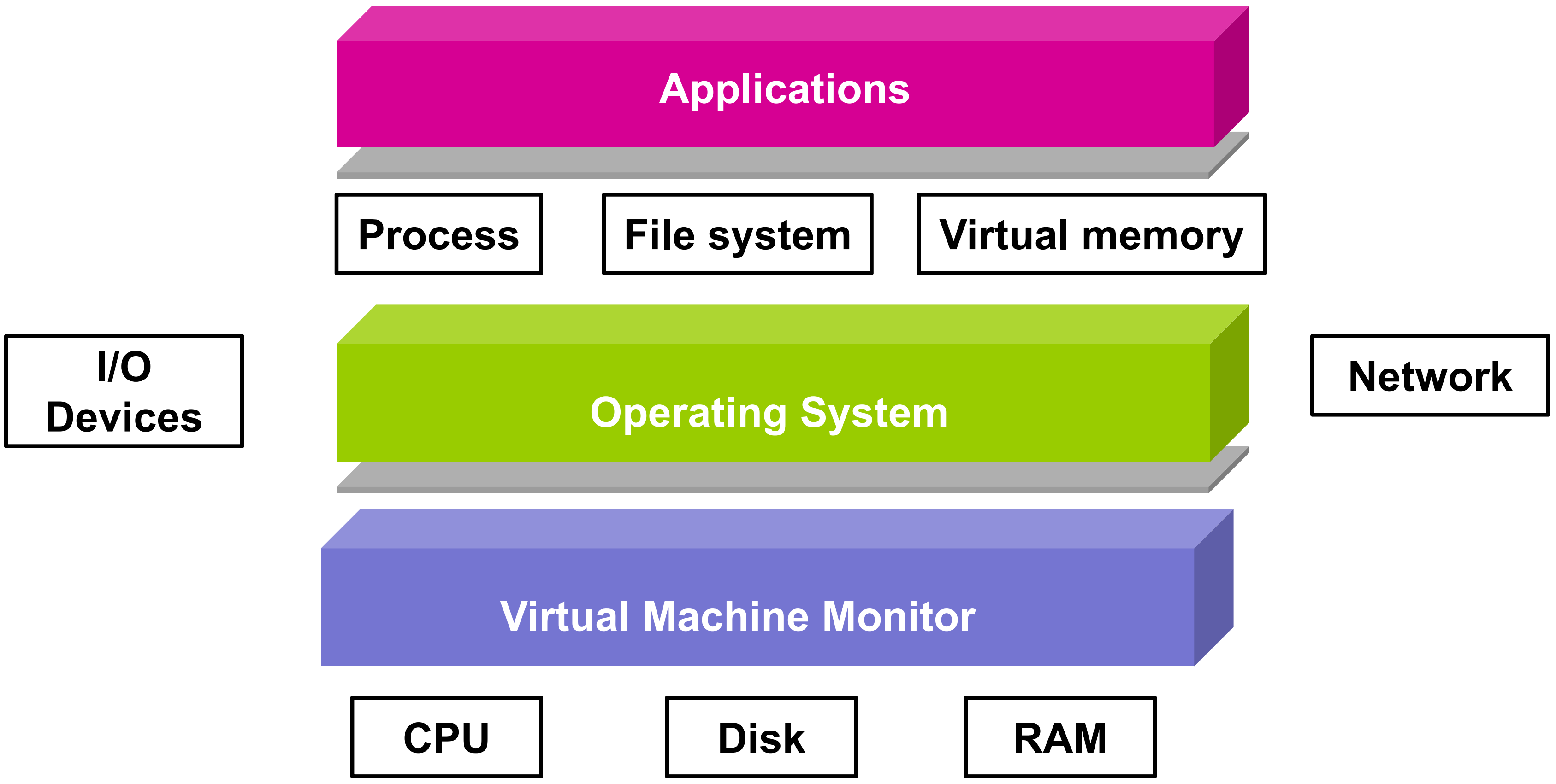
Winter 2018

Harsha V. Madhyastha

OS Abstractions



Virtual Machine Monitor



Applications

Process

File system

Virtual memory

I/O Devices

Operating System

Network

Virtual Machine Monitor

CPU

Disk

RAM

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?

