Virtualization and Cloud computing

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

CS 240: *Computing Systems and Concurrency*
Lecture 3

Marco Canini

Credits: Y. Agarwal, S. Seshan, CMU 15-440; A. Haeberlen, Z. Ives, Upenn NETS212; UCLouvain INGI2145

Today's outline

1. **Cloud computing**
2. Virtualization
3. What's really cloud computing?

Cloud computing

- Computing at scale
  - The need for scalability; scale of current services
  - Scaling up: From PCs to data centers
  - Problems with 'classical' scaling techniques

- Utility computing and cloud computing
  - What are utility computing and cloud computing?
  - What kinds of clouds exist today?
  - What kinds of applications run on the cloud?

How many users and objects?

- Flickr has > 10 billion photos
- Facebook has 2 billion monthly users
- Google is serving > 3.5 billion queries/day on more than 130 trillion pages
- > 5 billion videos/day watched on YouTube
- 300 hours of video are uploaded to YouTube every minute!

How much data?

- Modern applications use massive data:
  - Rendering 'Avatar' movie required >1 petabyte of storage
  - CERN's LHC will produce about 15 petabytes of data per year
  - In 2008, Google processed 20 petabytes per day
  - Dropbox has > 500 petabytes of user data
  - Google now designing for 1 exabyte of storage
  - NSA Utah Data Center is said to have 5 zettabyte (!)
- How much is a zettabyte?
  - 1,000,000,000,000,000,000,000 bytes (10^{21})
  - A stack of 1TB hard disks that is **25,400 km high**

How much computation?

- No single computer can process that much data
  - Need many computers!
- How many computers do modern services need?
  - Facebook is thought to have more than 60,000 servers
  - 1&1 Internet has over 70,000 servers
  - Akamai has > 95,000 servers in 71 countries
  - Intel has ~100,000 servers in 97 data centers
  - Microsoft has > 1 million servers in 2008
  - Google is thought to have more than 1 million servers, is planning for 10 million (according to Jeff Dean)

Scaling up

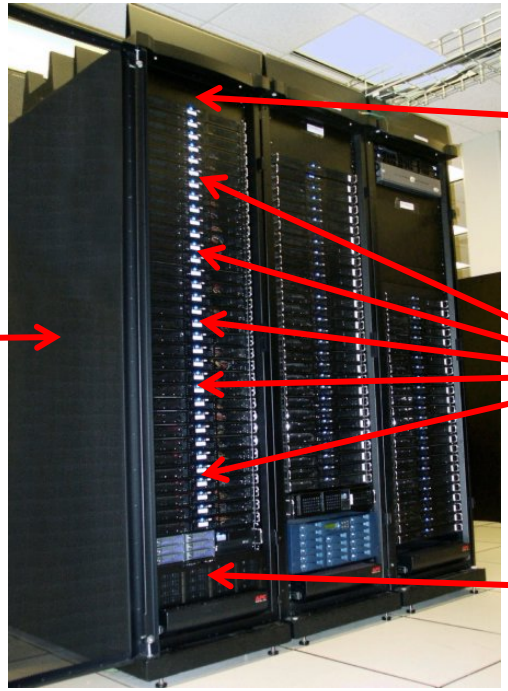
PC

Server

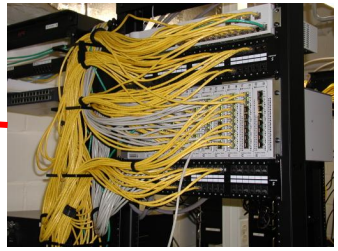
Cluster

- What if one computer is not enough?
  - Buy a bigger (server-class) computer
- What if the biggest computer is not enough?
  - Buy many computers

Clusters



Rack



Network switch (connects nodes with each other and with other racks)

Many nodes/blades (often identical)

Storage device(s)

- Characteristics of a cluster:
  - Many similar machines, close interconnection (same room?)
  - Often special, standardized hardware (racks, blades)
  - Usually owned and used by a single organization

Power and cooling

- Clusters need lots of power
  - Example: 140 Watts per server
  - Rack with 32 servers: 4.5kW (use special power supply!)
  - Most of this power is converted into heat

- Large clusters need massive cooling
  - 4.5kW is about 3 space heaters
  - And that's just one rack!

Scaling up

PC

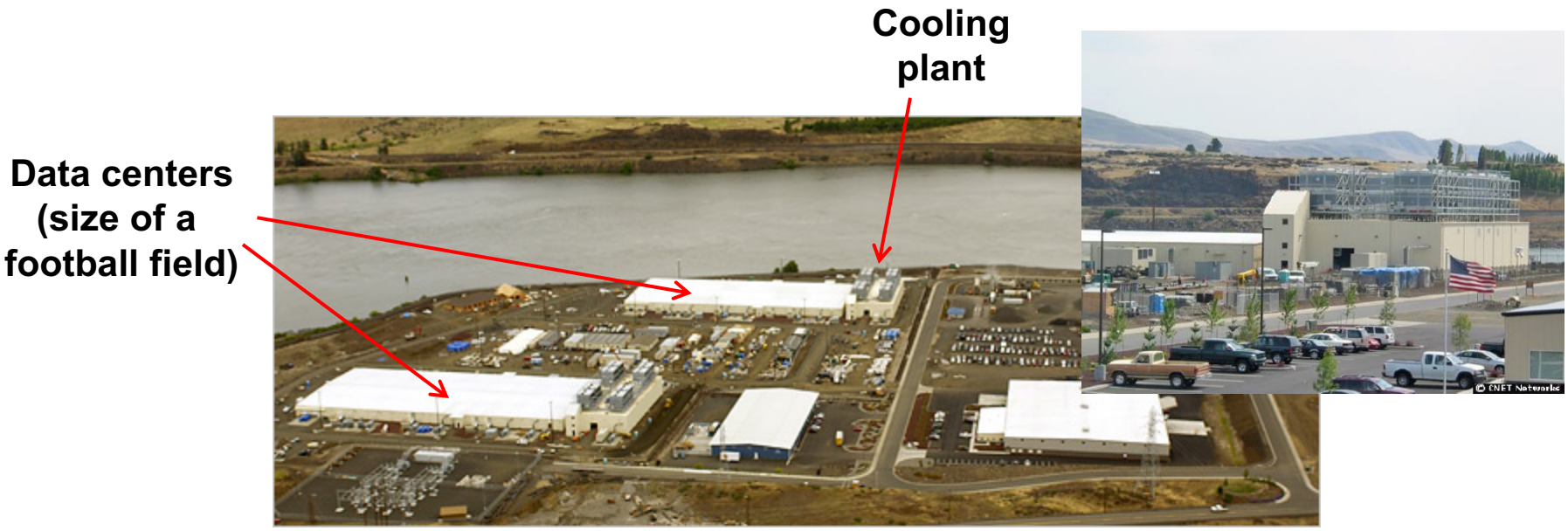
Server

Cluster

Data center

- What if your cluster is too big (hot, power hungry) to fit into your office building?
  - Build a separate building for the cluster
  - Building can have lots of cooling and power
  - Result: Data center

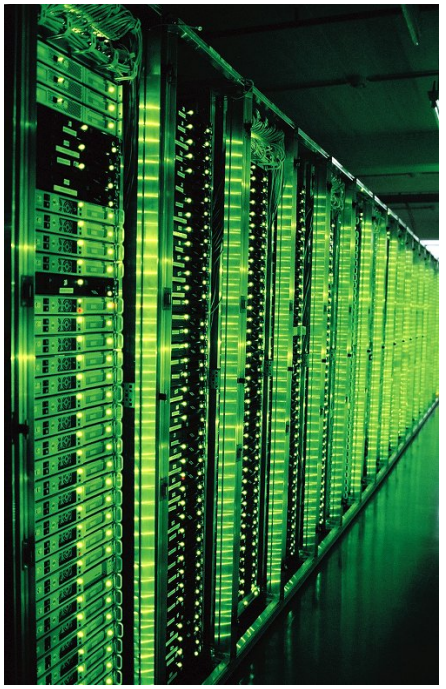
What does a data center look like?



Google data center in The Dalles, Oregon

- A warehouse-sized computer
  - A single data center can easily contain 10,000 racks with 100 cores in each rack (1,000,000 cores total)

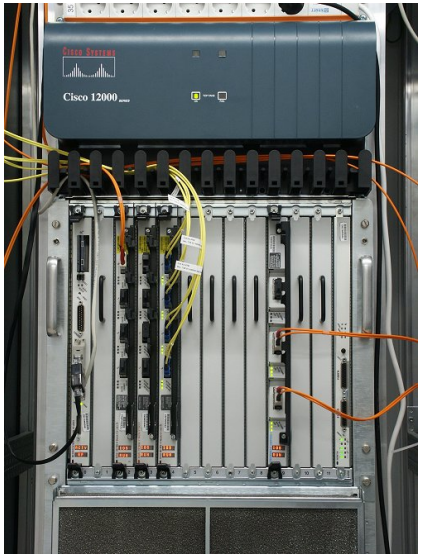
What's in a data center?



Source: 1&1

- Hundreds or thousands of racks

What's in a data center?



Source: 1&1

- Massive networking

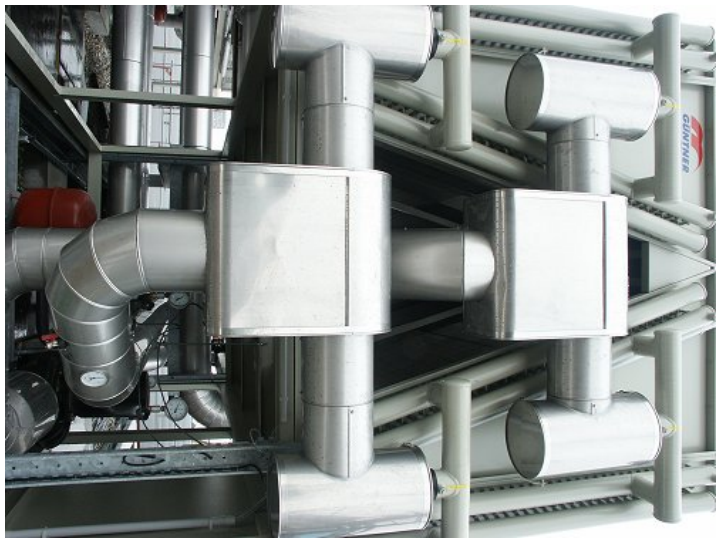
What's in a data center?



Source: 1&1

- Emergency power supplies

What's in a data center?



Source: 1&1

- Massive cooling

Energy matters!

Company	Servers	Electricity	Cost
eBay	16K	~$0.6*10^5$ MWh	~$3.7M/yr
Akamai	40K	~$1.7*10^5$ MWh	~$10M/yr
Rackspace	50K	~$2*10^5$ MWh	~$12M/yr
Microsoft	>200K	>$6*10^5$ MWh	>$36M/yr
Google	>500K	>$6.3*10^5$ MWh	>$38M/yr
USA (2006)	10.9M	$610*10^5$ MWh	$4.5B/yr

Source: Qureshi et al., SIGCOMM 2009

- Data centers consume a lot of energy
  - Makes sense to build them near sources of cheap electricity
  - Example: Price per KWh is 3.6ct in Idaho (near hydroelectric power), 10ct in California (long distance transmission), 18ct in Hawaii (must ship fuel)
  - Most of this is converted into heat → Cooling is a big issue!

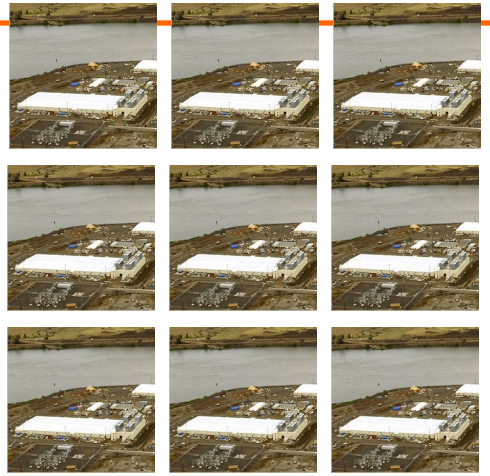
Scaling up

PC

Server

Cluster

Data center



Network of data centers

- What if even a data center is not big enough?
  - Build additional data centers
  - Where? How many?

