
NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE
CE/CZ4052
Cloud Computing
Virtualization in Cloud
Dr. Tan, Chee Wei
Email: cheewei.tan@ntu.edu.sg
Office: N4-02c-104

Outline

- Concepts
- Virtualization architecture
- CPU and OS basics
- Types of CPU virtualization
- Cloud infrastructures

What is virtualization?

- Virtualization is a broad term. It can be applied to all types of resources (CPU, memory, network, etc.)

- Allows one computer to “look like” multiple computers, doing multiple jobs, by sharing the resources of a single machine across multiple environments.

History

- Believe it or not:
  - virtualization started in 1960's in IBM's mainframe

Virtualization

'Nonvirtualized' system

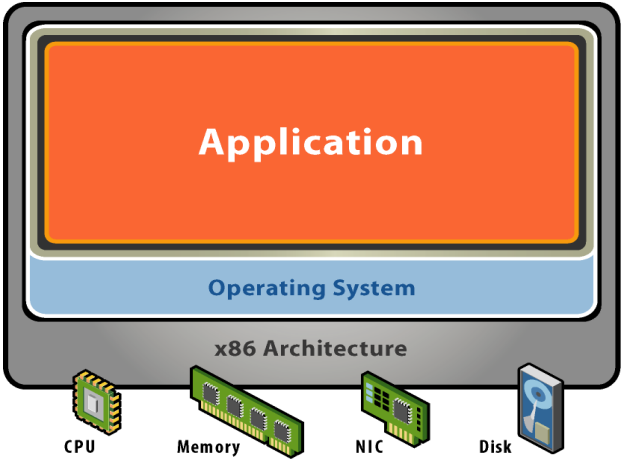
A single OS controls all hardware platform resources

Virtualized system

It makes it possible to run multiple Virtual Machines on a single physical platform

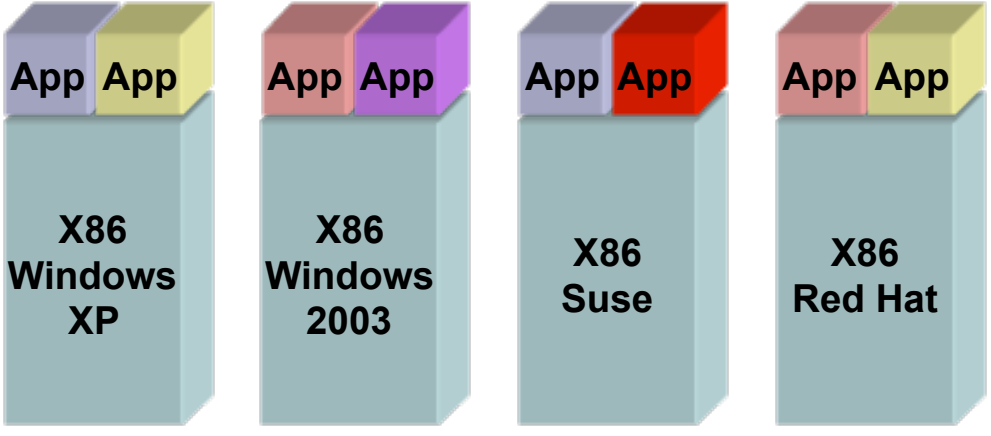
The old model

- A server for every application
- Software and hardware are tightly coupled



The old model

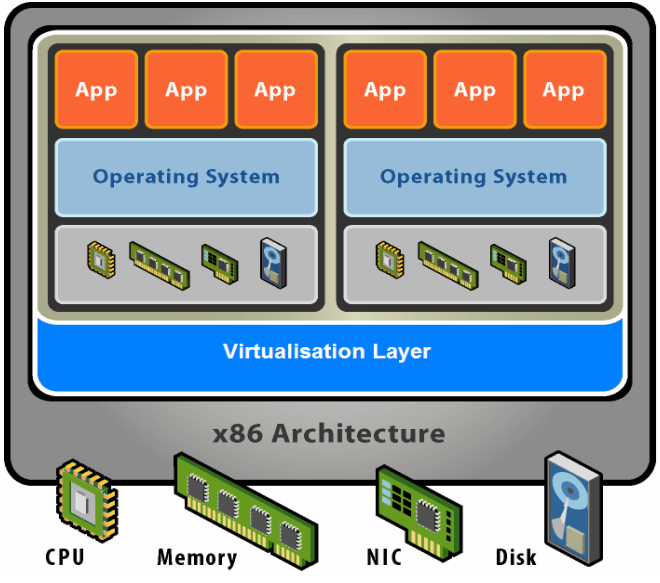
- Big disadvantage: low utilization



The new model

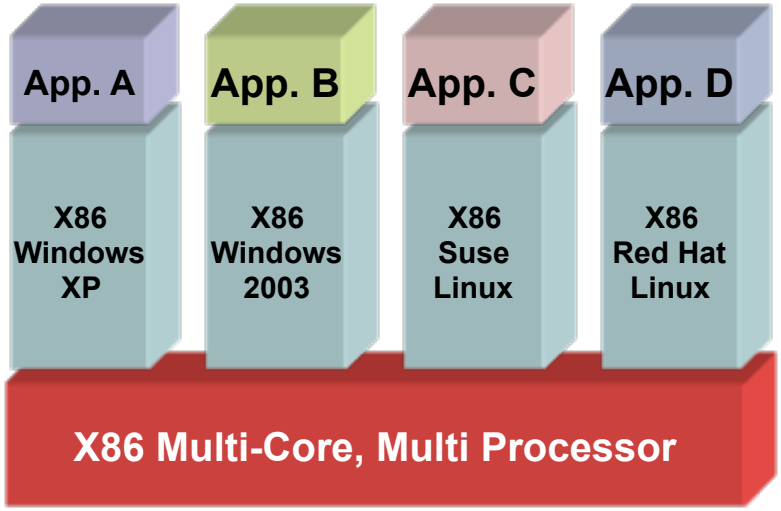
- Physical resources are virtualized. OS and applications as a single unit by encapsulating them into *virtual machines*

- Separate applications and hardware

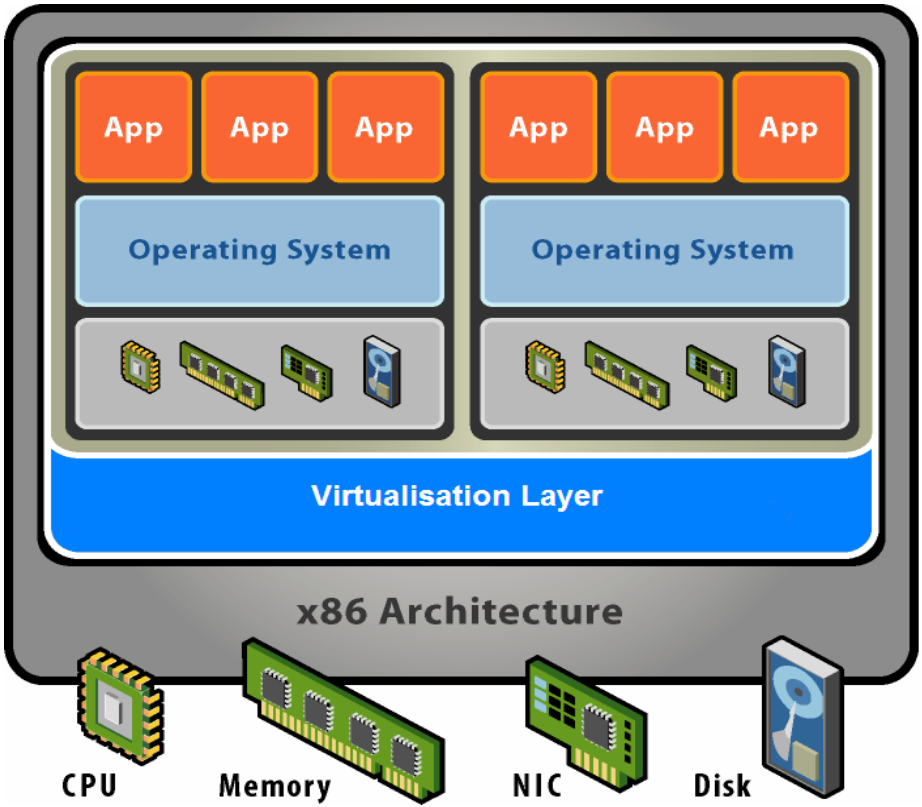


The new model

- Big advantage: improved utilization



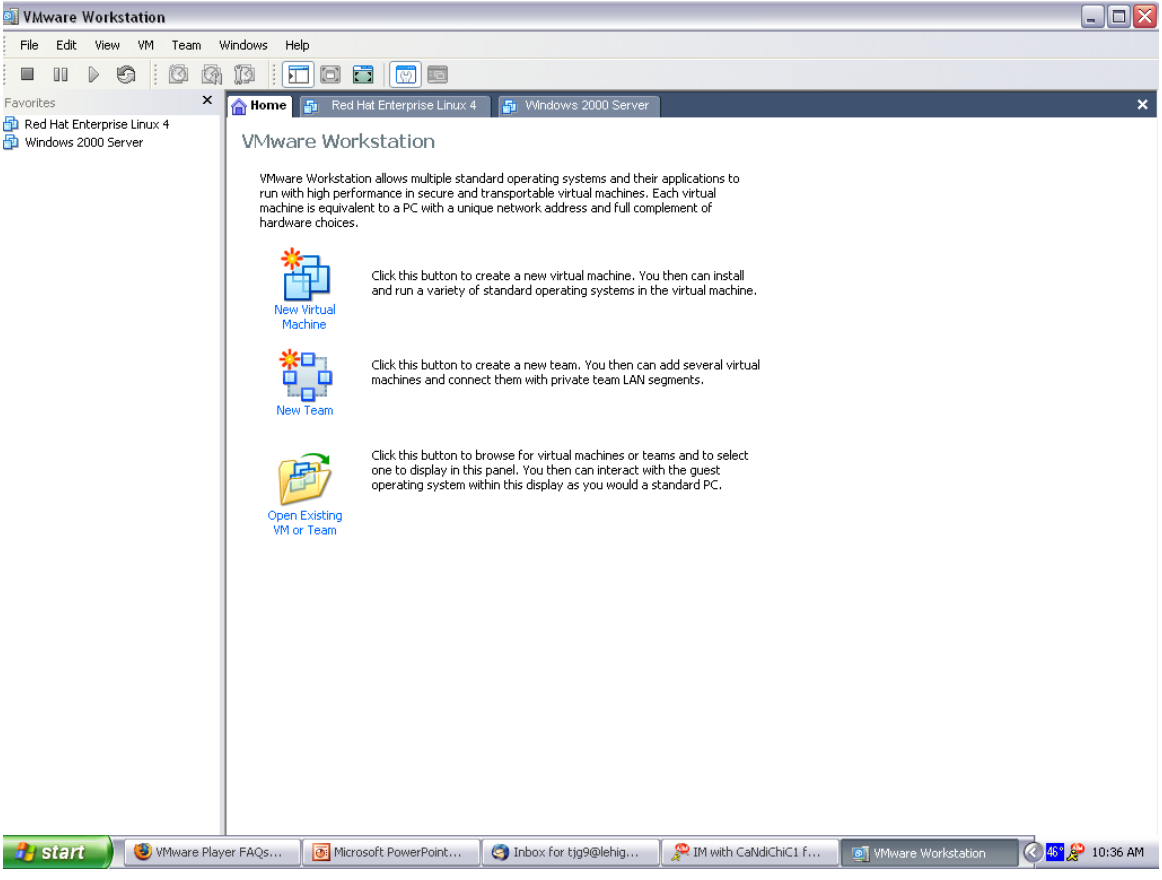
Some terms


App
App
App
App
App
App
Operating System
Operating System
Virtualisation Layer
x86 Architecture
CPU
Memory
NIC
Disk