- Resource utilization
  - Machines today are powerful, multiplex their hardware
    - » Example: Cloud services
  - Migrate VMs across machines without shutdown

- Software use and development
  - Can run multiple OSes simultaneously
    - » No need to dual boot
  - Can do system (e.g., OS) development at user-level

- Many other cool applications
  - Debugging, emulation, security, fault tolerance, …

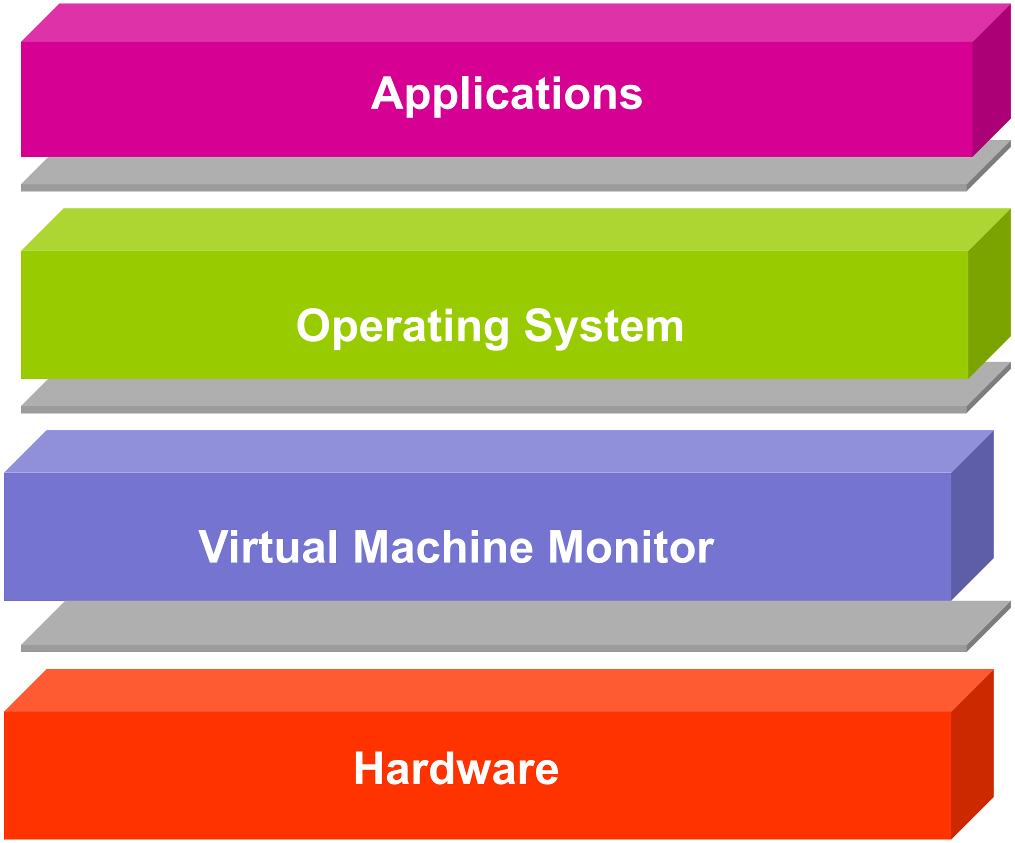
Example of Cool VMM Tricks

- 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

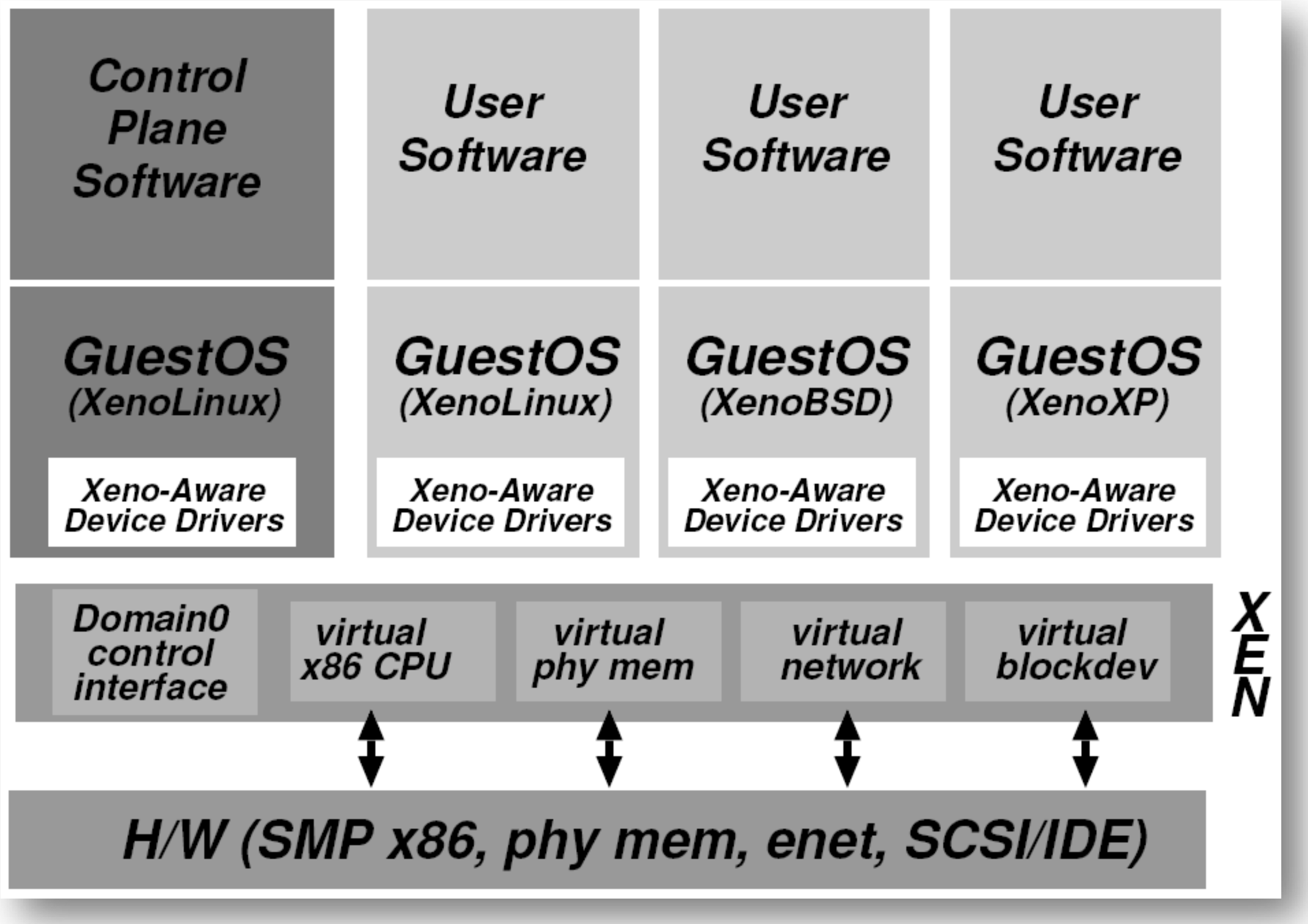
VMM Requirements

- Fidelity
  - OSes and applications work without modification
    - » (although 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