Global distribution

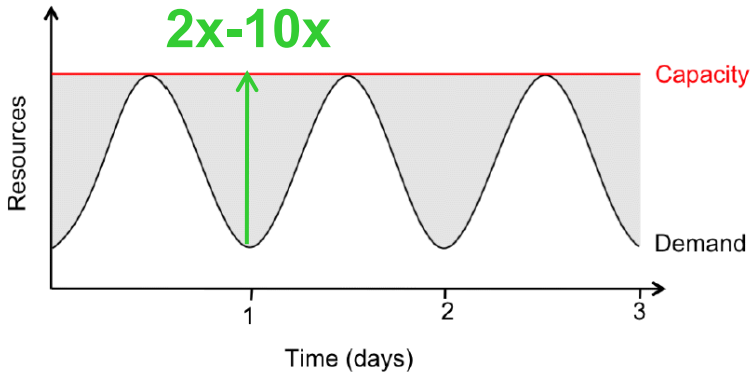
- Data centers are often globally distributed
  - Example above: Google data center locations (inferred)
- Why?
  - Need to be close to users (physics!)
  - Cheaper resources
  - Protection against failures

Cloud computing

- Computing at scale
  - The need for scalability; scale of current services
  - Scaling up: From PCs to data centers
  - Problems with 'classical' scaling techniques

- Utility computing and cloud computing
  - What are utility computing and cloud computing?
  - What kinds of clouds exist today?
  - What kinds of applications run on the cloud?

Problem #1: Difficult to dimension



Provisioning for the peak load

Provisioning below the peak

- Problem: Load can vary considerably
  - Peak load can exceed average load by factor 2x-10x [Why?]
  - But: Few users deliberately provision for less than the peak
  - Result: Server utilization in existing data centers ~5%-20%!!
  - Dilemma: Waste resources or lose customers!

Problem #2: Expensive

- Need to invest many $$$ in hardware
  - Even a small cluster can easily cost $100,000
  - Microsoft recently invested $499 million in a single data center
- Need expertise
  - Planning and setting up a large cluster is highly nontrivial
  - Cluster may require special software, etc.
- Need maintenance
  - Someone needs to replace faulty hardware, install software upgrades, maintain user accounts, ...

Problem #3: Difficult to scale

- Scaling up is difficult
  - Need to order new machines, install them, integrate with existing cluster - can take months!
  - Large scaling factors may require major redesign, e.g., new storage system, new interconnect, new building (!)

- Scaling down is difficult
  - What to do with superfluous hardware?
  - Server idle power is about 60% of peak $\rightarrow$ Energy is consumed even when no work is being done
  - Many fixed costs, such as construction

Summary: Computing at scale

- Modern applications require **huge amounts of processing and data**
  - Measured in petabytes, millions of users, billions of objects
  - Need special hardware, algorithms, tools to work at this scale
- Clusters and data centers can provide the resources we need
  - Main difference: Scale (room-sized vs. building-sized)
  - Special hardware; power and cooling are big concerns
- Clusters and data centers are not perfect
  - Difficult to dimension; expensive; difficult to scale

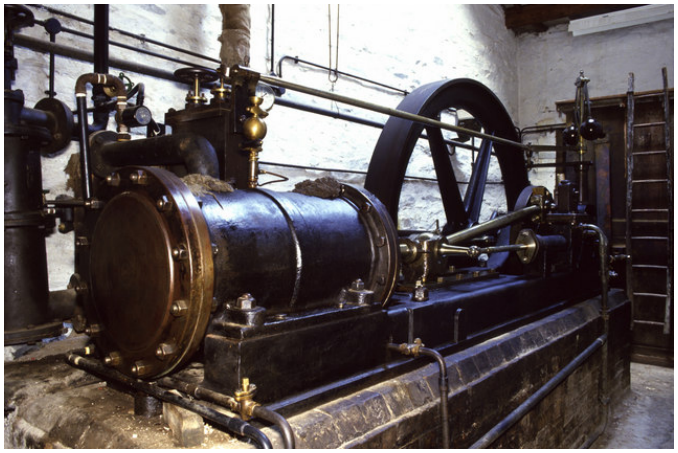
Cloud computing

- Computing at scale
  - The need for scalability; scale of current services
  - Scaling up: From PCs to data centers
  - Problems with 'classical' scaling techniques

- Utility computing and cloud computing
  - What are utility computing and cloud computing?
  - What kinds of clouds exist today?
  - What kinds of applications run on the cloud?

The power plant analogy

Waterwheel at the Neuhausen ob Eck Open-Air Museum



Steam engine at Stott Park Bobbin Mill

- It used to be that everyone had their own power source
  - Challenges are similar to the cluster: Needs large up-front investment, expertise to operate, difficult to scale up/down...

Scaling the power plant




- Then people started to build large, centralized power plants with very large capacity...

Metered usage model

- Power plants are connected to customers by a network
- Usage is metered, and everyone (basically) pays only for what they actually use

Why is this a good thing?

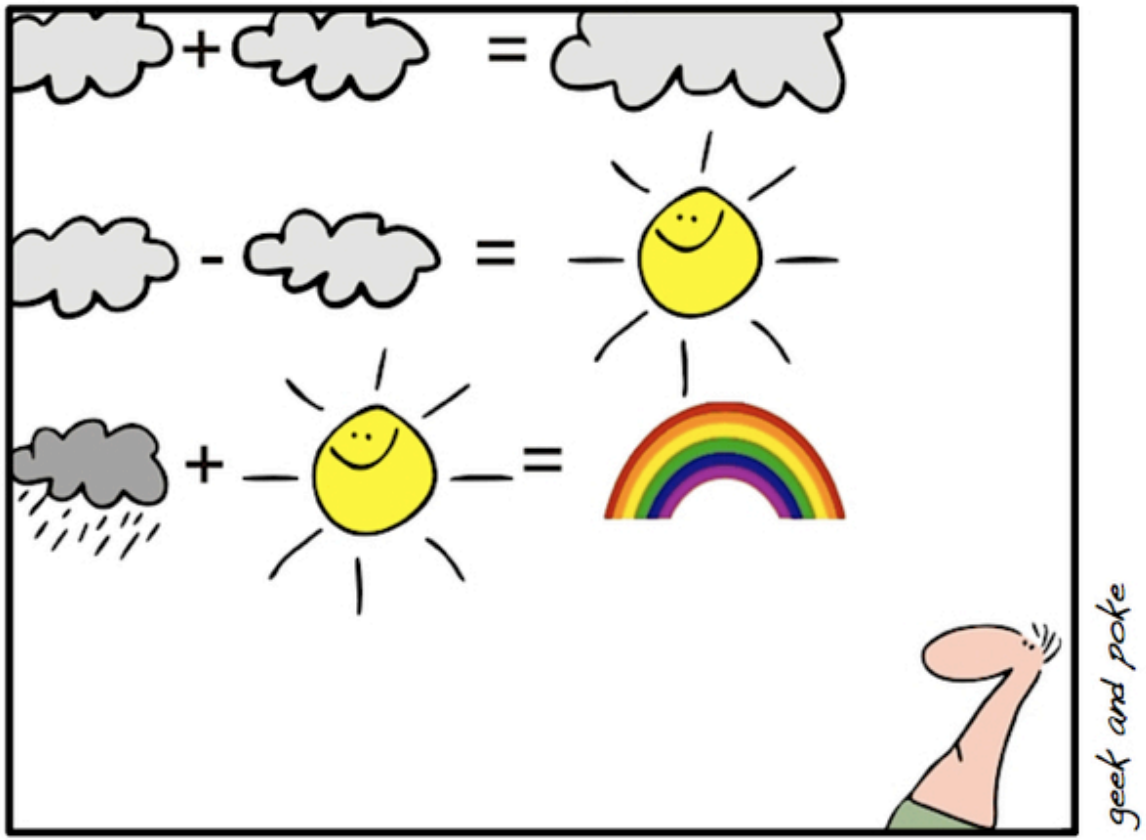
Electricity

- Economies of scale
  - Cheaper to run one big power plant than many small ones
- Statistical multiplexing
  - High utilization!
- No up-front commitment
  - No investment in generator; pay-as-you-go model
- Scalability
  - Thousands of kilowatts available on demand; add more within seconds

Computing

- Cheaper to run one big data center than many small ones
- High utilization!
- No investment in data center; pay-as-you-go model
- Thousands of computers available on demand; add more within seconds

What is cloud computing?



SIMPLY EXPLAINED – PART 17:
CLOUD COMPUTING

http://geekandpoke.typepad.com/geekandpoke/images/2008/05/16/se17.jpg

What is cloud computing?

The interesting thing about Cloud Computing is that we've redefined Cloud Computing to include everything that we already do.... I don't understand what we would do differently in the light of Cloud Computing other than change the wording of some of our ads.

Larry Ellison, quoted in the Wall Street Journal, September 26, 2008

A lot of people are jumping on the [cloud] bandwagon, but I have not heard two people say the same thing about it. There are multiple definitions out there of "the cloud".

Andy Isherwood, quoted in ZDnet News, December 11, 2008

So what is it, really?

- According to NIST:

  Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction

- Essential characteristics:
  - On-demand self service
  - Broad network access
  - Resource pooling
  - Rapid elasticity
  - Measured service

Other terms you may have heard

- Utility computing
  - The service being sold by a cloud
  - Focuses on the business model (pay-as-you-go), similar to classical utility companies

- The Web
  - The Internet's information sharing model
  - Some web services run on clouds, but not all

- The Internet
  - A network of networks
  - Used by the web; connects (most) clouds to their customers

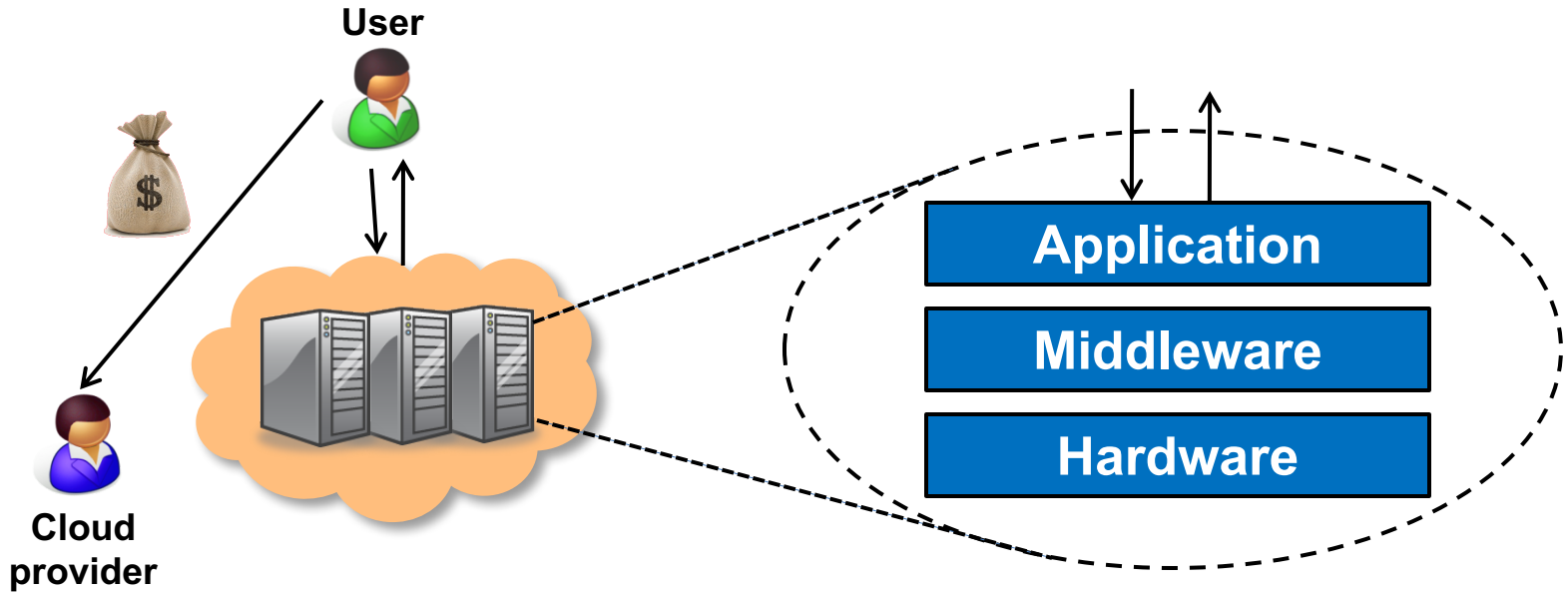
Everything as a Service

- What kind of service does the cloud provide?
  - Does it offer an entire application, or just resources?
  - If resources, what kind / level of abstraction?