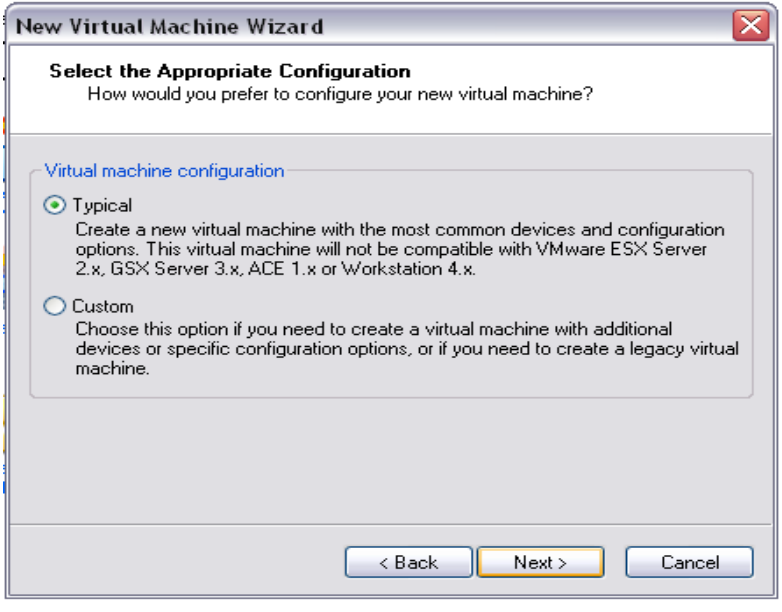
Hosted architecture

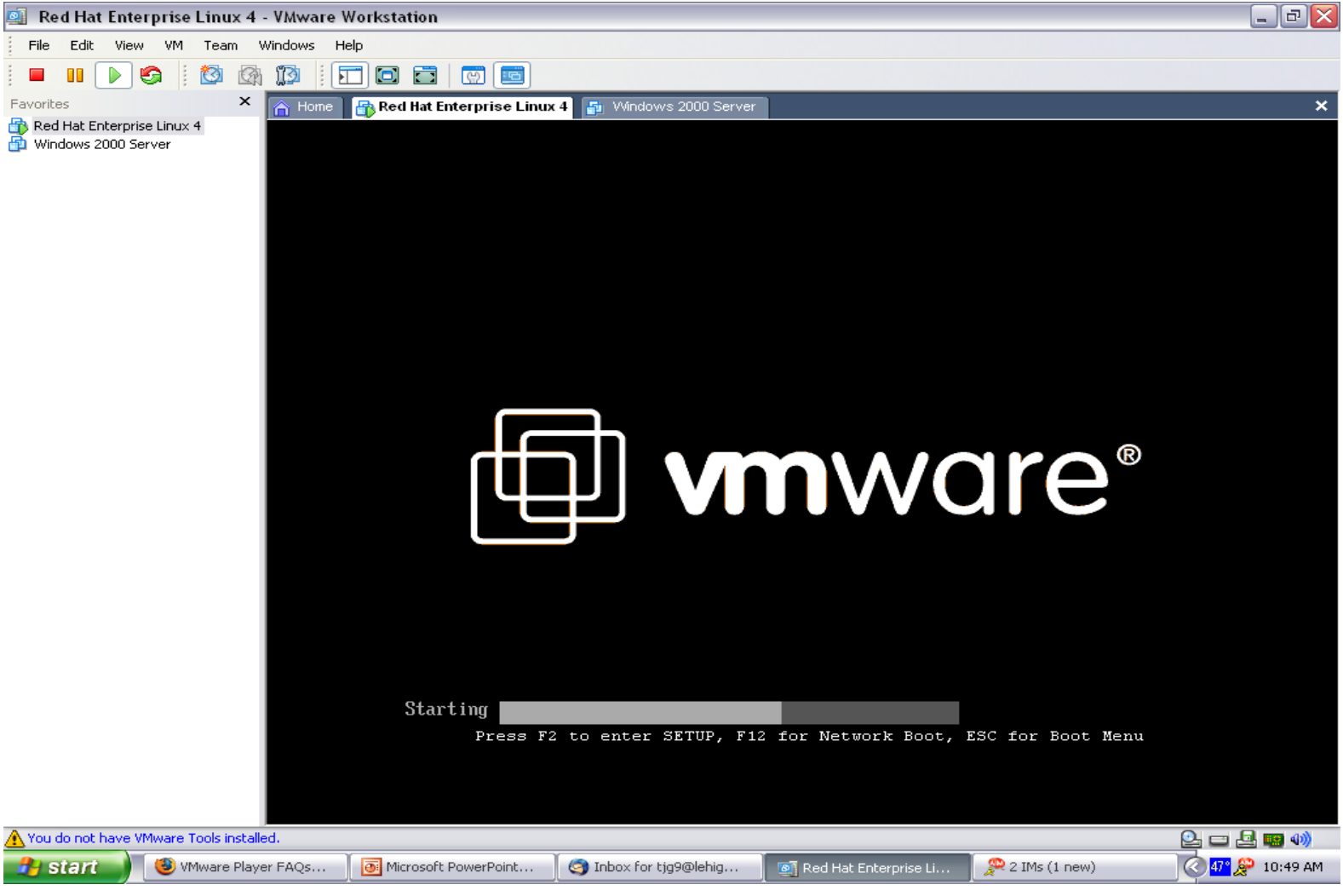
- A hosted architecture installs and runs the virtualization layer as an application on top of an operating system

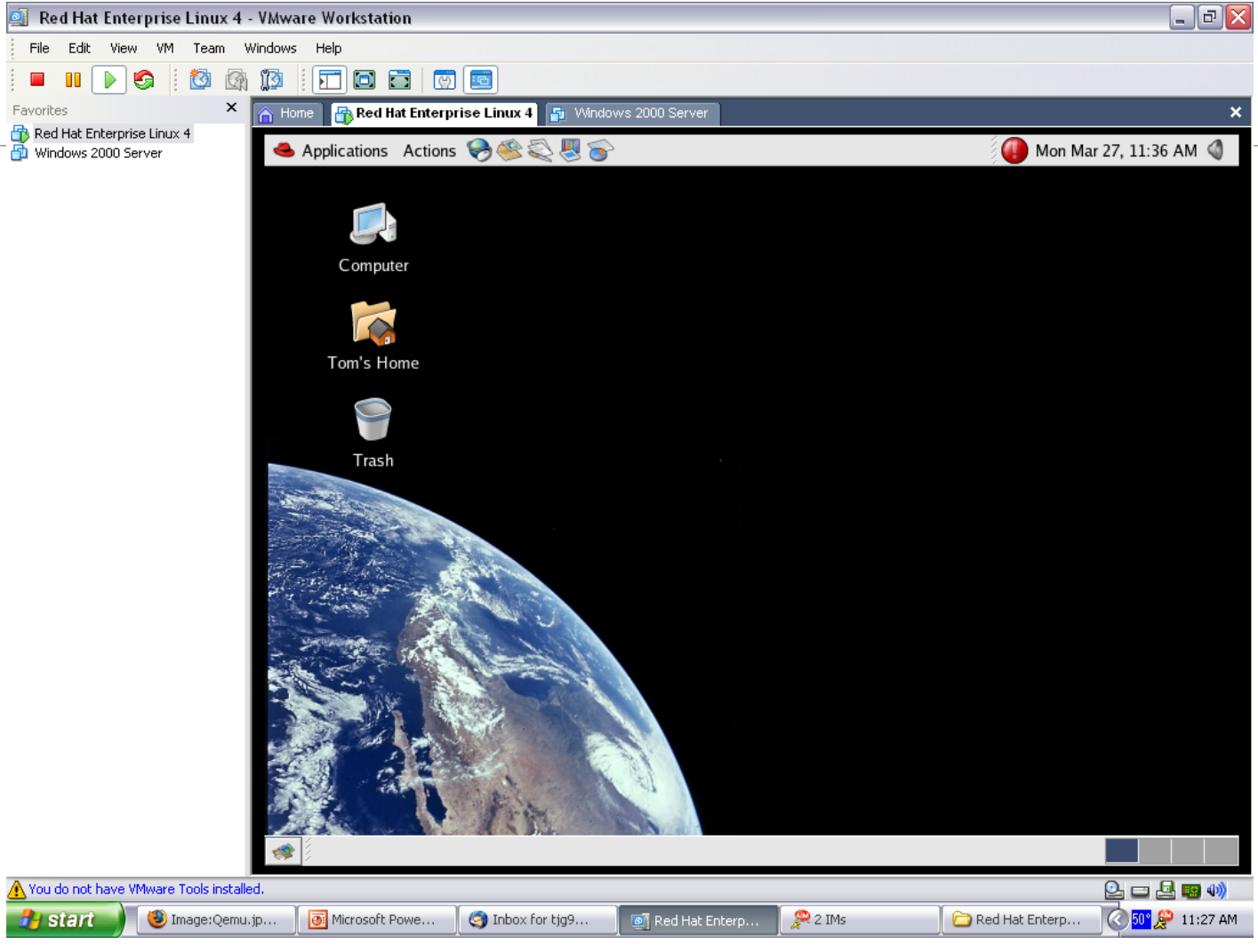
Hosted architecture example




New Virtual Machine Wizard
Workstation
5
Welcome to the New Virtual Machine Wizard
This wizard will guide you through the steps of creating a new virtual machine.
< Back
Next >
Cancel


New Virtual Machine Wizard
Select the Appropriate Configuration
How would you prefer to configure your new virtual machine?
Virtual machine configuration
Typical
Create a new virtual machine with the most common devices and configuration options. This virtual machine will not be compatible with VMware ESX Server 2.x, GSX Server 3.x, ACE 1.x or Workstation 4.x.
Custom
Choose this option if you need to create a virtual machine with additional devices or specific configuration options, or if you need to create a legacy virtual machine.
< Back
Next >
Cancel


Red Hat Enterprise Linux 4 - VMware Workstation
File Edit View VM Team Windows Help
Favorites
Red Hat Enterprise Linux 4
Windows 2000 Server
Home
Red Hat Enterprise Linux 4
Windows 2000 Server
vmware®
Starting
Press F2 to enter SETUP, F12 for Network Boot, ESC for Boot Menu
You do not have VMware Tools installed.
start
VMware Player FAQs...
Microsoft PowerPoint...
Inbox for tjg9@lehig...
Red Hat Enterprise Li...
2 IMs (1 new)
10:49 AM


Red Hat Enterprise Linux 4 - VMware Workstation
File Edit View VM Team Windows Help
Favorites
Red Hat Enterprise Linux 4
Windows 2000 Server
Home
Red Hat Enterprise Linux 4
Windows 2000 Server
Applications Actions
Mon Mar 27, 11:36 AM
Computer
Tom's Home
Trash
You do not have VMware Tools installed.
start
Image:Qemu.jp...
Microsoft Powe...
Inbox for tjg9...
Red Hat Enterp...
2 IMs
Red Hat Enterp...
11:27 AM

Hosted architecture

- Indirect access to hardware through the host OS
  - performance penalty, usually for desktops and personal use

Hypervisor architecture

- The hypervisor architecture installs the virtualization layer, called **hypervisor**, directly on a clean x86-based system
  - Installer is usually an ISO installing a tailor-made OS

Hypervisor architecture

- It has direct access to hardware resources
  - A hypervisor is more efficient than a hosted architecture and delivers greater scalability, robustness, and performance
  - For production use

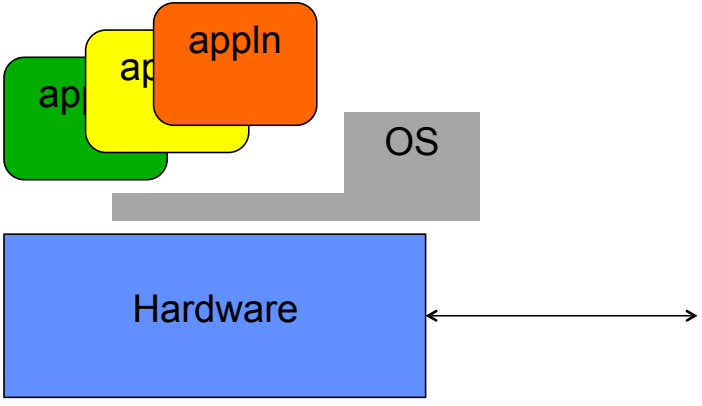
CPU virtualization

OS review

Credit: Prof. John Kubiatowicz's slides for CS162, Spring 2015, UC Berkeley

What is an operating system?

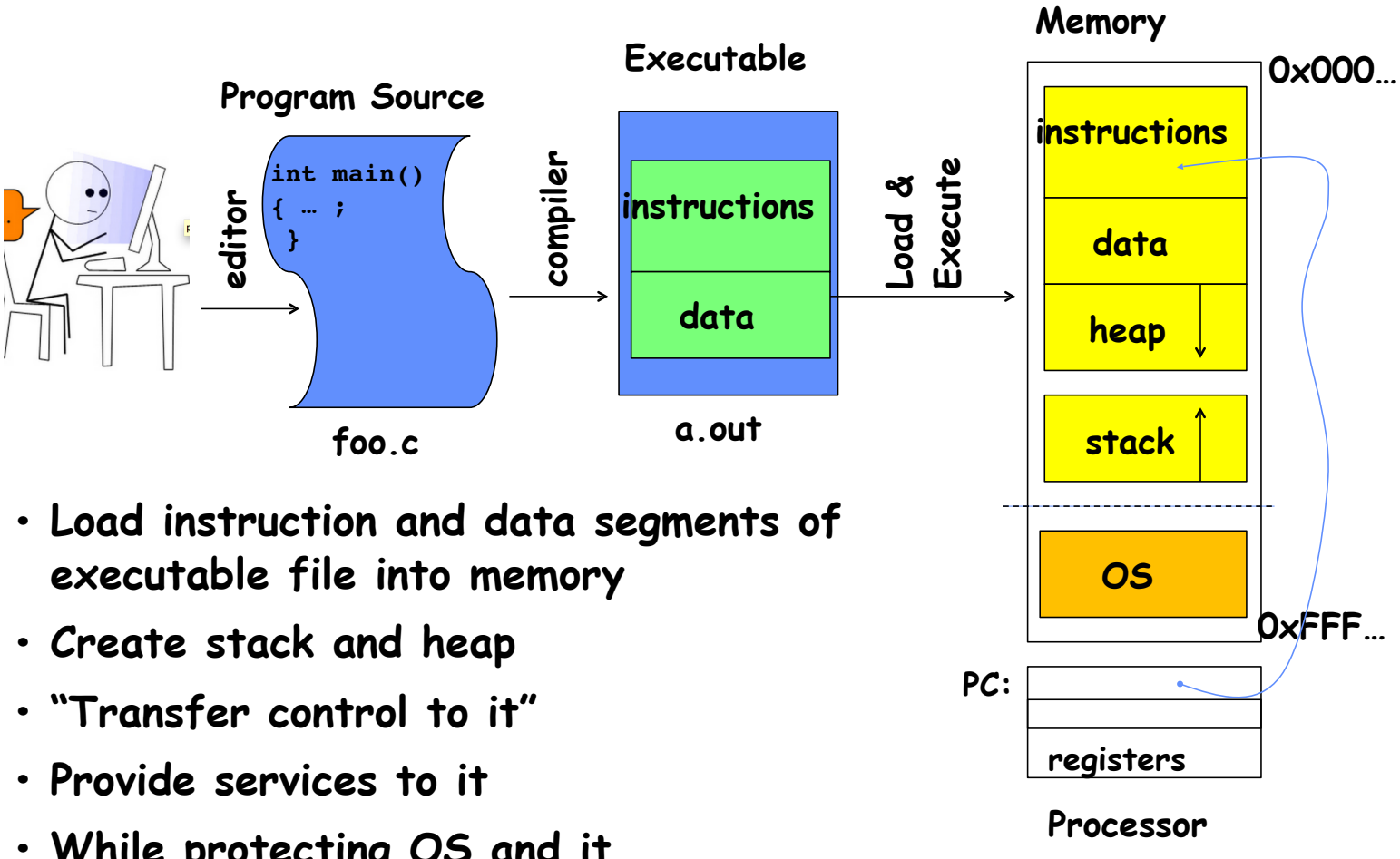
- Special layer of software that provides application software access to hardware resources
  - Convenient abstraction of complex hardware devices
  - Protected access to shared resources
  - Security and authentication
  - Communication amongst logical entities