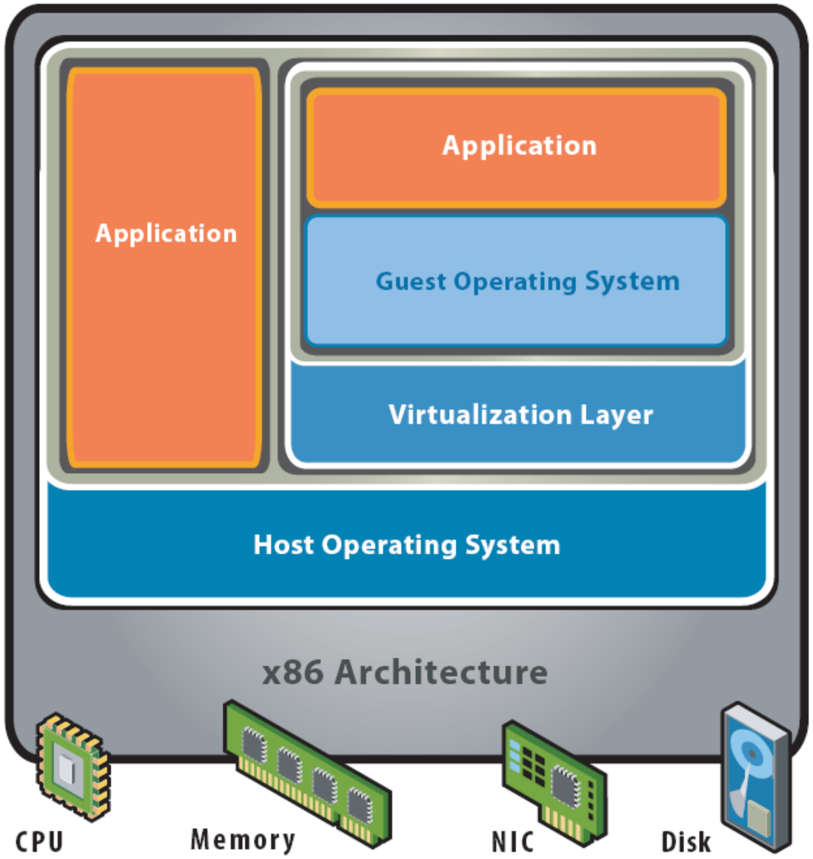
VMware Hypervisor Model



Xen Architecture



VMware Hosted Architecture



Hosted Architecture

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

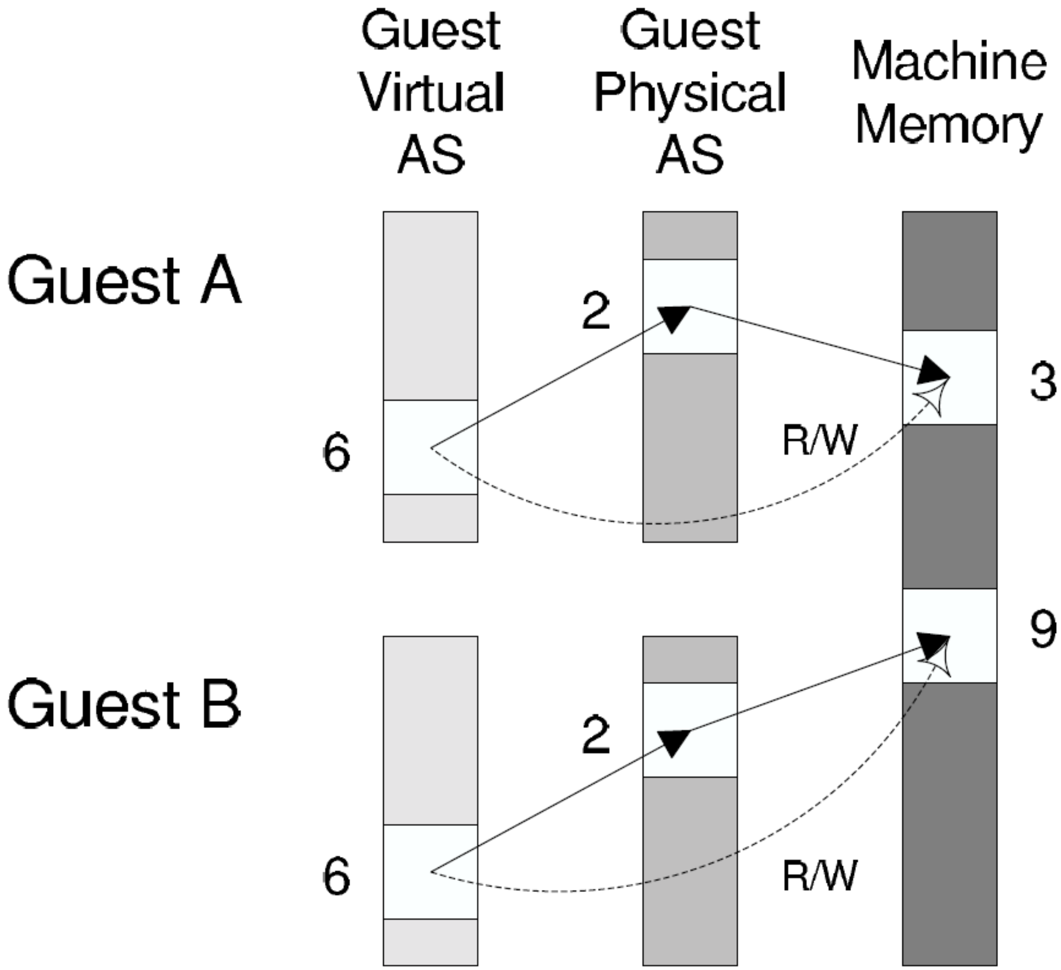
Virtualizing Memory

- OS assumes full control over memory

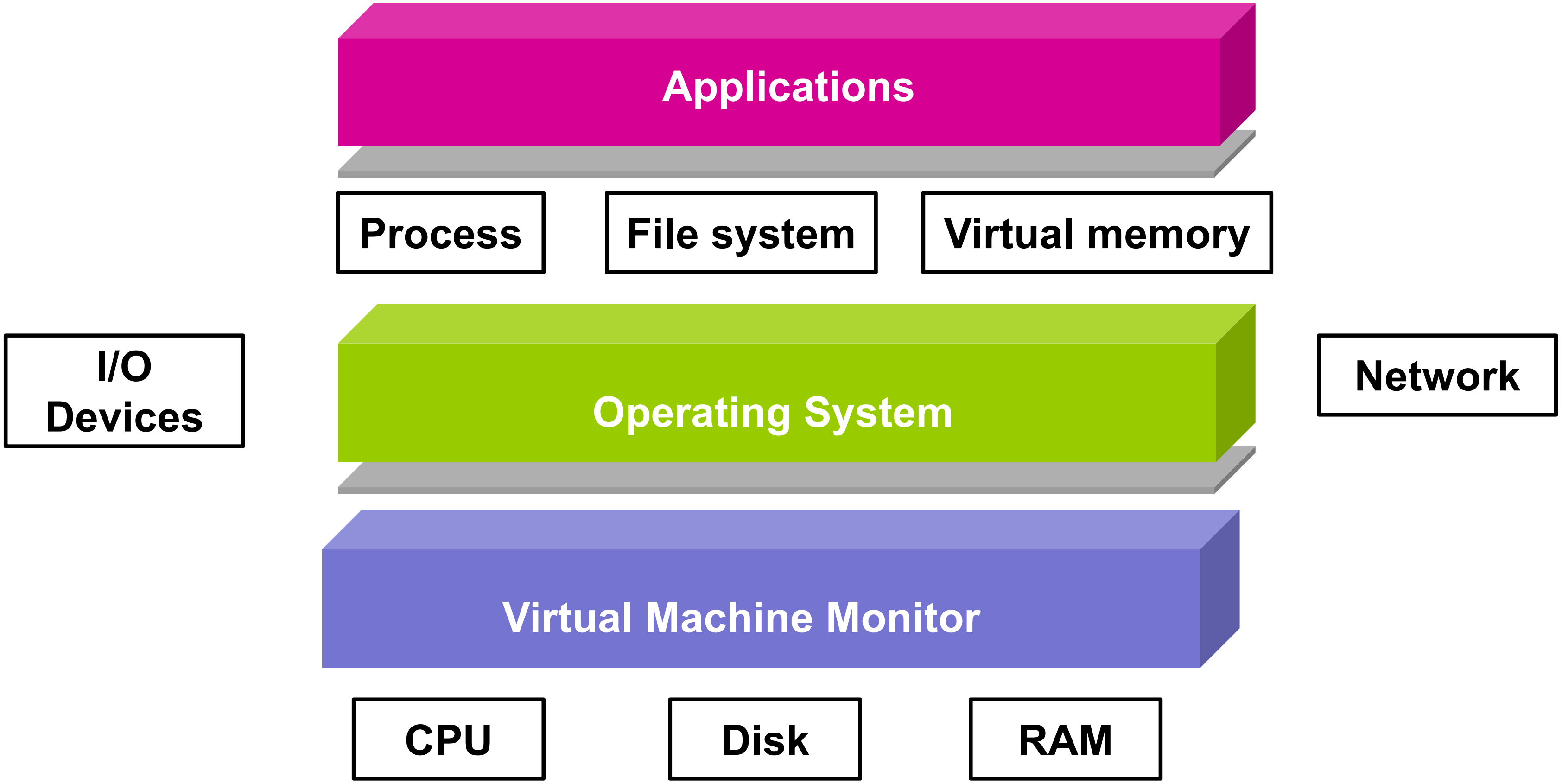
- But VMM partitions memory among VMs
  - VMM needs to control mappings for isolation
    - » OS can only map to a physical page given to it by VMM

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

Shadow Page Tables



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 LFS
- Tolerating faults through redundancy
  - RAID, replication

Reminders

- Submit peer feedback for Project 4
- Submit teaching evaluations

- Final exam: 7-9pm next Monday (April 23rd)
  - Email me if you have a conflict
  - Monitor Piazza for room assignment

- Solve sample exams before review session
  - 12-3pm on April 21st in CHRYS 220

Final exam details

- Closed book, closed notes
- No computers, phones, calculators, etc.

- 2 hr. exam – start at 7pm (not 7:10pm!)

- Focus is on virtual memory to dist. systems
- Includes projects 3 and 4
- But first-half topics may be needed

How to study?

- Review **all** of project 3 and project 4
- A lot of lecture material **not** in the projects

- Do sample exams, time yourself
  - Reflect on midterm strengths/weaknesses
- Redo all the discussion questions

- Study groups: ask each other questions
  - Textbook is a good source of questions

Exam-taking tips

- Skim problems – answer easiest first
- Read coding questions carefully
  - Think and design before writing code
- Don’t get bogged down by any 1 question
  - Stuck? Answer part of the question well
  - Can get partial credit even on tough questions
- Write down assumptions
  - May not get full credit, helps with partial credit
- Familiarity helps you avoid time pressure

- Good luck for the final!