Three types commonly distinguished:

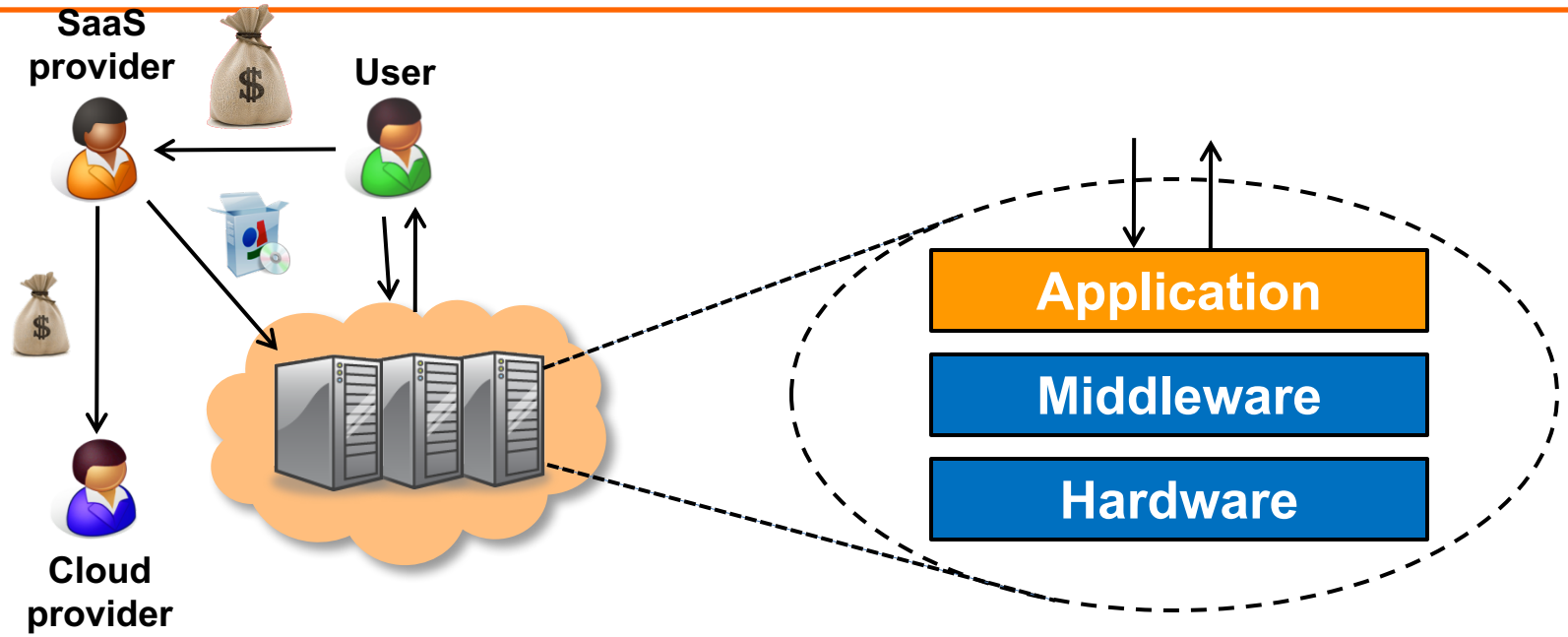
- **Software as a service (SaaS)**
  - Analogy: Restaurant. Prepares & serves entire meal, does the dishes, ...
- **Platform as a service (PaaS)**
  - Analogy: Take-out food. Prepares meal, but does not serve it
- **Infrastructure as a service (IaaS)**
  - Analogy: Grocery store. Provides raw ingredients
- Other *aaS types have been defined, but are less common
  - Desktop, DB, Communication, Network, Monitoring, …

Software as a Service (SaaS)



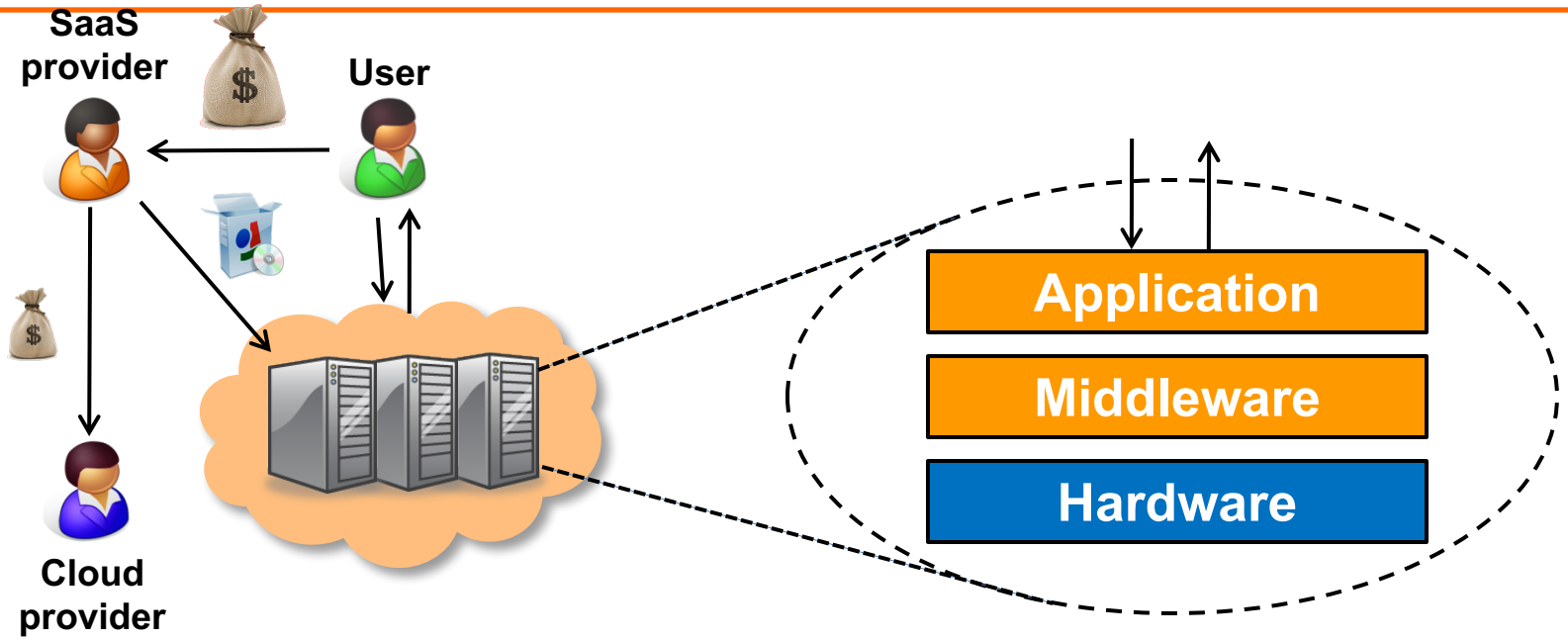
- Cloud provides an entire application
  - Word processor, spreadsheet, CRM software, calendar...
  - Customer pays cloud provider
  - Example: Google Apps, Salesforce.com, Concur, Doodle

Platform as a Service (PaaS)



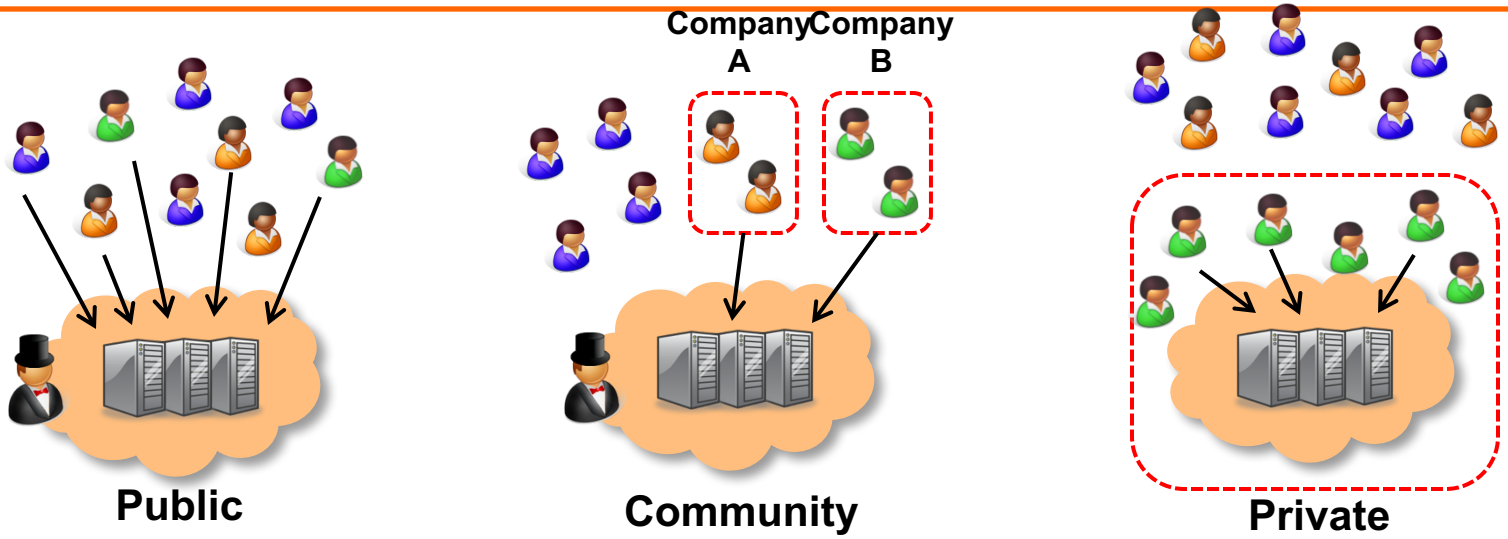
- Cloud provides middleware/infrastructure
  - For example, Microsoft Common Language Runtime (CLR)
  - Customer pays SaaS provider for the service; SaaS provider pays the cloud for the infrastructure
  - Example: Windows Azure, Google App Engine, Heroku

Infrastructure as a Service (IaaS)



- Cloud provides raw computing resources
  - Virtual machine, blade server, storage, network, ...
  - Customer pays SaaS provider for the service; SaaS provider pays the cloud for the resources
  - Examples: Amazon Web Services, DigitalOcean, Joyent

Private/hybrid/community clouds



- Who can become a customer of the cloud?
  - Public cloud: Commercial service; open to (almost) anyone
    Example: Amazon AWS, Microsoft Azure, Google App Engine
  - Community cloud: Shared by several similar organizations.
    Example: Google's "Gov Cloud"
  - Private cloud: Shared within a single organization.
    Example: Internal datacenter of a large company.