Four fundamental OS concepts

- Thread
  - Single unique execution context
  - Program Counter, Registers, Execution Flags, Stack
- Address Space w/ Translation
  - Programs execute in an address space that is distinct from the memory space of the physical machine
- Process
  - An instance of an executing program is a process consisting of an address space and one or more threads of control
- Dual Mode operation/Protection
  - Only the "system" has the ability to access certain resources
  - The OS and the hardware are protected from user programs and user programs are isolated from one another by controlling the translation from program virtual addresses to machine physical addresses

OS Bottom Line: Run Programs



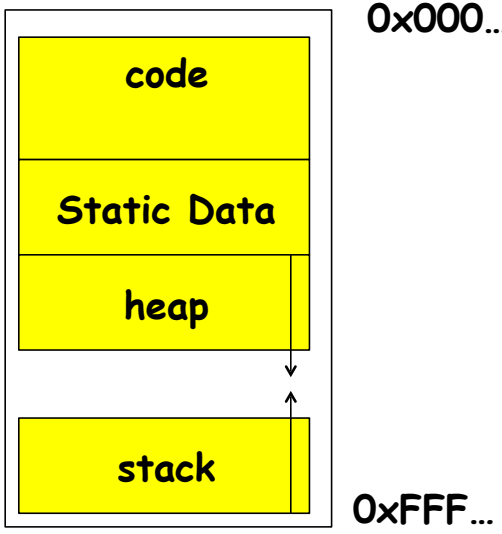
- Load instruction and data segments of executable file into memory
- Create stack and heap
- "Transfer control to it"
- Provide services to it
- While protecting OS and it

First OS Concept: Thread of Control

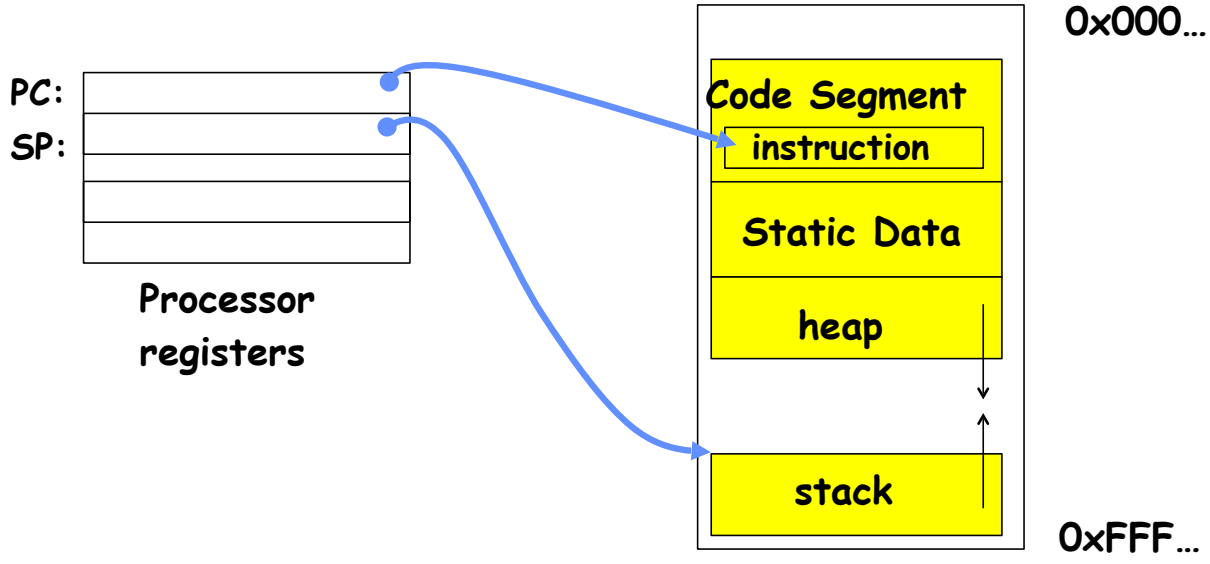
- Thread: Single unique execution context
  - Program Counter, Registers, Execution Flags, Stack
- A thread is executing on a processor when it is resident in the processor registers.
- PC register holds the address of executing instruction in the thread.
- Certain registers hold the context of thread
  - Stack pointer holds the address of the top of stack
    - » Other conventions: Frame Pointer, Heap Pointer, Data
  - May be defined by the instruction set architecture or by compiler conventions
- Registers hold the root state of the thread.
  - The rest is "in memory"

Second OS Concept: Program's Address Space

- Address space ⇒ the set of accessible addresses + state associated with them:
  - For a 32-bit processor there are 2^{32} = 4 billion addresses
- What happens when you read or write to an address?
  - Perhaps Nothing
  - Perhaps acts like regular memory
  - Perhaps ignores writes
  - Perhaps causes I/O operation
    - » (Memory-mapped I/O)
  - Perhaps causes exception (fault)

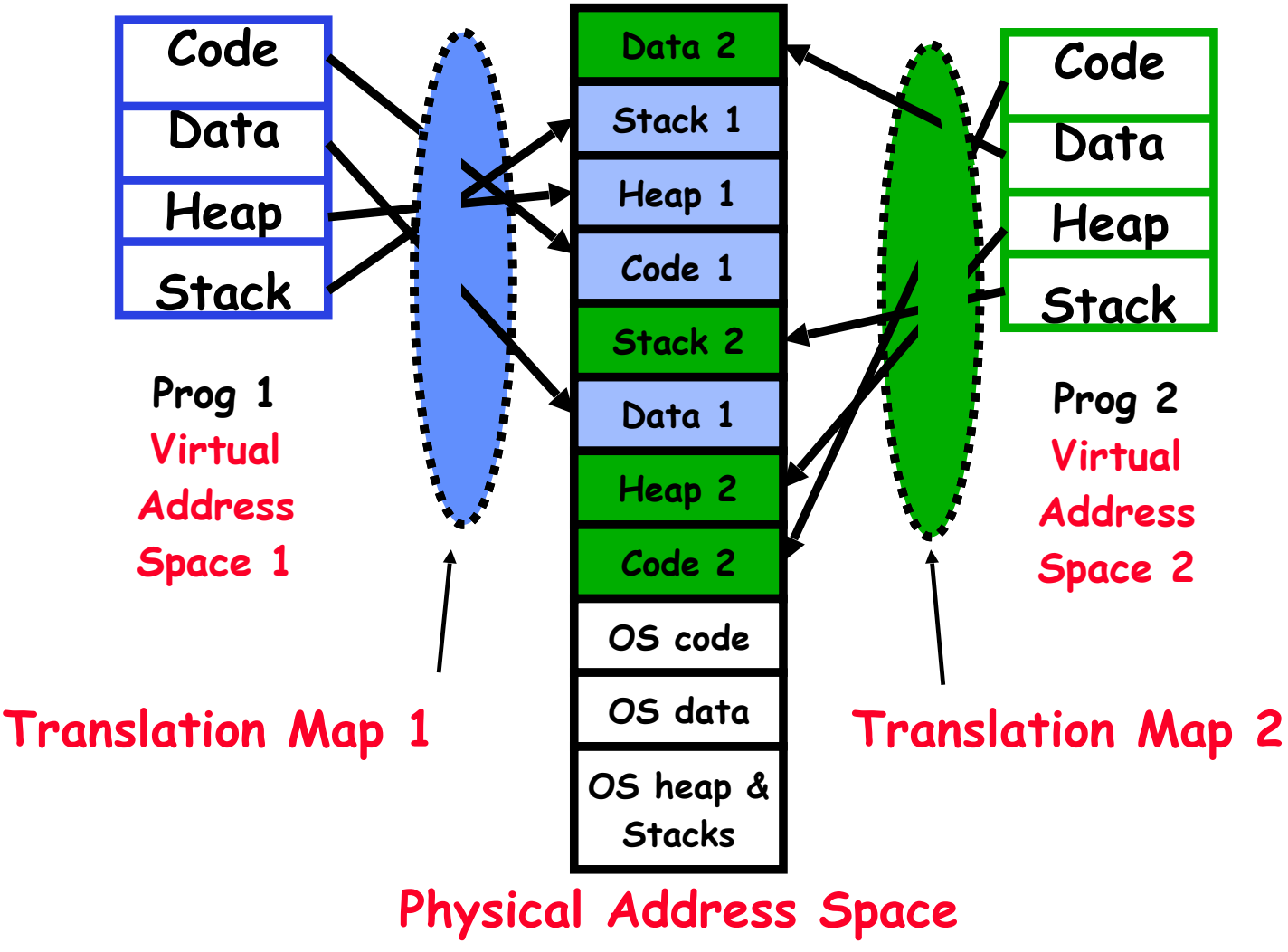


Address Space: In a Picture



- What's in the code segment? Data?
- What's in the stack segment?
  - How is it allocated? How big is it?
- What's in the heap segment?
  - How is it allocated? How big?

Providing Illusion of Separate Address Space: Load new Translation Map on Switch



Third OS Concept: Process

- Process: execution environment with Restricted Rights
  - Address Space with One or More Threads
  - Owns memory (address space)
  - Owns file descriptors, file system context, …
  - Encapsulate one or more threads sharing process resources
- Why processes?
  - Protected from each other!
  - OS Protected from them
  - Navigate fundamental tradeoff between protection and efficiency
  - Processes provides memory protection
- Application instance consists of one or more processes

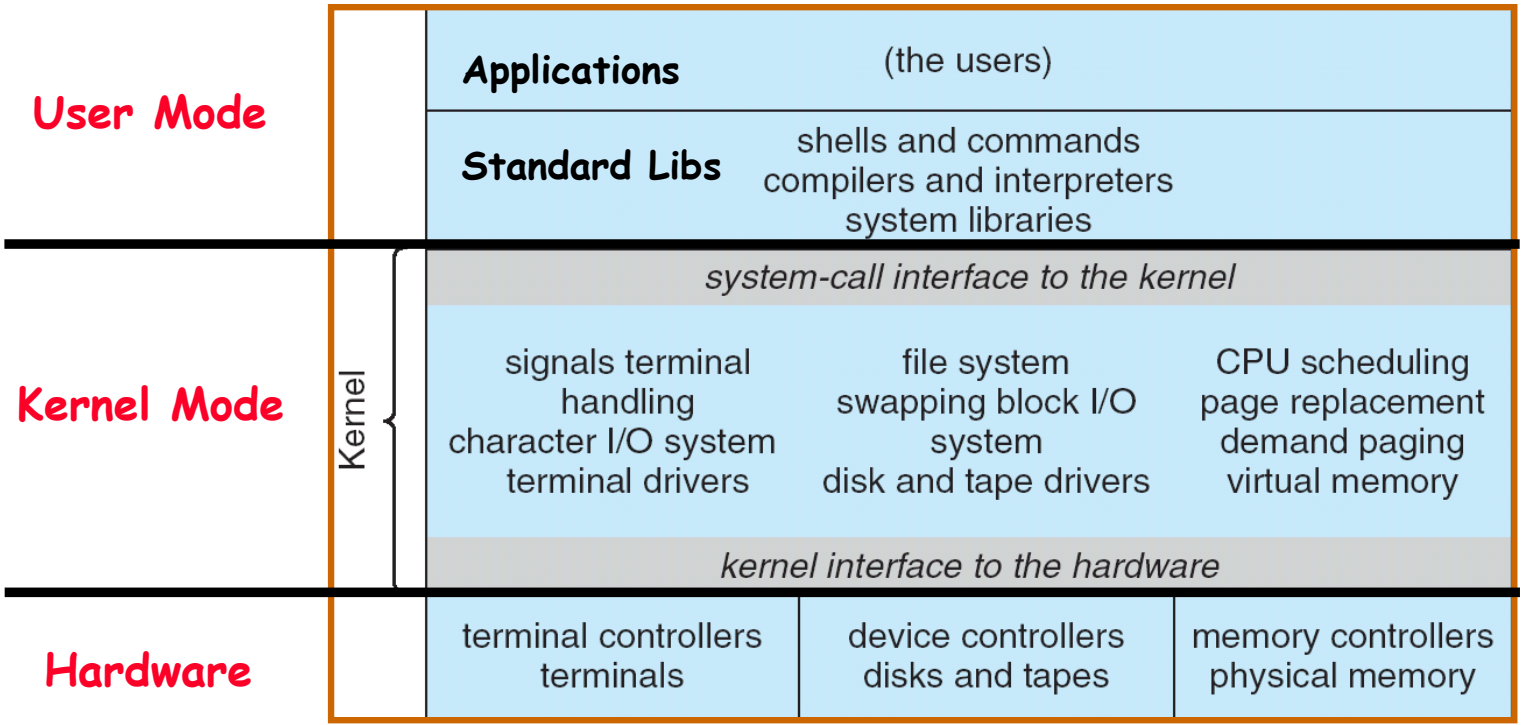
Protection

- Operating System must protect itself from user programs
  - Reliability: compromising the operating system generally causes it to crash
  - Security: limit the scope of what processes can do
  - Privacy: limit each process to the data it is permitted to access
  - Fairness: each should be limited to its appropriate share
- It must protect User programs from one another
- Primary Mechanism: limit the translation from program address space to physical memory space
  - Can only touch what is mapped in
- Additional Mechanisms:
  - Privileged instructions, in/out instructions, special registers
  - syscall processing, subsystem implementation
    - » (e.g., file access rights, etc)