Summary: Utility/cloud computing

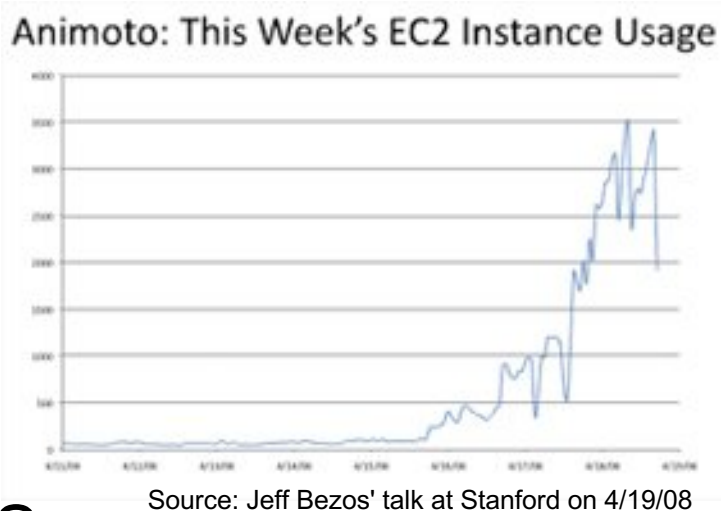
- Why is cloud computing attractive?
  - Analogy to 'classical' utilities (electricity, water, ...)
  - No up-front investment (pay-as-you-go model)
  - Low price due to economies of scale
  - Elasticity: can quickly scale up/down as demand varies
- Different types of clouds
  - SaaS, PaaS, IaaS; public/private/community clouds
- What runs on the cloud?
  - Many potential applications: Application hosting, backup/storage, scientific computing, content delivery, ...

Examples of cloud applications

- Application hosting
- Backup and Storage
- Content delivery
- E-commerce
- High-performance computing
- Media hosting
- On-demand workforce
- Search engines
- Web hosting

If interested, glance through
http://aws.amazon.com/solutions/case-studies/
for a list of many use cases

Case study: animoto



Source: Jeff Bezos' talk at Stanford on 4/19/08

- Animoto: Lets users create videos from their own photos/music
  - Auto-edits photos and aligns them with the music, so it "looks good"
- Built using Amazon EC2+S3+SQS
- Released a Facebook app in mid-April 2008
  - More than 750,000 people signed up within 3 days
  - EC2 usage went from 50 machines to 3,500 (x70 scalability!)

http://aws.amazon.com/solutions/case-studies/animoto/

http://animoto.com/blog/company/amazon-com-ceo-jeff-bezos-on-animoto/

Case study:

- Novartis Institutes for Biomedical Research is focused on the drug discovery phase of the ~10 year / $1 billion drug development process
- In 2013, NIBR ran a project to screen 10 M compounds against a common cancer target
- Compute requirements >> internal capacity / $
- The project ran across 10,500 EC2 Spot instances (~87,000 cores) for $4,232 in 9 hours
- Equiv. of 39 years of computational chemistry

http://aws.amazon.com/solutions/case-studies/novartis/

Is the cloud good for everything?

- No.
- Sometimes it is problematic
  - Auditability requirements
  - Legislative frameworks
- Example: Personal data privacy
  - EU Data Protection law
- Example: Processing medical records (US)
  - HIPAA (Health Insurance Portability and Accountability Act) privacy and security rule

Summary: Cloud applications

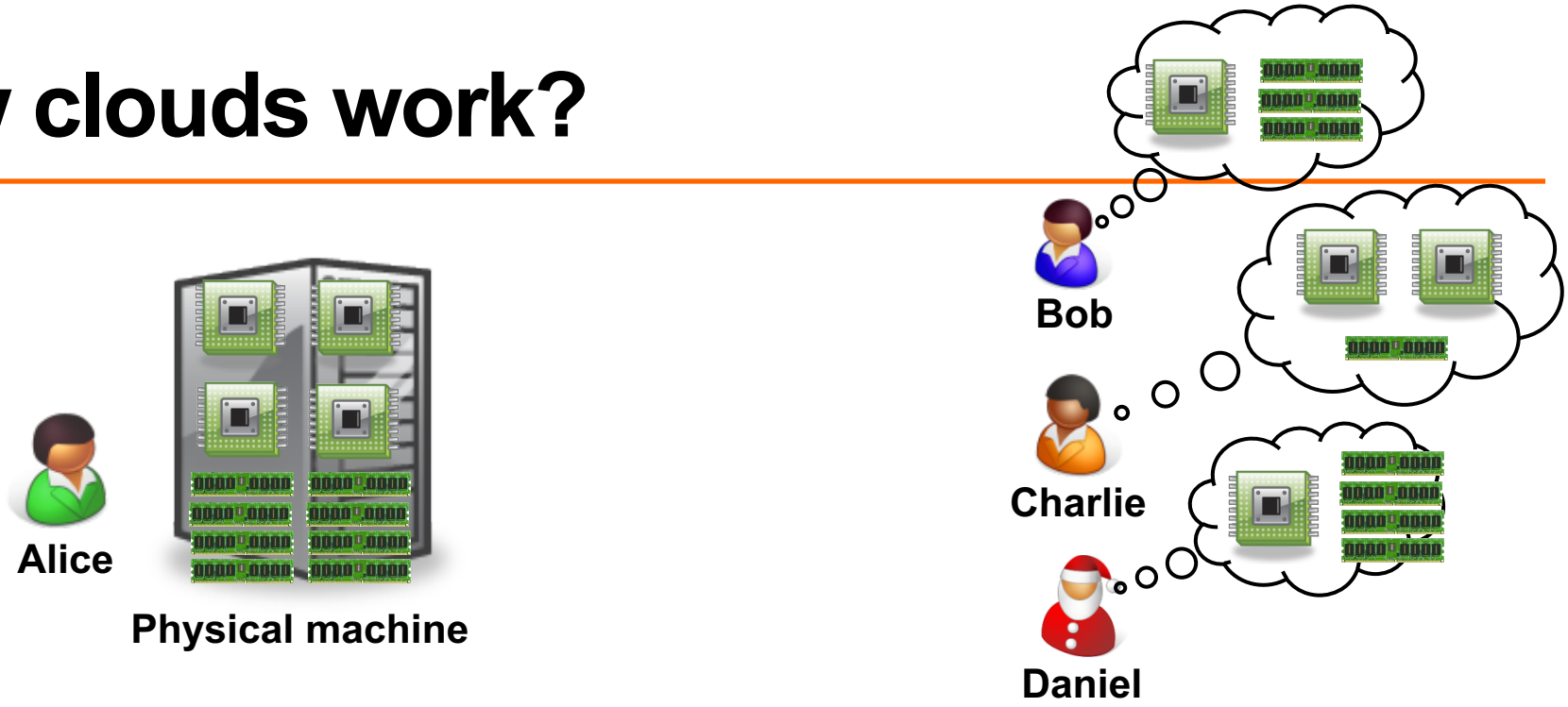
- Clouds are good for many things...
  - Applications that involve large amounts of computation, storage, bandwidth
  - Especially when lots of resources are needed quickly (Novartis example) or load varies rapidly (Animoto example)

- ... but not for all things
  - and might not be the cheapest solution either

Today's outline

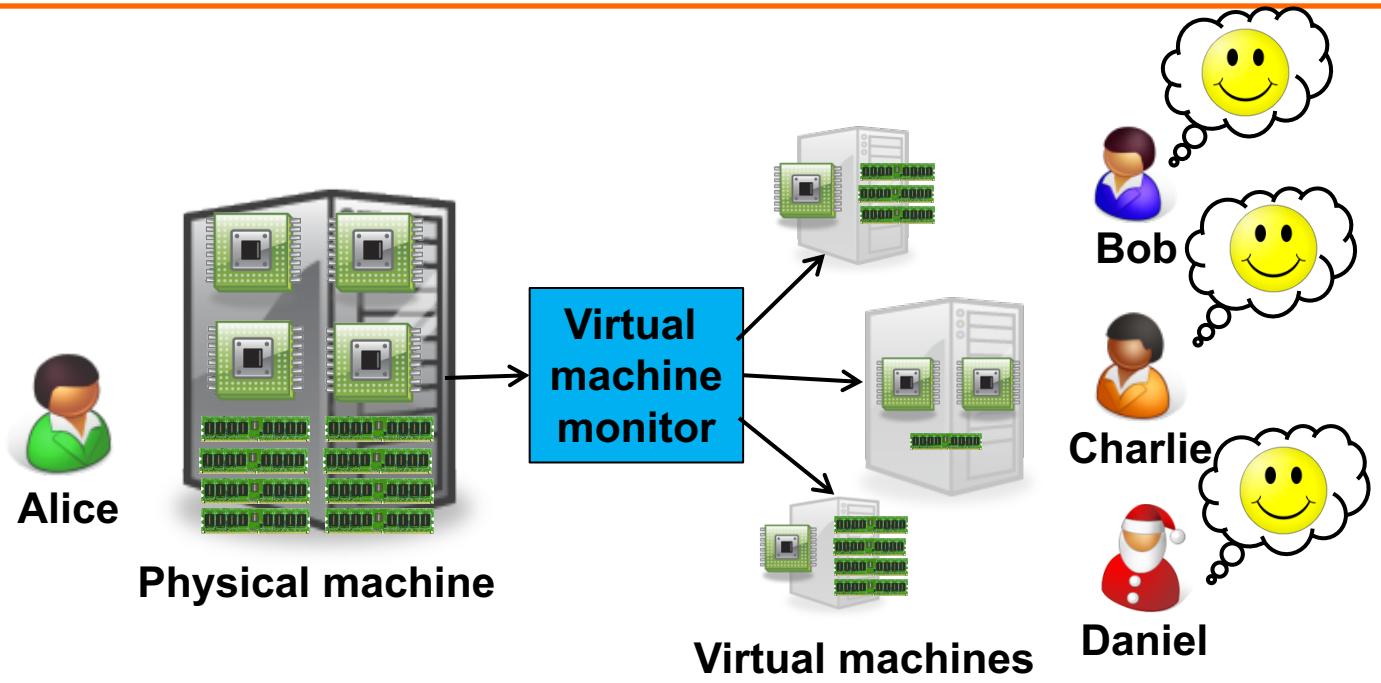
1. Cloud computing
2. **Virtualization**
3. What's really cloud computing?

How clouds work?



- Suppose Alice has a machine with 4 CPUs and 8 GB of memory, and three customers:
  - Bob wants a machine with 1 CPU and 3GB of memory
  - Charlie wants 2 CPUs and 1GB of memory
  - Daniel wants 1 CPU and 4GB of memory
- What should Alice do?

Virtualization is key enabler



- Alice can sell each customer a virtual machine (VM) with the requested resources
  - From each customer's perspective, it appears as if they had a physical machine all by themselves (isolation)

Virtualization

“a technique for hiding the physical characteristics of computing resources from the way in which other systems, applications, or end users interact with those resources. This includes making a single physical resource appear to function as multiple logical resources; or it can include making multiple physical resources appear as a single logical resource”

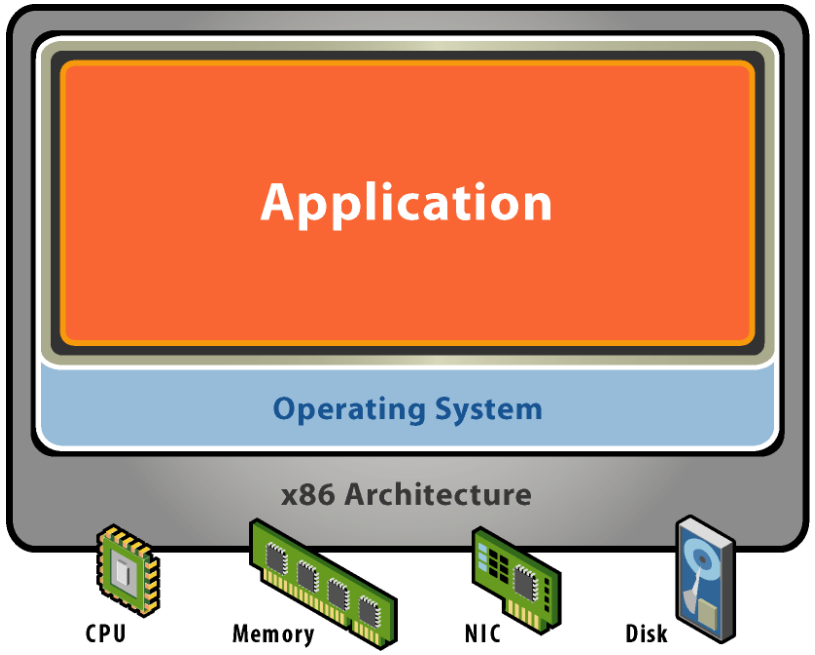
The idea of Virtualization: from 1960's

- IBM VM/370 – A VMM for IBM mainframe
  - Multiple OS environments on expensive hardware
  - Desirable when few machine around
- Popular research idea in 1960s and 1970s
  - Entire conferences on virtual machine monitors
  - Hardware/VMM/OS designed together
  - Allowed multiple users to share a batch oriented system
- Interest died out in the 1980s and 1990s
  - Hardware got more cheaper
  - Operating systems got more powerful (e.g. multi-user)

A Return to Virtual Machines

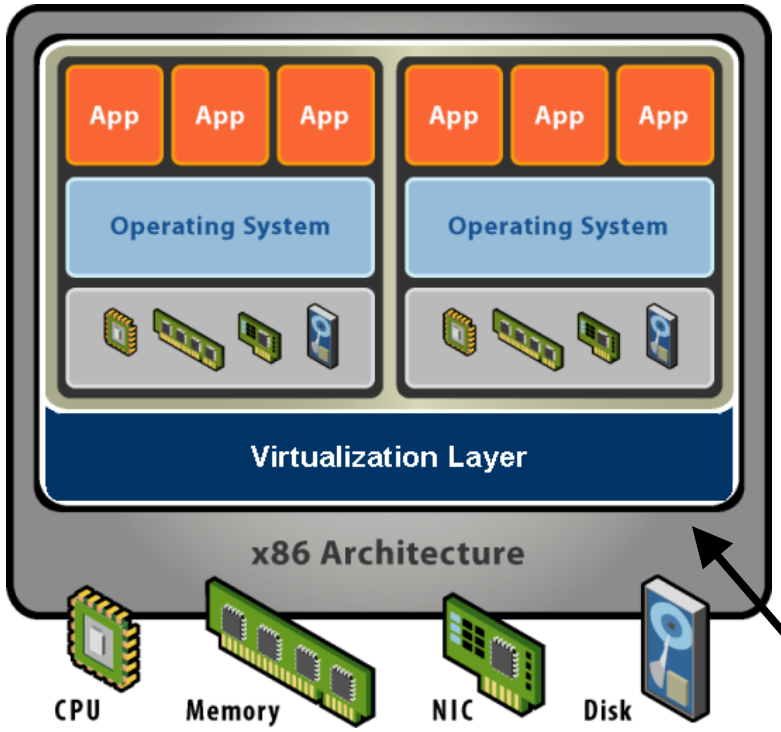
- Disco: Stanford research project (SOSP ’97)
  - Run commodity OSes on scalable multiprocessors
  - Focus on high-end: NUMA, MIPS, IRIX
- Commercial virtual machines for x86 architecture
  - VMware Workstation (now EMC) (1999-)
  - Connectix VirtualPC (now Microsoft)
- Research virtual machines for x86 architecture
  - Xen (SOSP ’03)
  - plex86
- OS-level virtualization
  - FreeBSD Jails, User-mode-linux, UMLinux

Starting Point: A Physical Machine



- Physical Hardware
  - Processors, memory, chipset, I/O devices, etc.
  - Resources often grossly underutilized
- Software
  - Tightly coupled to physical hardware
  - Single active OS instance
  - OS controls hardware

What is a Virtual Machine?



- Software Abstraction
  - Behaves like hardware
  - Encapsulates all OS and application state
- Virtualization Layer
  - Extra level of indirection
  - Decouples hardware, OS
  - Enforces isolation
  - Multiplexes physical hardware across VMs

Virtual Machine Monitor or Hypervisor

Virtualization Properties, Features

- Isolation
  - Fault isolation
  - Performance isolation (+ software isolation, …)
- Encapsulation
  - Cleanly capture all VM state
  - Enables VM snapshots, clones
- Portability
  - Independent of physical hardware
  - Enables migration of live, running VMs (freeze, suspend,…)
- Interposition
  - Transformations on instructions, memory, I/O
  - Enables transparent resource overcommitment, encryption, compression, replication …

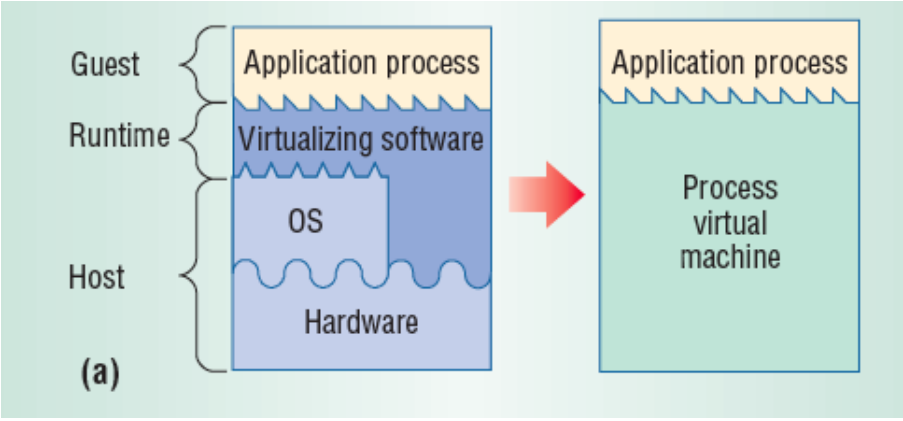
Types of Virtualization

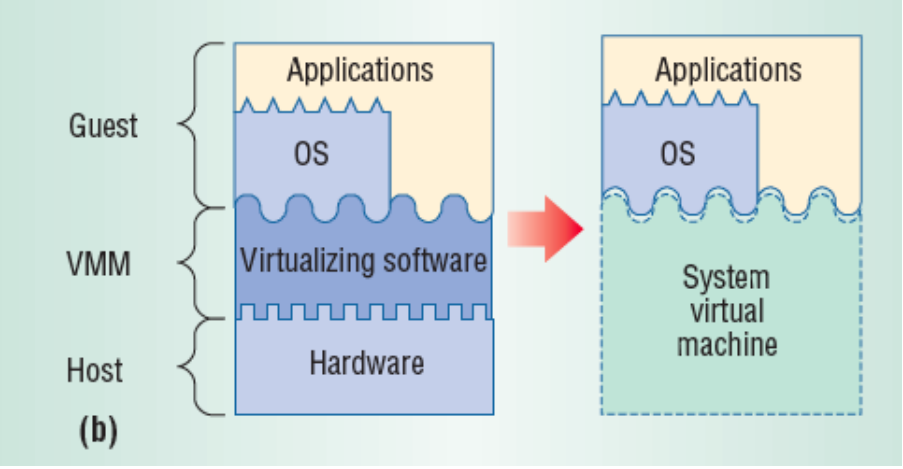
Process Virtualization

- Language-level Java, .NET, Smalltalk
- OS-level processes, Solaris Zones, BSD Jails, Docker Containers
- Cross-ISA emulation Apple 68K-PPC-x86

System Virtualization

- VMware Workstation, Microsoft VPC, Parallels
- VMware ESX, Xen, Microsoft Hyper-V





Language Level Virtualization

- not-really-virtualization but using same techniques, providing similar features
  - Programming language is designed to run within custom-built virtualized environment (e.g. Oracle JVM)
- Virtualization is defined as providing APIs that define a set of features made available to a language and programs written in that language to provide an improved execution environment
  - JVM compiled to run on many systems
  - Programs written in Java run in the JVM no matter the underlying system
  - Similar to **interpreted languages**

Types of VMs – Emulation

- Another (older) way for running one OS on a different OS
  - Virtualization requires underlying CPU to be same as guest was compiled for while Emulation allows guest to run on different CPU
- Need to translate all guest instructions from guest CPU to native CPU
  - Emulation, not virtualization
- Useful when host and guest have different processor architectures
  - Company replacing outdated servers with new servers containing different CPU architecture, but still want to run old applications
- Performance challenge – orders of magnitude slower than native code
  - New machines faster than older machines so can reduce slowdown
- Where do you think it is used still?
  - Very popular – especially in gaming to emulate old consoles

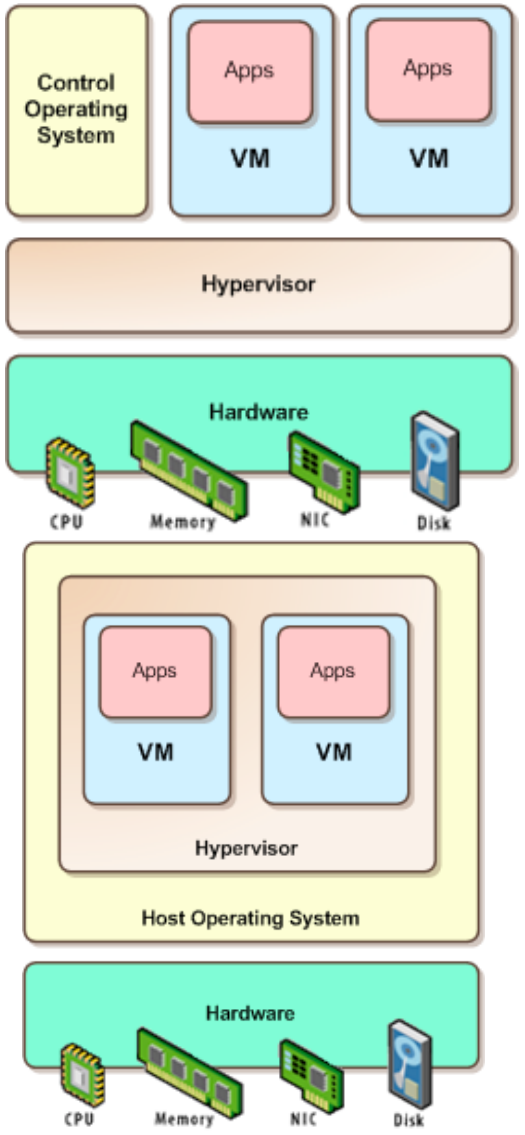
VMs – Application Containers

- Some goals of virtualization are segregation of apps, performance and resource management, easy start, stop, move, and management of them
- Can do those things without full-fledged virtualization
  - If applications compiled for the host operating system, don't need full virtualization to meet these goals
- **Containers / zones** for example create virtual layer between OS and apps
  - Only one kernel running – host OS
  - OS and devices are virtualized, providing resources within zone with impression that they are only processes on system
  - Each zone has its own applications; networking stack, addresses, and ports; user accounts, etc
  - CPU and memory resources divided between zones
    - Zone can have its own scheduler to use those resources

Types of System Virtualization

- Native/Bare metal (Type 1)
  - Higher performance
  - ESX, Xen, HyperV

- Hosted (Type 2)
  - Easier to install
  - Leverage host's device drivers
  - VMware Workstation, Parallels