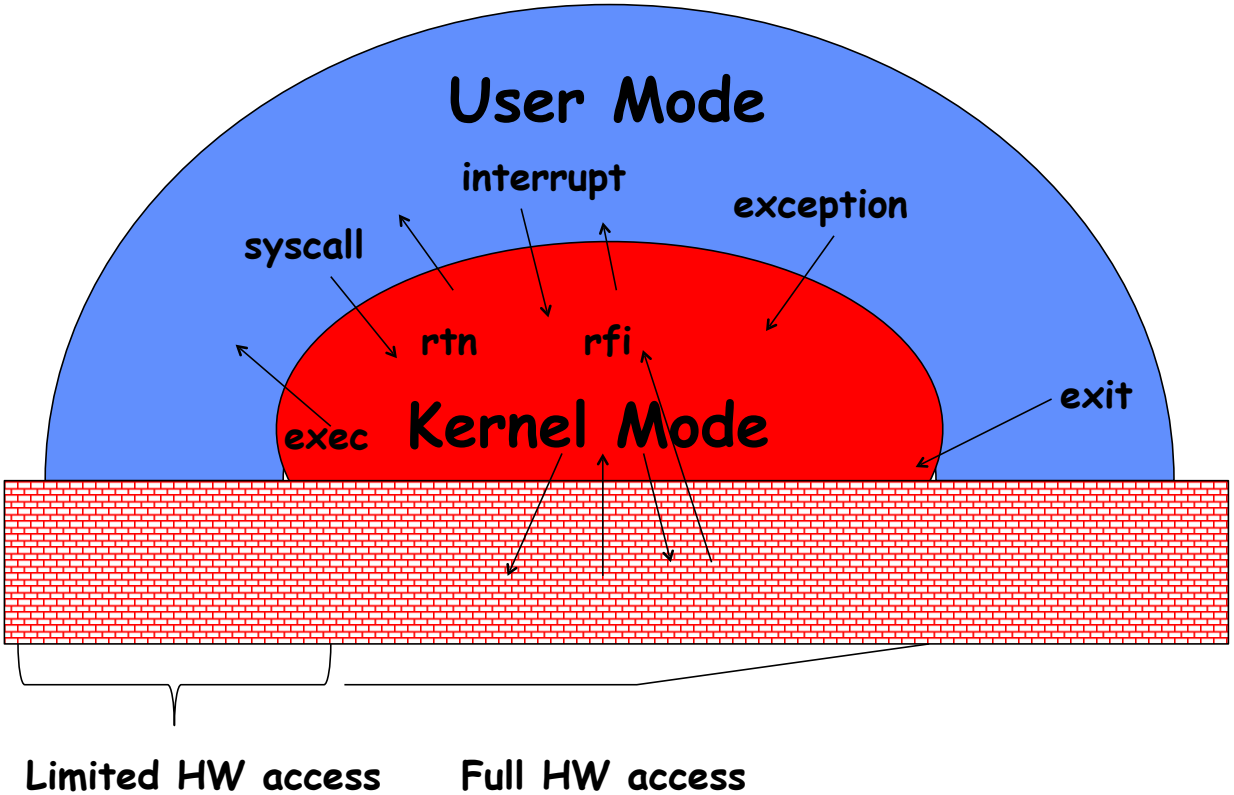
Fourth OS Concept: Dual Mode Operation

- Hardware provides at least two modes:
  - "Kernel" mode (or "supervisor" or "protected")
  - "User" mode: Normal programs executed
- What is needed in the hardware to support "dual mode" operation?
  - a bit of state (user/system mode bit)
  - Certain operations / actions only permitted in system/kernel mode
    - » In user mode they fail or trap
  - User->Kernel transition sets system mode AND saves the user PC
    - » Operating system code carefully puts aside user state then performs the necessary operations
  - Kernel->User transition clears system mode AND restores appropriate user PC
    - » return-from-interrupt

For example: UNIX System Structure

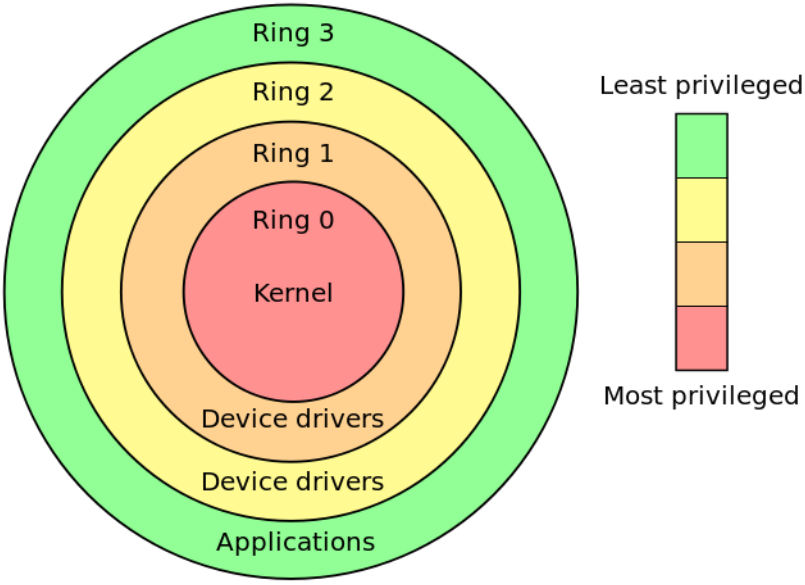


User/Kernel (Privileged) Mode



Protection rings

- Enforced in hardware in Intel x86 architectures



Source: Wikipedia

How to virtualize a CPU?

- Basically, the CPU does not care whether you are the guest OS or not
  - Have the PC point to somewhere in the RAM
- If it's unprivileged code, code that execute in userspace
  - It's safe to run no matter it is from the guest OS or host OS
  - Why?

What about privileged code?

- Currently, there’re 3 implementations
  - Full virtualization
  - Para virtualization
  - Hardware-assisted virtualization

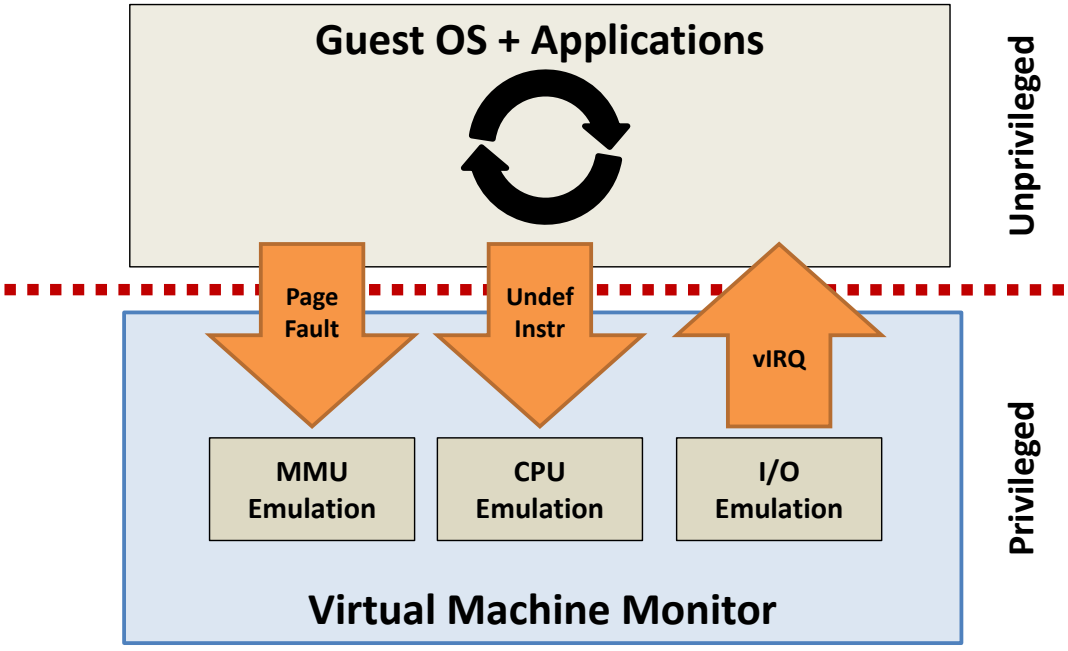
Types of CPU virtualization

Full virtualization

Ring 3	Guest applications
Ring 2	
Ring 1	Guest OS kernel
Ring 0	Hypervisor, Host OS
Host hardware	

Full virtualization

- Hardware is emulated by the hypervisor



Full virtualization

- The hypervisor presents a complete set of emulated hardware to the VM's guest operating system, including the CPU, motherboard, memory, disk, disk controller, and network cards.
- For example, Microsoft Virtual Server 2005 emulates an Intel 21140 NIC card and Intel 440BX chipset.
- Regardless of the actual physical hardware on the host system, the emulated hardware remains the same.

Full virtualization

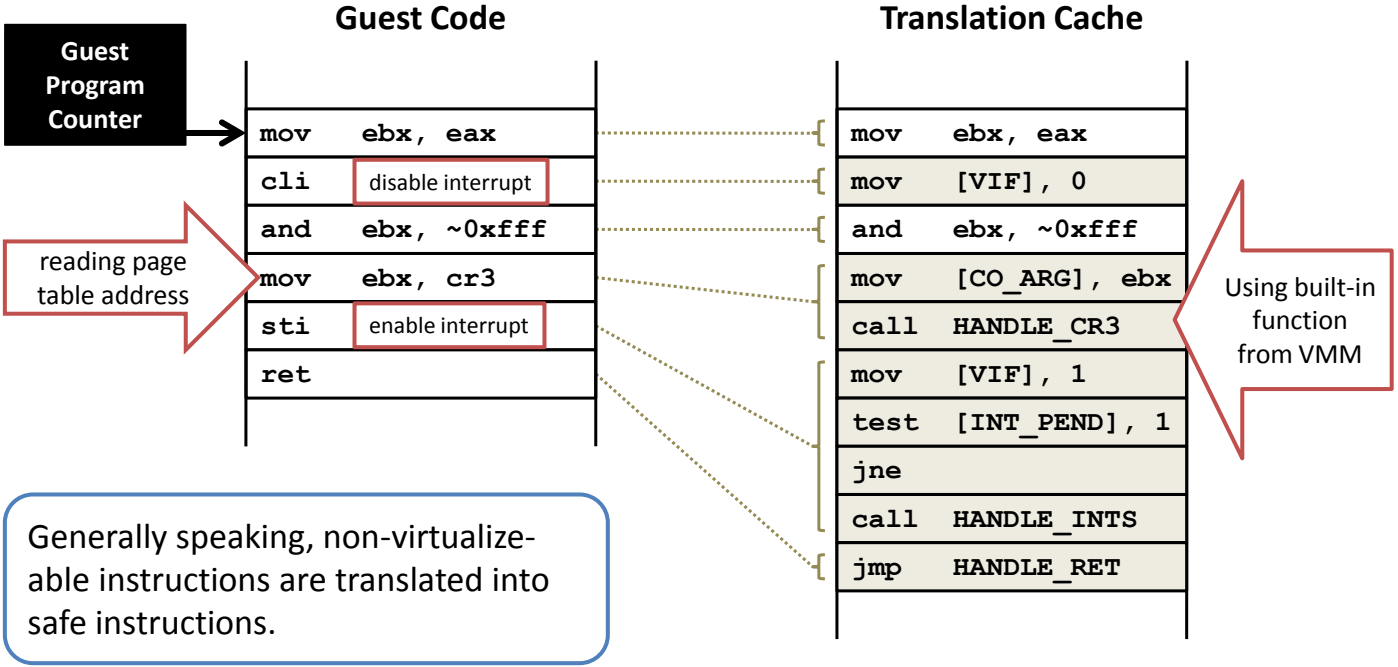
- Binary translation – step 1: trapping I/O calls

Ring 3	Guest applications
Ring 2	
Ring 1	Guest OS kernel
Ring 0	Hypervisor, Host OS
Host hardware	

whenever the guest OS asks for hardware, e.g. asking BIOS for a list of hardware, it's trapped by the hypervisor

Full virtualization

- Binary translation – step 2: emulate/translate



Full virtualization

- The guest OS is tricked to think that it's running privileged code in Ring 0, while it's actually running in Ring 1 of the host with the hypervisor emulating the hardware and trapping privileged code

- Unprivileged instructions are directly executed on CPU

Full virtualization

- Advantages:
  - Keeps the guest OS unmodified
  - Prevents an unstable VMs from impacting system performance; VM portability
- Disadvantages:
  - Performance is not good

Para-virtualization

- Developed to overcome the performance penalty of full virtualization with hardware emulation
- “Para” means “besides,” “with,”, or “alongside.”