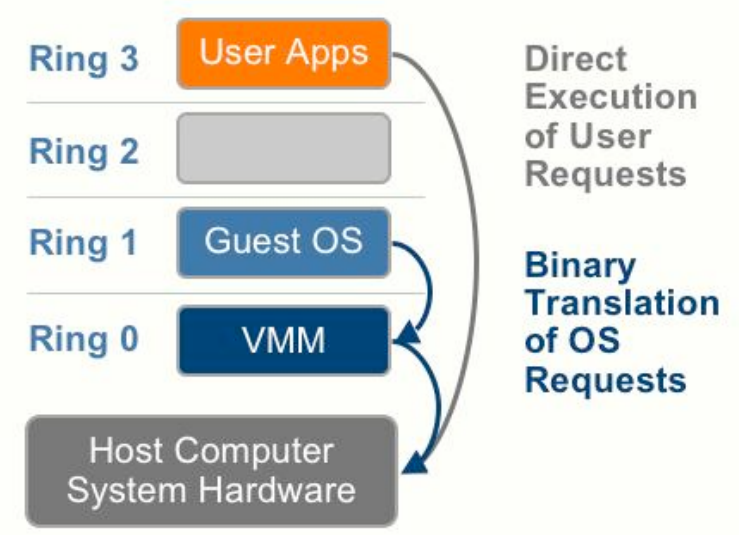


Attribution: http://itechthoughts.wordpress.com/tag/full-virtualization/

Types of Virtualization

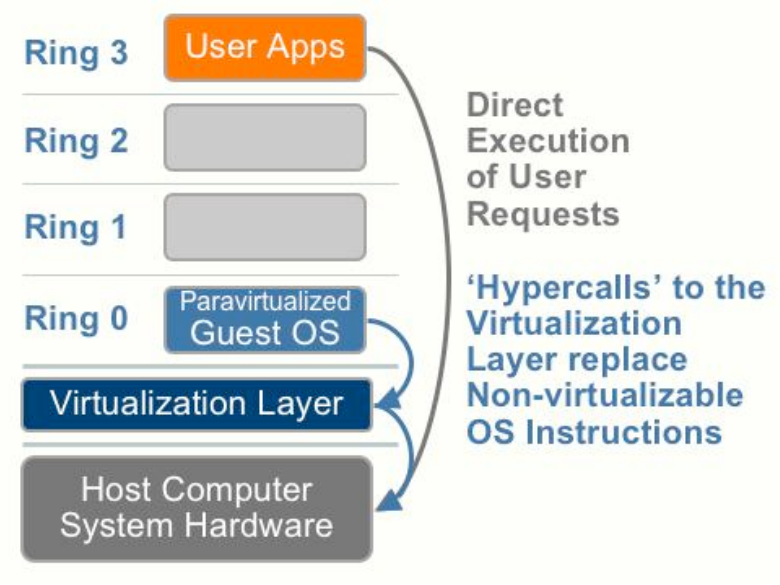
Full virtualization (e.g. VMWare ESX)

- Unmodified OS, virtualization is transparent to OS
- VM looks exactly like a physical machine



Para virtualization (e.g. XEN)

- OS modified to be virtualized,
- Better performance at cost of transparency



Attribution http://forums.techarena.in/guides-tutorials/1104460.htm

What is a Virtual Machine Monitor?

- Classic Definition (Popek and Goldberg '74)

> A virtual machine is taken to be an *efficient, isolated duplicate* of the real machine. We explain these notions through the idea of a *virtual machine monitor* (VMM). See Figure 1. As a piece of software a VMM has three essential characteristics. First, the VMM provides an environment for programs which is essentially identical with the original machine; second, programs run in this environment show at worst only minor decreases in speed; and last, the VMM is in complete control of system resources.

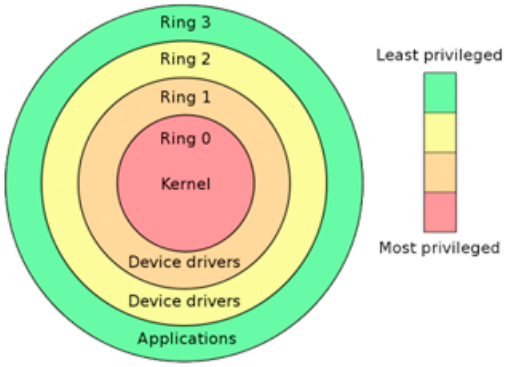
- VMM Properties
  - **Equivalent execution:** Programs running in the virtualized environment run identically to running natively
  - **Performance:** A statistically dominant subset of the instructions must be executed directly on the CPU
  - **Safety and isolation:** A VMM most completely control access to system resources

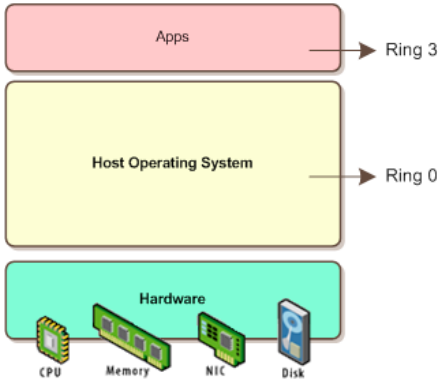
VMM Implementation Goals

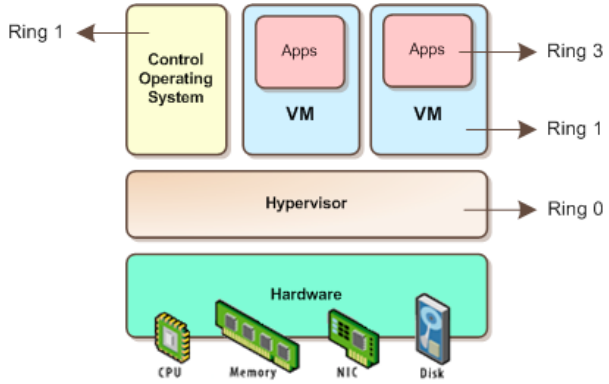
- Should efficiently virtualize the hardware
  - Provide illusion of multiple machines
  - Retain control of the physical machine

- Which subsystems should be virtualized?
  - Processor => Processor Virtualization
  - Memory => Memory Virtualization
  - I/O Devices => I/O virtualization

Processor Virtualization





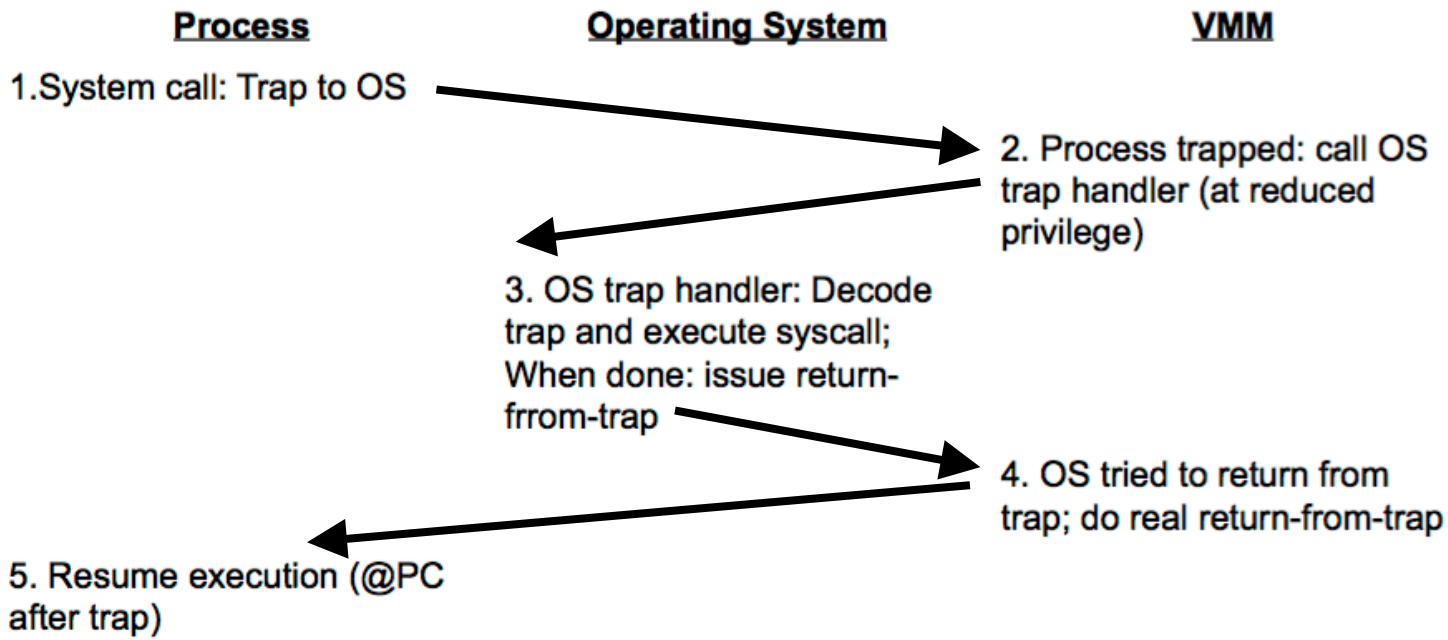


- An architecture is classically/strictly virtualizable if all its sensitive instructions (those that violate safety and encapsulation) are a subset of the privileged instructions.

- all instructions either trap or execute identically
- instructions that access privileged state trap

Attribution: http://itechthoughts.wordpress.com/tag/full-virtualization/

System Call Example



- Run guest operating system deprivileged
- All privileged instructions trap into VMM
- VMM emulates instructions against virtual state
  e.g. disable virtual interrupts, not physical interrupts
- Resume direct execution from next guest instruction

x86 Virtualization challenges

- x86 not Classically Virtualizable
  - x86 ISA has instructions that RD/WR privileged state
  - …But which don't trap in unprivileged mode
- Example: POPF instruction
  - Pop top-of-stack into EFLAGS register
  - EFLAGS.IF bit privileged (interrupt enable flag)
  - POPF silently ignores attempts to alter EFLAGS.IF in unprivileged mode! => no trap to return control to VMM

- Techniques to address inability to virtualize x86
  - Replace non-virtualizable instructions with easily Virtualized ones statically (Paravirtualization)
  - Perform Binary Translation (Full Virtualization)

Virtualizing the CPU

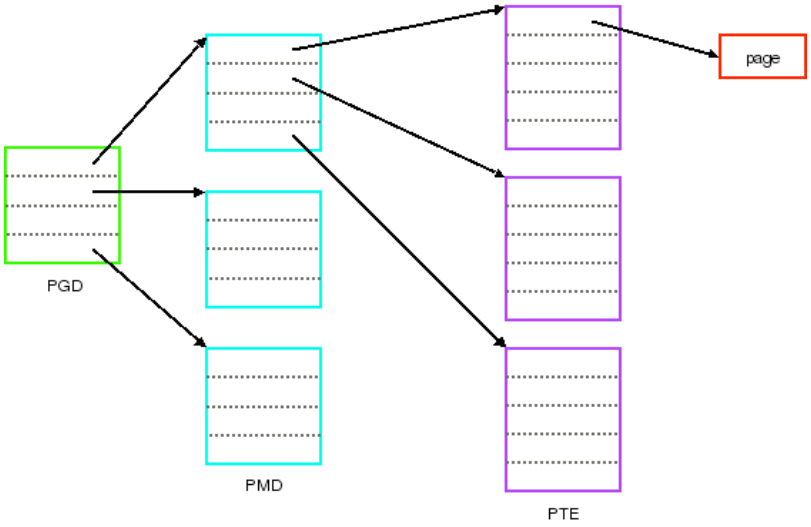
- The VMM still need to multiplex VMs on CPUs
- How could this be done?
  - # Physical CPUs more than #Virtual CPUs?
  - # Virtual CPUs more than #Physical CPUs?
- Timeslice the VMs, each VM runs OS/Apps
- Use simple CPU scheduler
  - Round robin, work-conserving (give extra to other VM)
  - Can oversubscribe and give more #VCPUs that actual

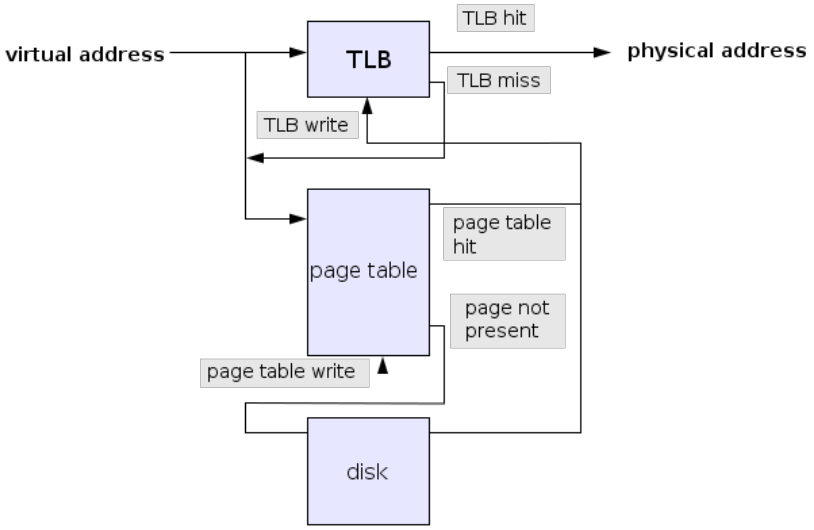
Virtualizing Memory

- OS assumes that it has full control over memory
  - Management: Assumes it owns it all
  - Mapping: Assumes it can map any Virtual-> Physical
- However, VMM partitions memory among VMs
  - VMM needs to assign hardware pages to VMs
  - VMM needs to control mapping for isolation
    - Cannot allow OS to map any Virtual => hardware page
- Hardware-managed TLBs make this difficult
  - On TLB misses, the hardware walks page tables in mem
  - VMM needs to control access by OS to page tables

x86 Memory Management Primer

- The processor operates with virtual addresses
- Physical memory operates with physical addresses
- x86 includes a hardware translation lookaside buffer (TLB)
  - Maps virtual to physical page addresses
- x86 handles TLB misses in HW
  - HW walks the page tables => Inserts virtual to physical mapping





Shadow Page Tables

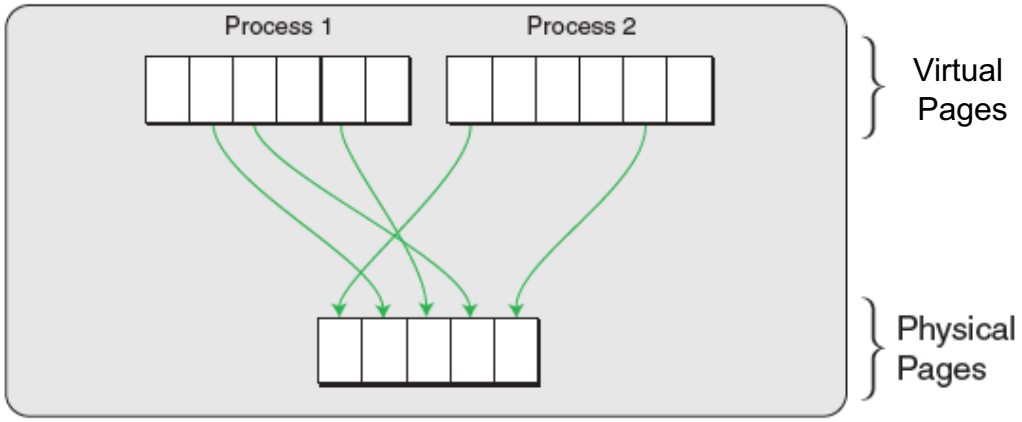
Three abstractions of memory

- **Machine:** actual hardware memory (e.g. 2GB of DRAM)
- **Physical:** abstraction of hardware memory, OS managed
  - E.g. 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)
- **Virtual:** virtual address space
  - Standard 2^{32} address space

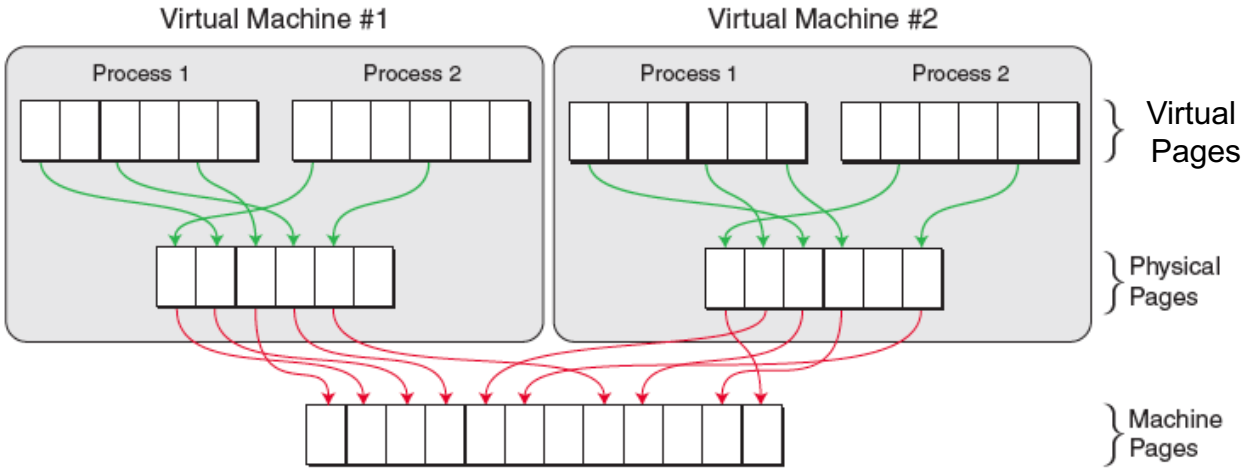
- In each VM, OS creates and manages page tables for its virtual address spaces without modification
  - But these page tables **are not used** by the MMU

Three Abstractions of Memory

- Native



- Virtualized



Shadow Page Tables (continued)

- VMM creates and manages page tables that map virtual pages directly to machine pages
  - These tables are loaded into the MMU on a context switch
  - VMM page tables are the shadow page tables
- VMM needs to keep its V => M tables consistent with changes made by OS to its V=>P tables
  - VMM maps OS page tables as read only
  - When OS writes to page tables, trap to VMM
  - VMM applies write to shadow table and OS table, returns
  - Also known as memory tracing
  - Again, more overhead...

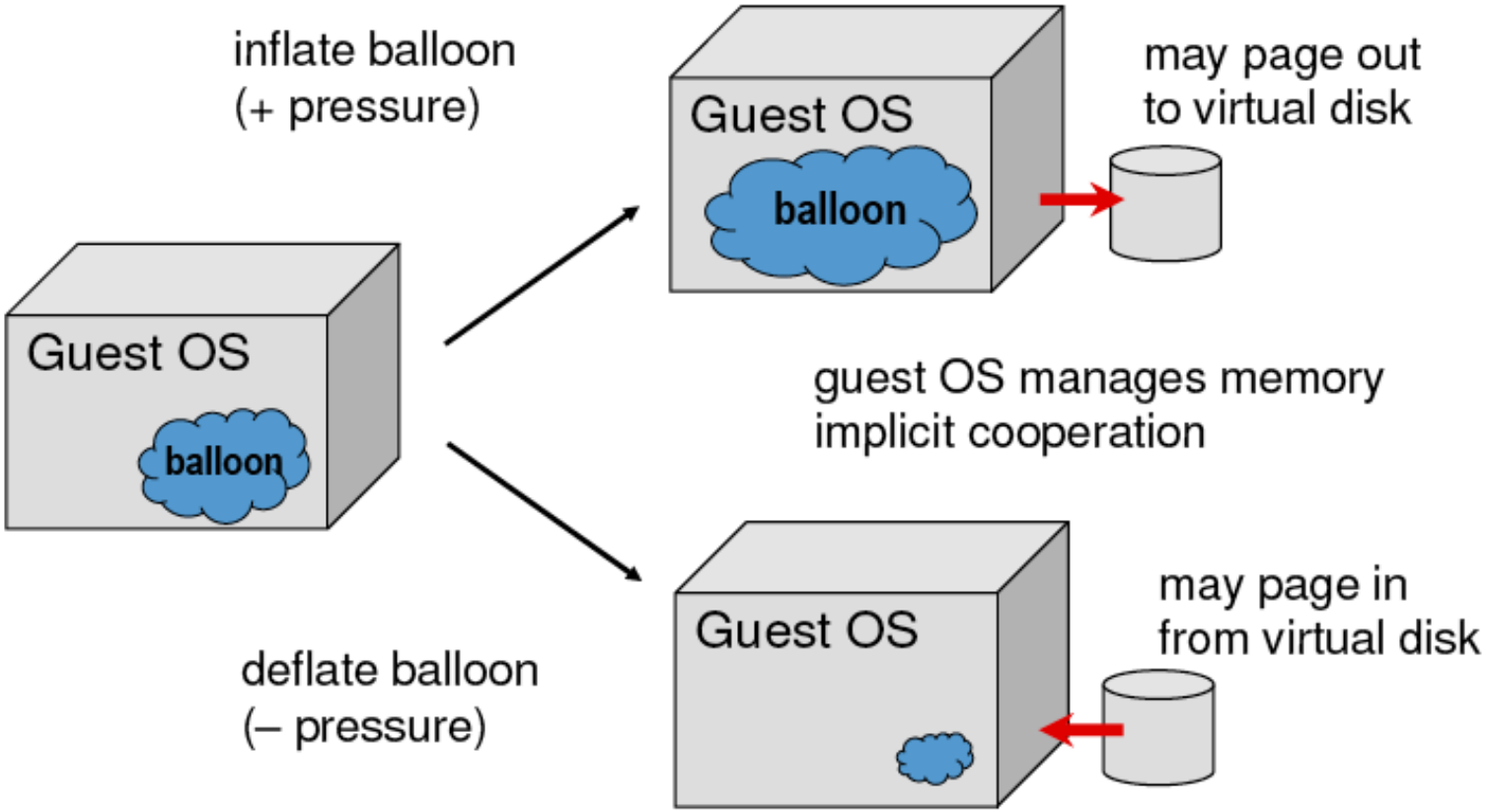
Memory Management / Allocation

- VMMs tend to have simple memory management
  - Static policy: VM gets 8GB at start
    - Dynamic adjustment is hard since OS cannot handle
    - No swapping to disk

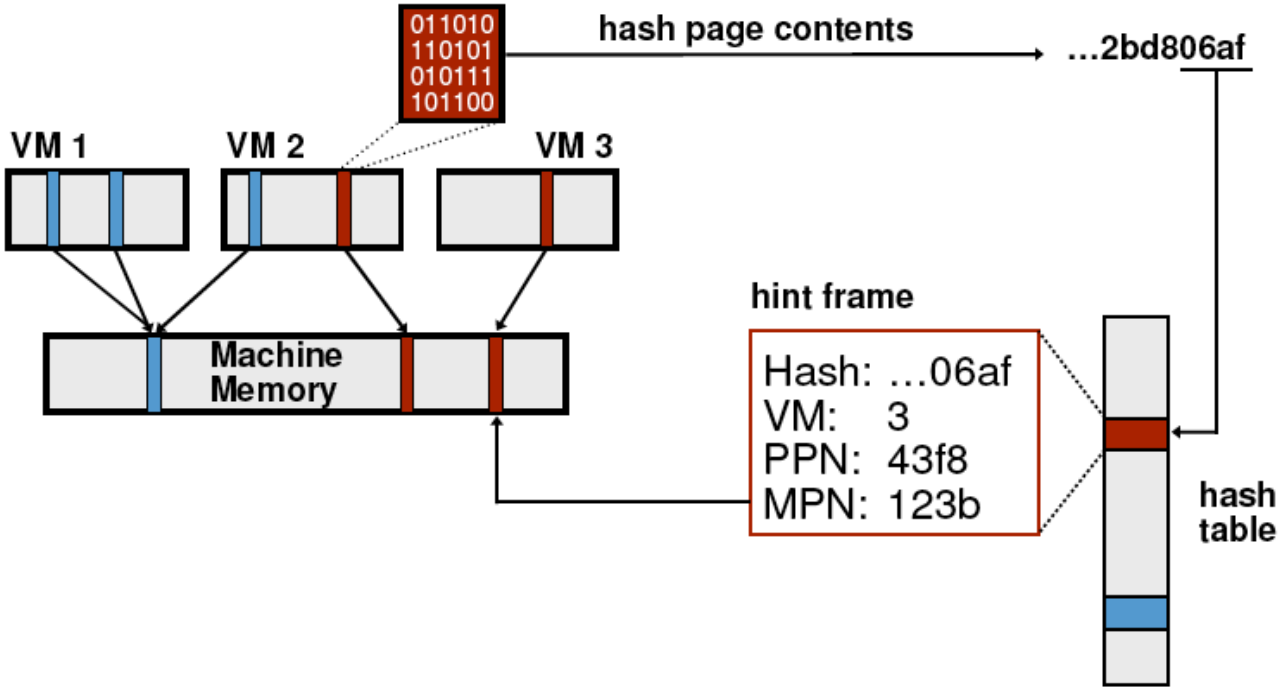
- More sophistication: Overcommit with balooning
  - Baloon driver runs inside OS => consume hardware pages
  - Baloon grows or shrinks (gives back mem to other VMs)

- Even more sophistication: memory de-duplication
  - Identify pages that are shared across VMs!