Para-virtualization

Ring 3	Guest applications
Ring 2	
Ring 1	Modified Guest OS kernel
Ring 0	Hypervisor, Host OS
Host hardware	

Para-virtualization

- Can be done in two ways:
  - A recompiled OS kernel. Easy for Linux, Windows doesn't support
  - Paravirtualization drivers for some hardware, e.g. GPU, NIC

Para-virtualization

- Guest OS is aware that it runs in a virtualized environment. It talks to the hypervisor through specialized APIs to run privileged instructions.
- These system calls, in the guest OS, are also called “hypercalls.”
- Performance is improved. The hypervisor can focus on isolating VMs and coordinating.

Hardware-assisted

Non-root mode	Ring 3	Guest applications
	Ring 2	
	Ring 1	
	Ring 0	Guest OS kernel
Root mode	Ring -1	Hypervisor
Host hardware		

likely to emerge as the standard for server virtualization into the future

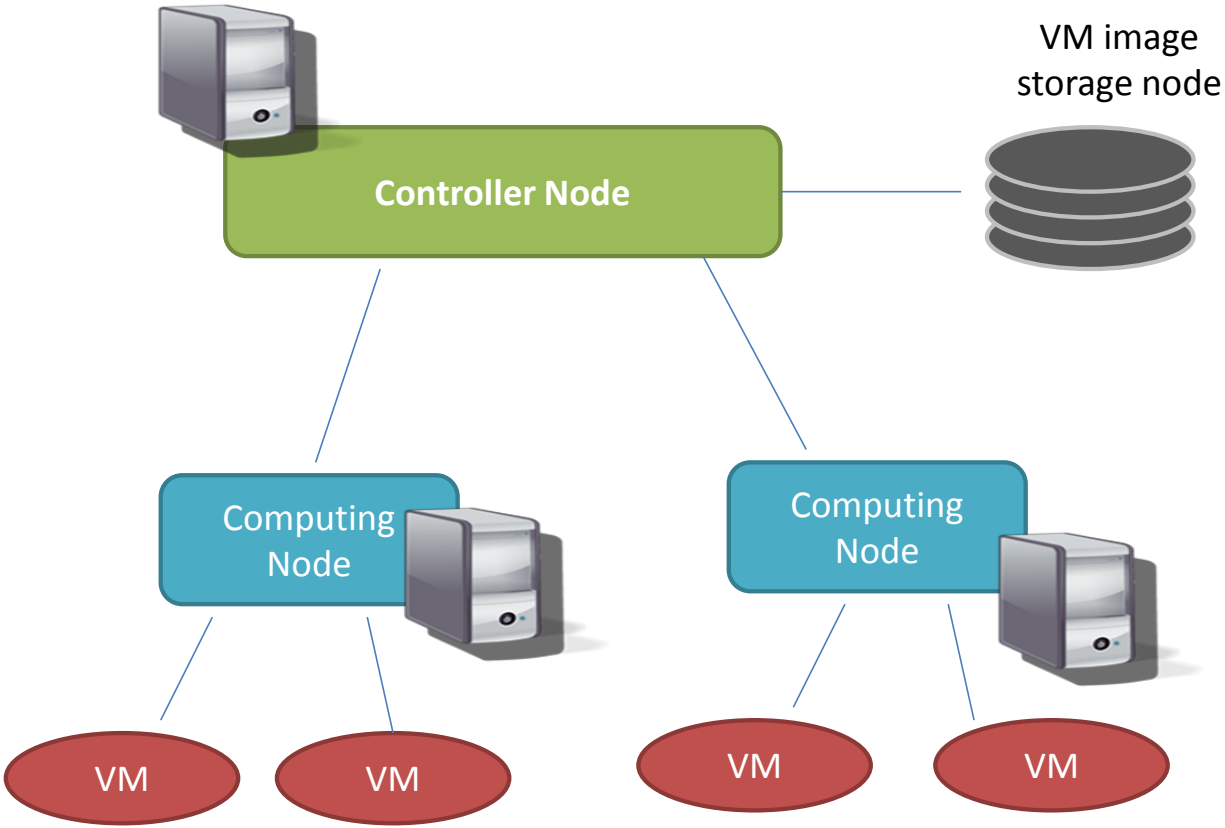
e.g., Intel® VT-x, AMD® V

Cloud infrastructures

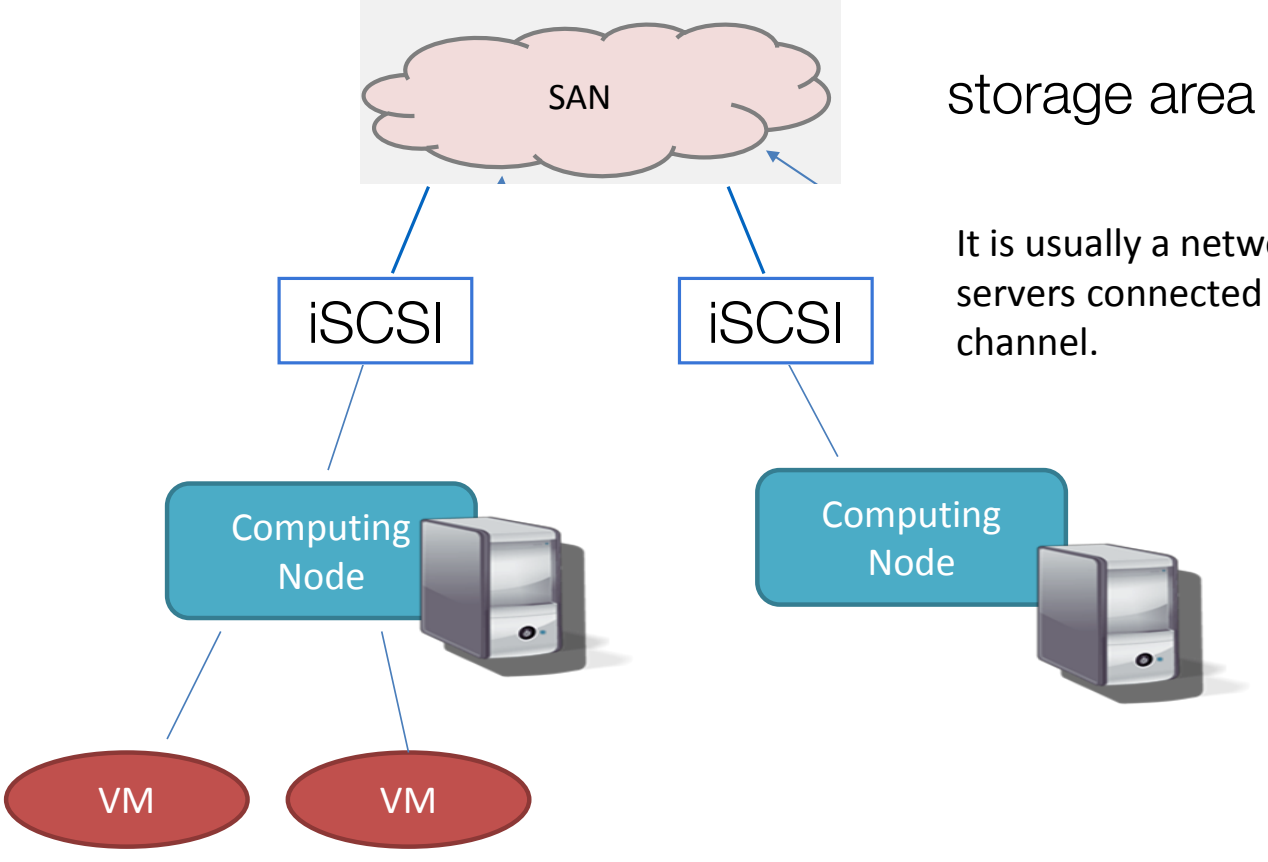
Cloud & Virtualization

- Cloud computing is usually related to virtualization
  - Because of elasticity
  - Launching new VMs in a virtualized environment is cheap and fast
- A cloud infrastructure is in fact a virtual machine management infrastructure

An IaaS Cloud



Subtleties

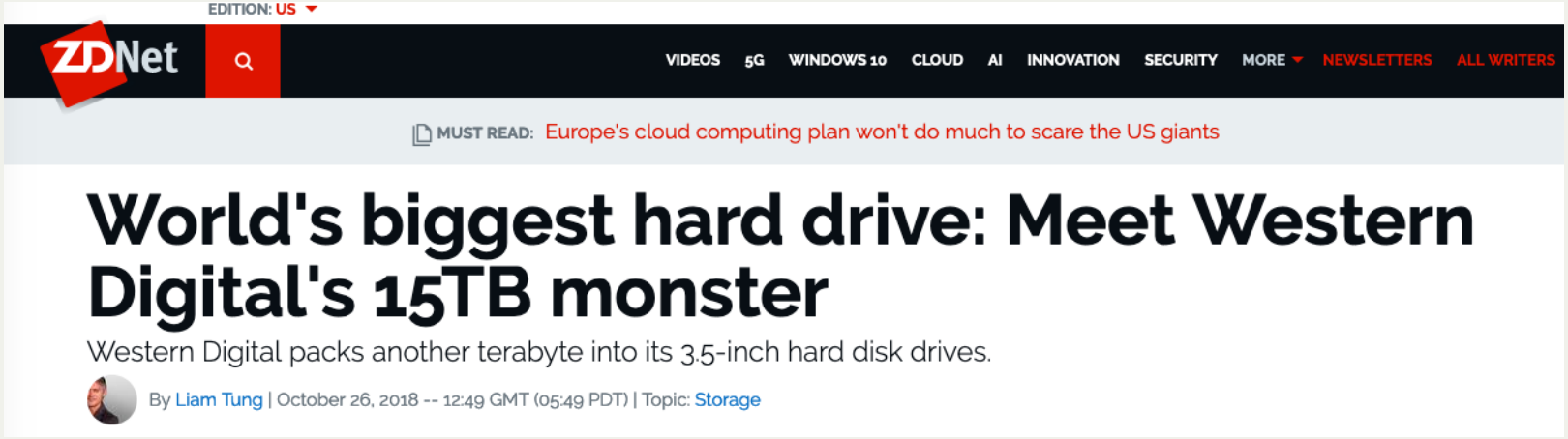


storage area network

It is usually a network of storage servers connected using fabric channel.

RAID (Redundant Array of Inexpensive Disks)

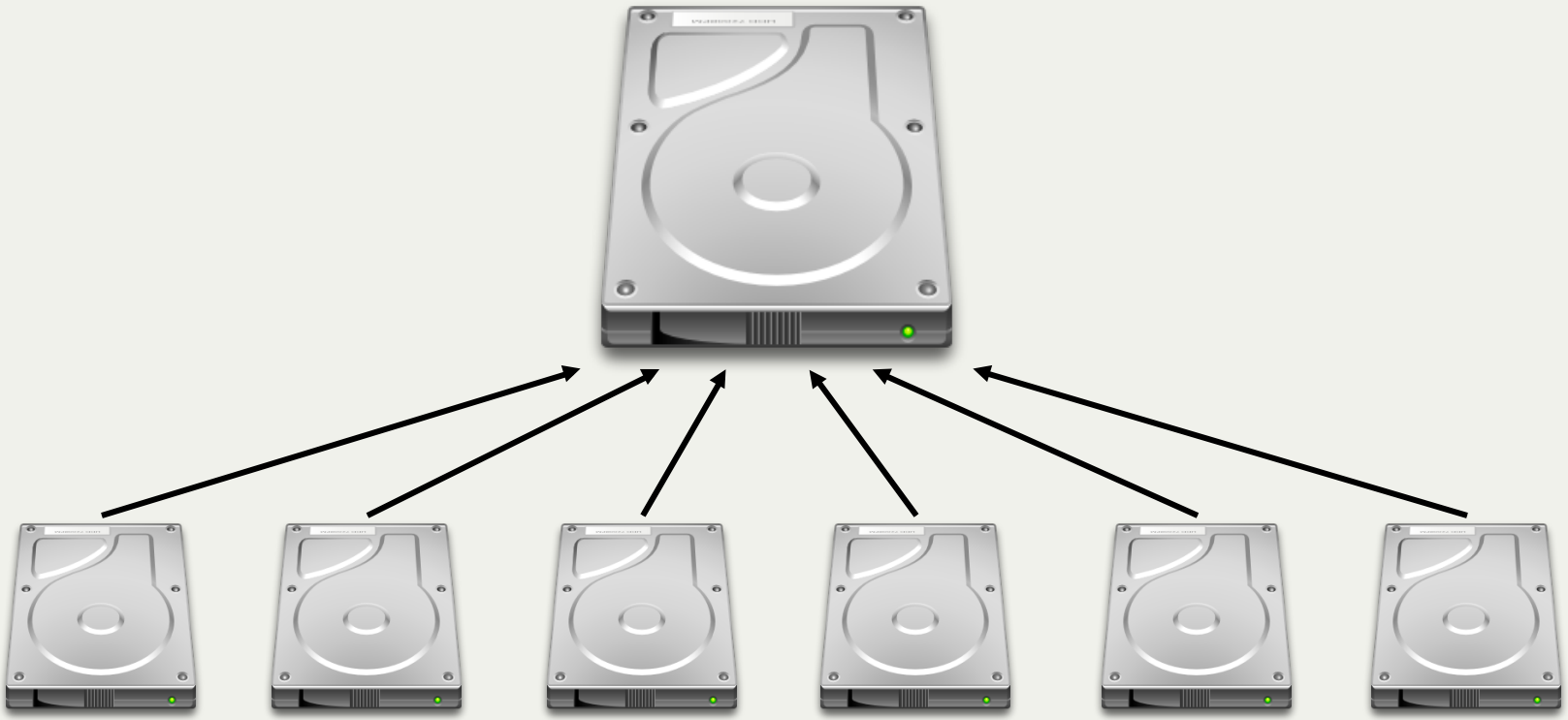
- Disks have limited space: biggest disk today is ~15TB
- What if you need more than 15TB?
  - Could make bigger and bigger disks -- but cost is non-linear
- Use *virtualization*: put multiple physical disks together to look like one bigger virtual disk



Slides from Stanford University, CS110 Principles of Computer Systems

RAID (Redundant Array of Inexpensive Disks)

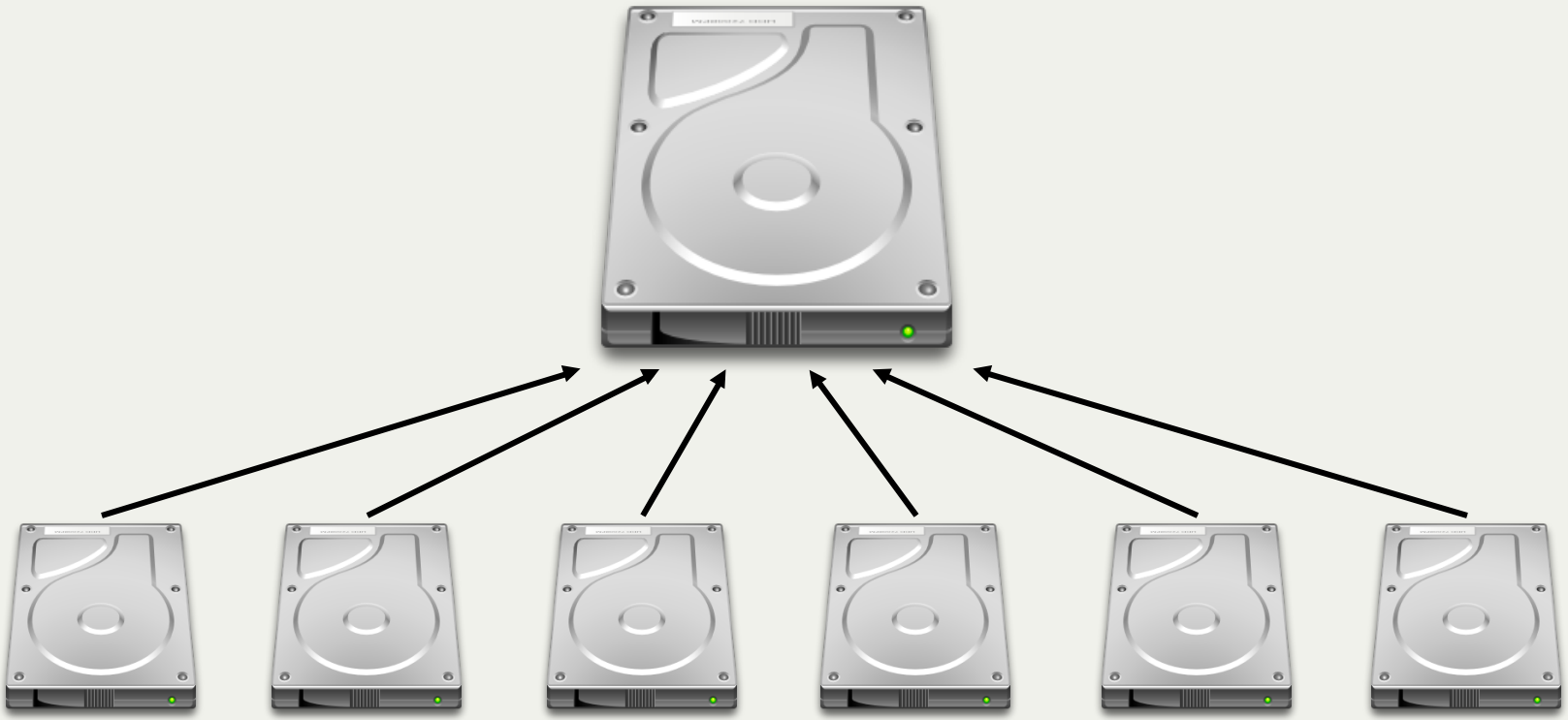
- Disks have limited space: biggest disk today is ~15TB
- What if you need more than 15TB?
  - Could make bigger and bigger disks -- but cost is non-linear
- Use *virtualization*: put multiple physical disks together to look like one bigger virtual disk



Slides from Stanford University, CS110 Principles of Computer Systems

RAID: a lot of advantages

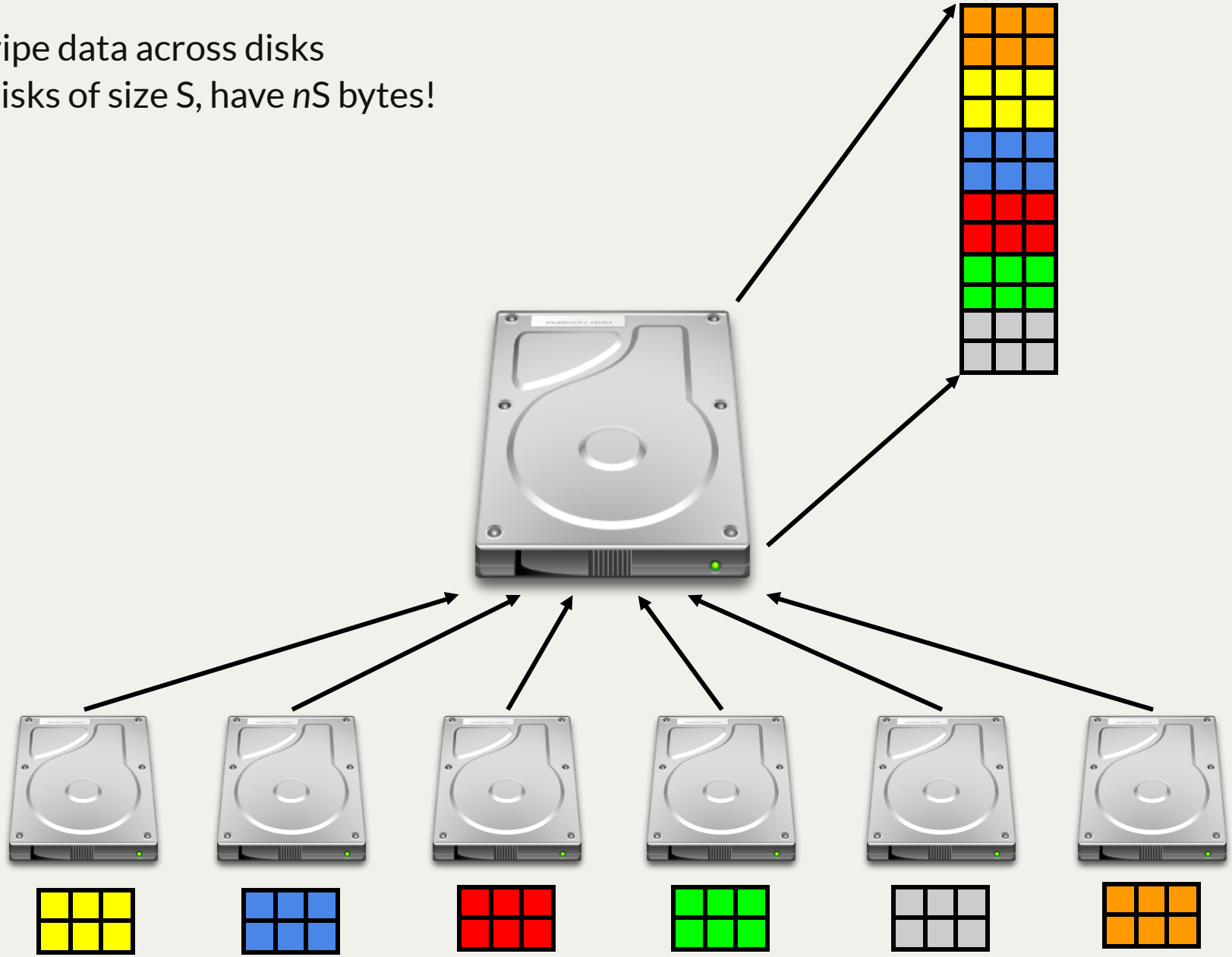
- Size: we can make arbitrarily large disks
- Speed: if we lay out data well, we can read from N disks in parallel, not just one
- Cost: N inexpensive disks is cheaper than one huge disk



Slides from Stanford University, CS110 Principles of Computer Systems

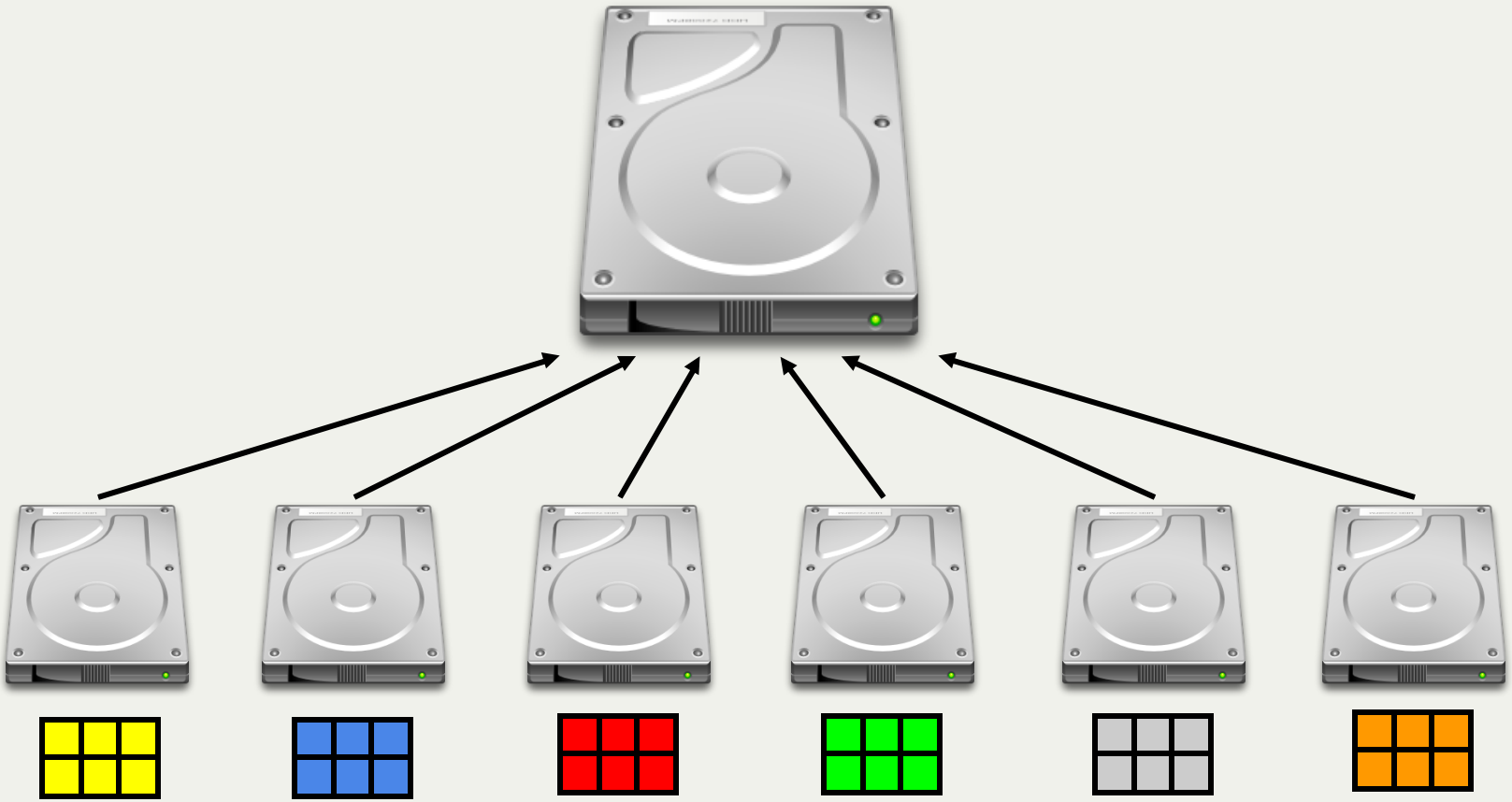
RAID 0

- Stripe data across disks
- *n* disks of size S, have *n*S bytes!



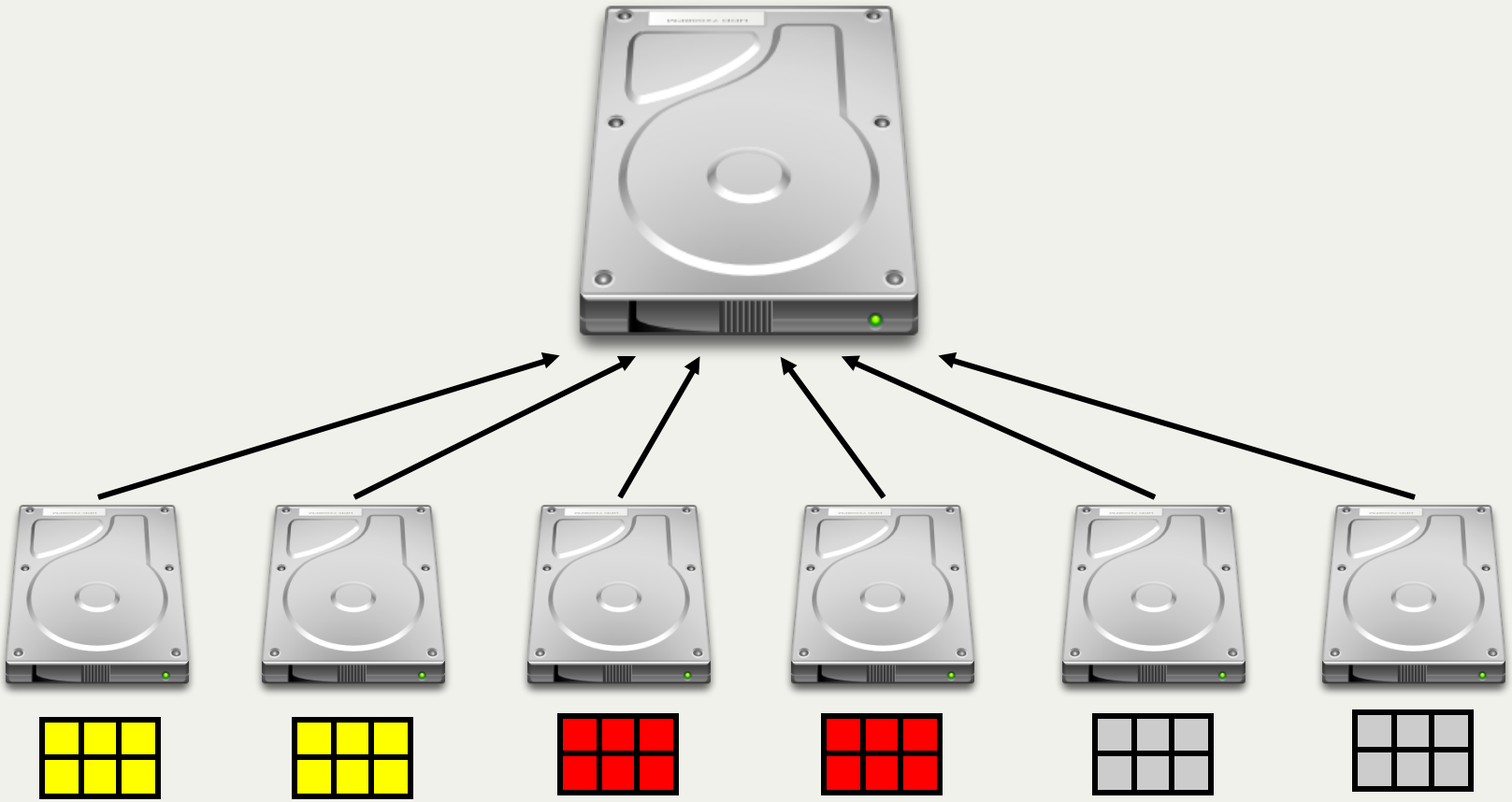
RAID 0 Problems

- If *one* disk fails, the entire RAID array fails
- Suppose each disk has a probability *p* of failing per month
- Probability each disk does not fail is *(1-p)*
- Probability all *n* disks do not fail is $(1-p)^n$
  - Suppose p = 0.001; if n=20, there's a 2% chance the RAID array will fail each month



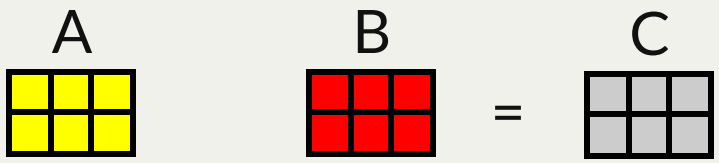
Redundant Array of Inexpensive Disks: RAID 1

- Key idea: arrange the data on the disks so the array can survive failures
- Simplest approach is mirroring, RAID 1
- Halves capacity, but still less expensive than a big disk
- Probability 2 replicas fail is $1-(1-p^2)^{n/2}$
  - If p = 0.001, if n=20, there's a .00001% chance the RAID array will fail each month



The Power of XOR

- There are better ways to have recovery data than simple replication
- Exclusive OR (XOR)
- Suppose we have two drives, A and B
- One extra drive C: C = A B
- If B fails, then you can recover B: B = A C

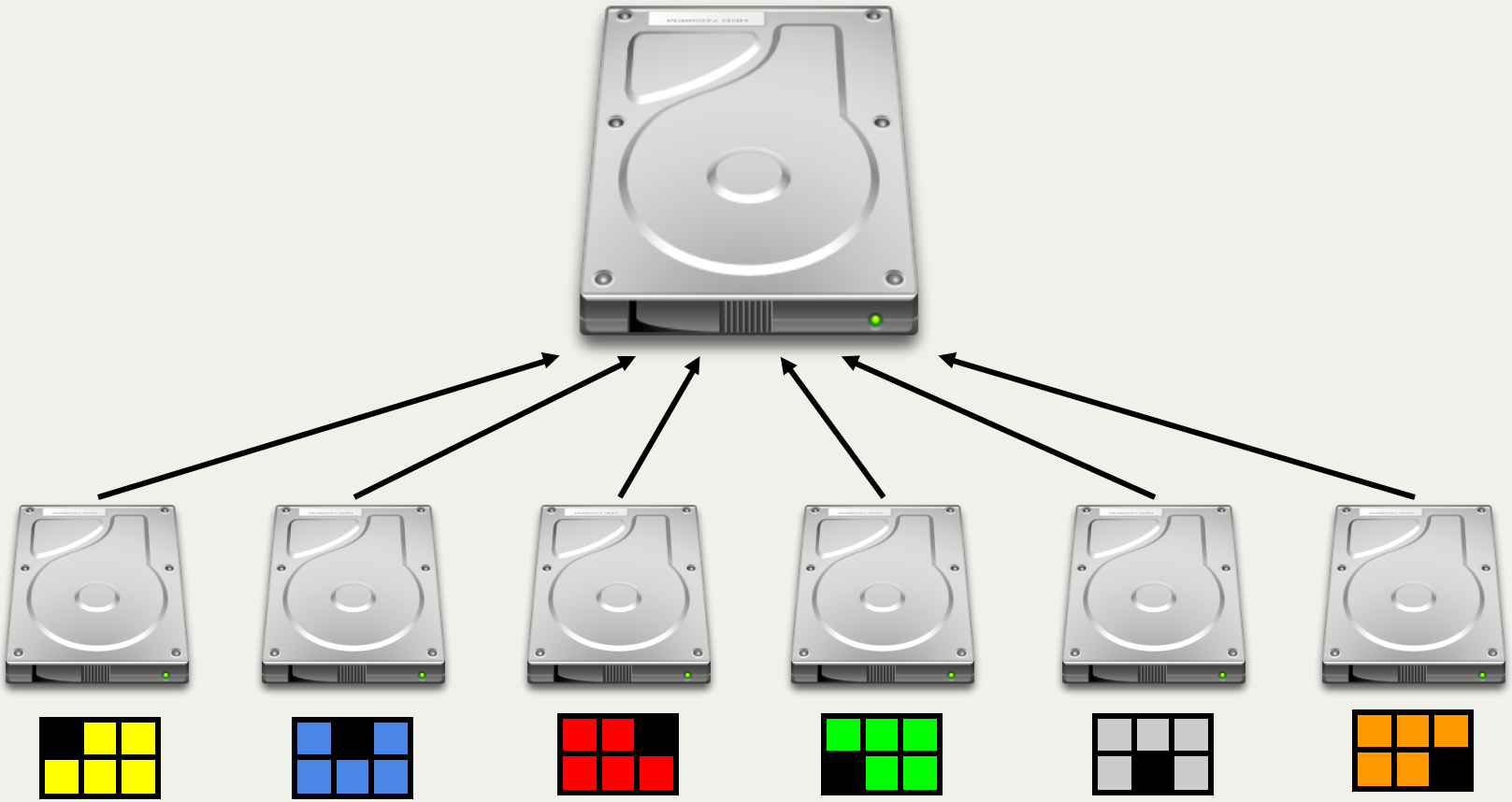


A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Slides from Stanford University, CS110 Principles of Computer Systems

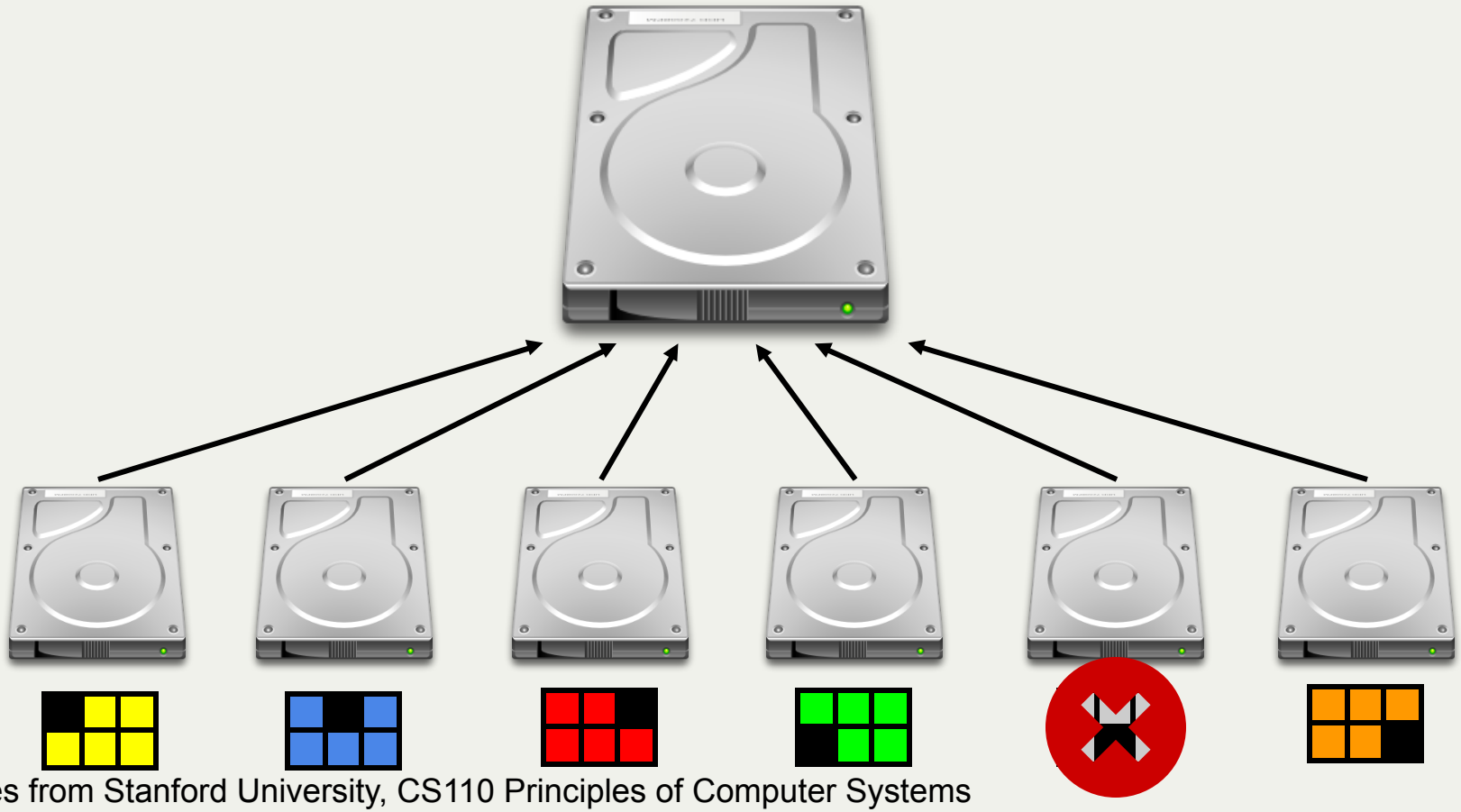
RAID 5: Resiliency With Less Cost

- RAID 5 stripes the data across disks, sets aside 1 disk worth of storage as *parity*
- Parity is the XOR of all of that sector on all of the other drives
  - Writes write two drives: data and parity; parity is spread: lose *1/n* of storage
- Requires *two* drives to fail: n=6, p=0.001, failure ≈ 0.000015
  - If one drive fails, it can be recovered from the parity bits (just XOR other disks)



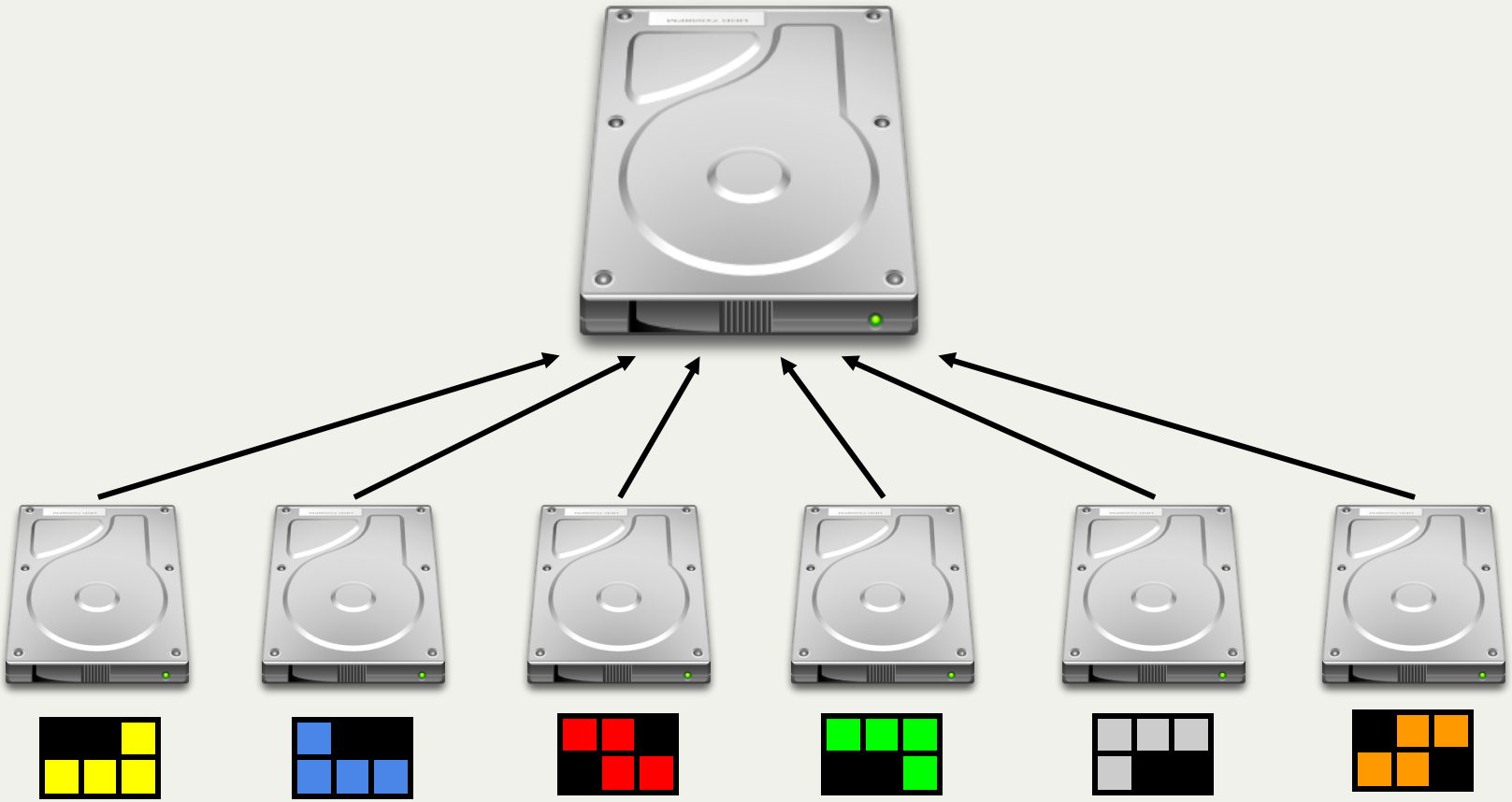
RAID 5: Resiliency With Less Cost

- Suppose we have 6 disks total, one parity disk
- We lose disk 4
- Question 1: Can we still service reads? If so, how does one read from disk 4?
- Question 2: Can we still service writes? If so, how does one write to disk 4?
- Question 3: How do we recover disk 4?



Reed-Solomon Coding

- What if chances more than can fail becomes dangerous (thousands of drives)?
- Reed-Solomon coding: turn *k* data blocks into *n*, can recover from any *(n-k)* failures
  - E.g., turn 223 data blocks into 255, can recover from any 32 failures
  - Used in CDs, DVDs, QR codes, Mars Rovers, and most cloud storage systems
- RAID 6: use Reed-Solomon to have *two* parity drives



RAID invented in 1988 (4 years after first Macintosh)

A Case for Redundant Arrays of Inexpensive Disks (RAID)

David A Patterson, Garth Gibson, and Randy H Katz

Computer Science Division
Department of Electrical Engineering and Computer Sciences
571 Evans Hall
University of California
Berkeley, CA 94720
(pattrsn@ginger berkeley edu)

Abstract *Increasing performance of CPUs and memories will be squandered if not matched by a similar performance increase in I/O While the capacity of Single Large Expensive Disks (SLED) has grown rapidly, the performance improvement of SLED has been modest Redundant Arrays of Inexpensive Disks (RAID), based on the magnetic disk technology developed for personal computers, offers an attractive alternative to SLED, promising improvements of an order of magnitude in performance, reliability, power consumption, and scalability This paper introduces five levels of RAIDs, giving their relative cost/performance, and compares RAID to an IBM 3380 and a Fujitsu Super Eagle*

ure of magnetic disk technology is the growth in the maximum number of bits that can be stored per square inch, or the bits per inch in a track times the number of tracks per inch Called M A D , for maximal areal density, the "First Law in Disk Density" predicts [Frank87]

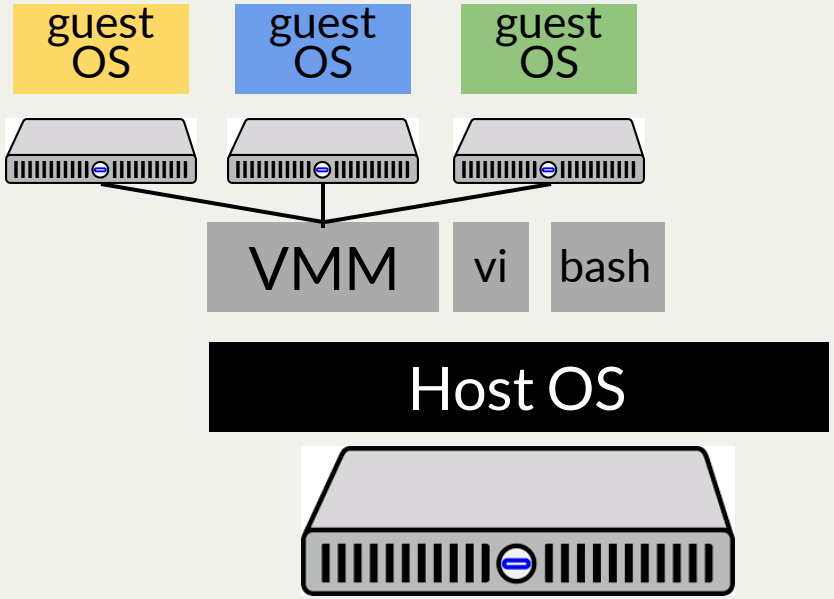
$$MAD = 10^{(Year-1971)/10}$$

Magnetic disk technology has doubled capacity and halved price every three years, in line with the growth rate of semiconductor memory, and in practice between 1967 and 1979 the disk capacity of the average IBM data processing system more than kept up with its main memory [Stevens81]

Described up to RAID 5 (also, RAID 2, RAID 3, RAID 4)

Virtual Machine

- Software that makes code (running in a process) think that it's running on raw hardware
- A *virtual machine monitor* runs in the *host operating system*
  - It loads and run disk images for *guest operating systems*
  - Operations in the guest operating system that are normally not allowed trap into the virtual machine monitor
    - Guest operating system tries to change page tables
    - Guest operating system tries to disable interrupts
  - Virtual machine monitor emulates the hardware

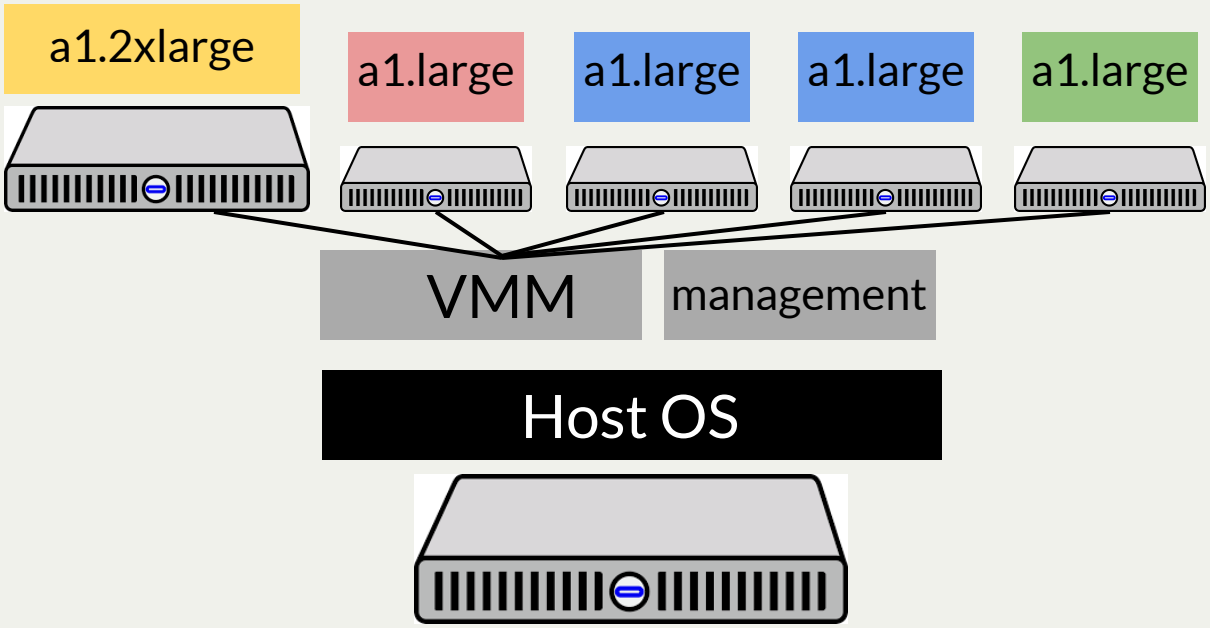


Amazon Elastic Compute Cloud (EC2)

- Amazon computing service: a virtual computer is called an *instance*
- Many different kinds of instance: general purpose, memory-optimized, compute-optimized, GPUs, etc.
- There's generally a full instance size, and you can have $1/2^n$ of it
  - Four a1.large is the same as one a1.2xlarge
  - Two a1.2x large is the same as one a1.4xlarge

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
a1.medium	1	N/A	2 GiB	EBS Only	$0.0255 per Hour
a1.large	2	N/A	4 GiB	EBS Only	$0.051 per Hour
a1.xlarge	4	N/A	8 GiB	EBS Only	$0.102 per Hour
a1.2xlarge	8	N/A	16 GiB	EBS Only	$0.204 per Hour
a1.4xlarge	16	N/A	32 GiB	EBS Only	$0.408 per Hour
a1.metal	16	N/A	32 GiB	EBS Only	$0.408 per Hour

Amazon EC2 Example



	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
a1.medium	1	N/A	2 GiB	EBS Only	$0.0255 per Hour
a1.large	2	N/A	4 GiB	EBS Only	$0.051 per Hour
a1.xlarge	4	N/A	8 GiB	EBS Only	$0.102 per Hour
a1.2xlarge	8	N/A	16 GiB	EBS Only	$0.204 per Hour
a1.4xlarge	16	N/A	32 GiB	EBS Only	$0.408 per Hour
a1.metal	16	N/A	32 GiB	EBS Only	$0.408 per Hour

Virtual Machine Advantages

- Move whole images anywhere: completely decouple all software from hardware
- Can replicate computer images: run more copies
  - If your service is overloaded, scale out by spinning up more instances
- Can arbitrarily start/stop/resume instances very quickly
  - Must faster than shutting down machines
- Complete software encapsulation
  - Common technique used in software tutorials: download this VM image and run it
  - Web hosting: one server can run 100 virtual machines, each one thinks it has a complete, independent computer to configure and use
- Complete software isolation
  - In theory, two VMs are completely isolated, can maybe only sense something due to timing (e.g., if they are sharing a CPU), more on this later
- Enabled us to have cloud computing
  - Original business case was the desktop! E.g., need to run Windows and Linux in parallel, don't want 2 machines.

Modern Virtual Machines Invented in 1997

Using the SimOS Machine Simulator to Study Complex Computer Systems

MENDEL ROSENBLUM, EDOUARD BUGNION, SCOTT DEVINE, and STEPHEN A. HERROD

Computer Systems Laboratory, Stanford University

SimOS is an environment for studying the hardware and software of computer systems. SimOS simulates the hardware of a computer system in enough detail to boot a commercial operating system and run realistic workloads on top of it. This paper identifies two challenges that machine simulators such as SimOS must overcome in order to effectively analyze large complex workloads: handling long workload execution times and collecting data effectively. To study long-running workloads, SimOS includes multiple interchangeable simulation models for each hardware component. By selecting the appropriate combination of simulation models, the user can explicitly control the tradeoff between simulation speed and simulation detail. To handle the large amount of low-level data generated by the hardware simulation models, SimOS contains flexible annotation and event classification mechanisms that map the data back to concepts meaningful to the user. SimOS has been extensively used to study new computer hardware designs, to analyze application performance, and to study operating systems. We include two case studies that demonstrate how a low-level machine simulator such as SimOS can be used to study large and complex workloads.

Latency Numbers Every Programmer Should Know (Peter Norvig and Jeff Dean)

L1 cache reference	0.5ns		
Branch mispredict	5ns		
L2 cache reference	7ns		
Mutex lock/unlock	25ns		
Main memory reference	100ns		
Compress 1K with Zippy	3,000ns	3us	
Send 1K over 1Gbps network	10,000ns	10us	
Read 4K randomly from SSD	150,000ns	150us	
Read 1MB sequentially from RAM	250,000ns	250us	
Round trip within a datacenter	500,000ns	500us	
Read 1MB sequentially from SSD	1,000,000ns	1,000us	1ms
Hard disk seek	10,000,000ns	10,000us	10ms
Read 1MB sequentially from disk	20,000,000ns	20,000us	20ms
Send packet CA->Netherlands->CA	150,000,000ns	150,000us	150ms

Caching

- Performance optimization
- Keeping a copy of some data
  - Usually, closer to where the data is needed
  - Or, something that might be reused (don't recompute)
- Used *everywhere* in computer systems
  - Registers
  - Processor caches
  - File system buffer cache
  - DNS caching
  - memcached
  - Database page cache
  - Spark analytics framework
  - Web browser page/image cache
  - Phone email/SMS cache

Why Is Caching Useful

- There is a basic tradeoff in performance and size
- If you make it bigger, it's slower
  - Takes longer to get to (due to size)
  - Addressing it is more complex (more bits to switch on)
- Faster storage is more expensive
  - 16GB RAM: $59.99
  - 1TB HDD: $59.99
  - 4TB HDD: $116.99
  - 4TB SSD: $499.99
- Think about the places your web page might be stored...