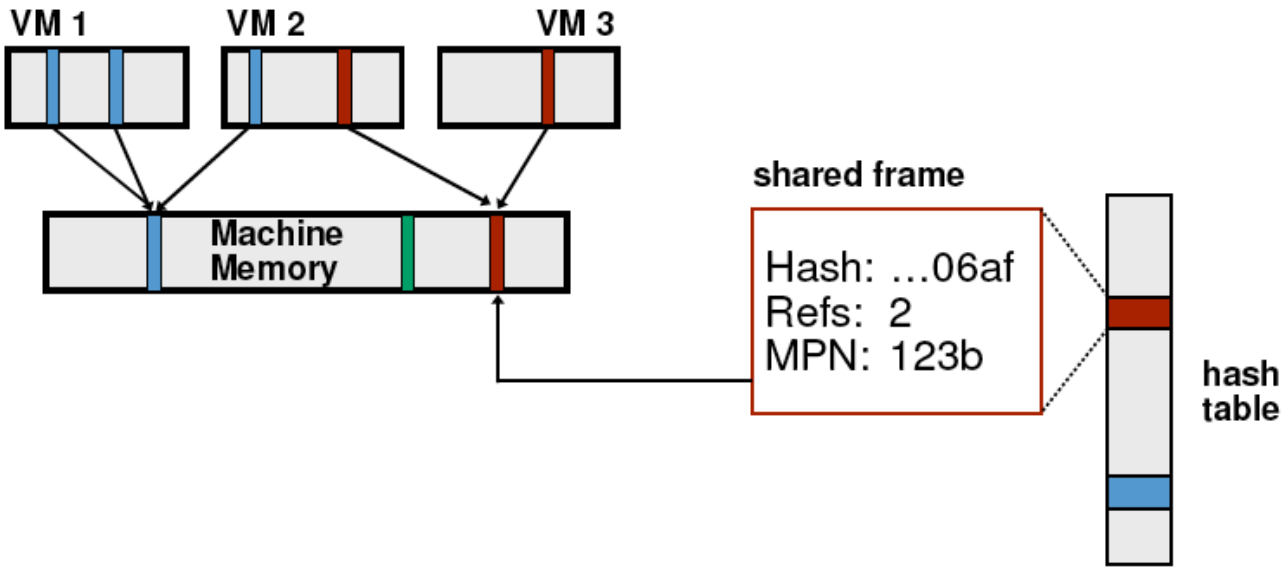
Memory Ballooning



Page Sharing

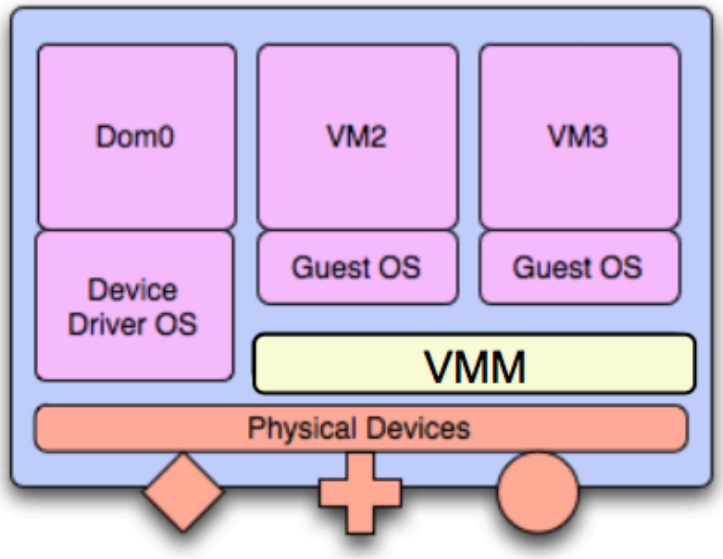


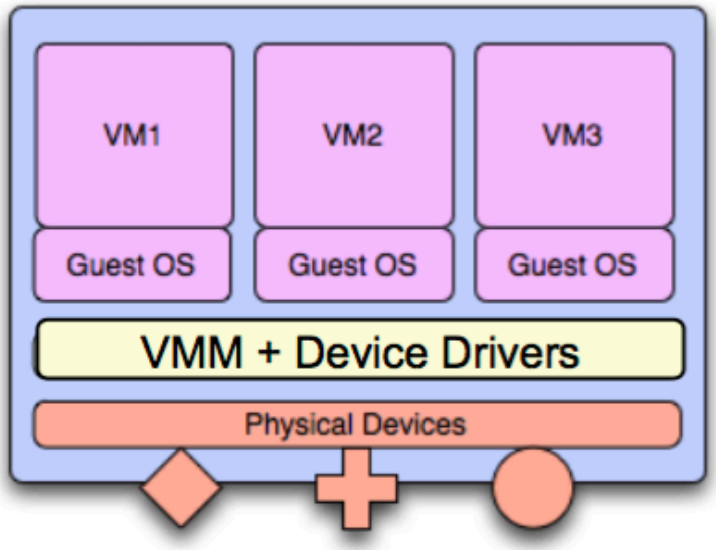
Page Sharing



I/O Virtualization

- **Challenge:** Lots of I/O devices
  - Problem: Writing device drivers for all I/O devices in the VMM layer is not a feasible option
- **Insight:** Device driver already written for popular OSes
- Solution: Present **virtual** I/O devices to **guest** VMs and channel I/O requests to a trusted **host** VM (popular OS)





Virtualizing I/O Devices

- However, overall I/O is complicated for VMMs
  - Many short paths for I/O in OSes for performance
  - Better if hypervisor needs to do less for I/O for guests,
  - Possibilities include direct device access, DMA pass-through, direct interrupt delivery (need H/W support!)
- Networking also complex as VMM and guests all need network access
  - VMM can **bridge** guest to network (direct access)
  - VMM can provide **network address translation (NAT)**
    - NAT address local to machine on which guest is running
    - VMM provides address translation to guest to hide its address

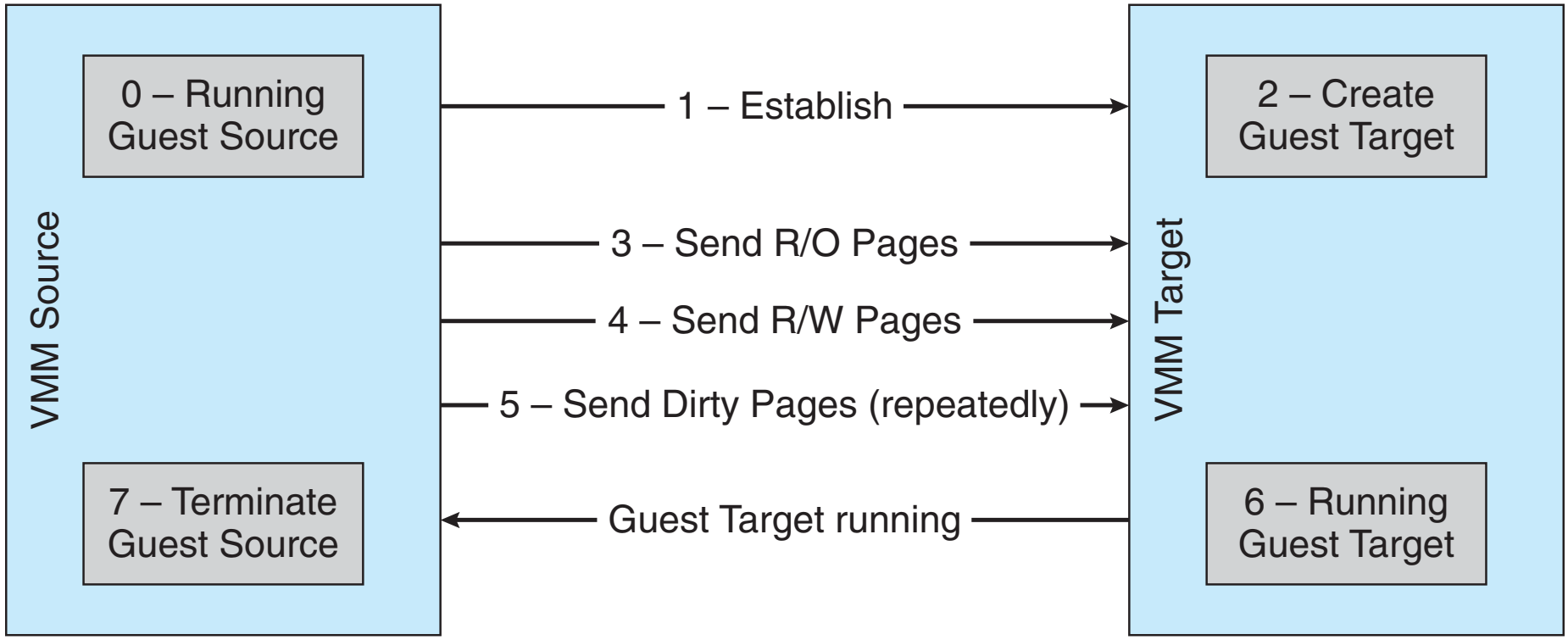
VM Storage Management

- VMM provides both boot disk + other storage
  - Type 1 VMM – storage guest root disks and config information within file system provided by VMM as a **disk image**
  - Type 2 VMM – store as files in the host file system/OS
- Example of supported operations:
  - Duplicate file (clone) -> create new guest
  - Move file to another system -> move guest
  - Convert formats: **Physical-to-Virtual (P-to-V)** and **Virtual-to-Physical (V-to-P)**
- VMM also usually provides access to network attached storage (just networking) => live migration

OS Component – Live Migration

- VMMs allow new functionality like live migration
  - Running guest OS can be moved between systems, without interrupting user access to the guest or its apps
  - Very useful for resource management, no downtime windows, etc
- How does it work?

Live Migration of VMs



OS Component – Live Migration

- VMMs allow new functionality like live migration
  - Running guest OS can be moved between systems, without interrupting user access to the guest or its apps
  - Very useful for resource management, no downtime windows, etc
- How does it work?
  1. Source VMM connects to the target VMM
  2. Target VMM creates a new guest (e.g. create a new VCPU, etc)
  3. Source sends all read-only guest memory pages to the target
  4. Source sends all RD/WR pages to the target, marking them clean
  5. Source repeats step 4, as some pages may be modified => dirty
  6. When cycle of steps 4 and 5 becomes very short, source VMM freezes guest, sends VCPU's final state + other state details, sends final dirty pages, and tells target to start running the guest
  7. Target acknowledges that guest running => source terminates guest

Summary: Virtualization

Resources (CPU, memory, …) are virtualized

- VMM (Hypervisor) has translation tables that map requests for virtual resources to physical resources
- Implementation must have low overheads
  - Types: full system vs para virtualization vs containers
    - different trade-offs

Properties: Isolation, Encapsulation, Portability, Interposition

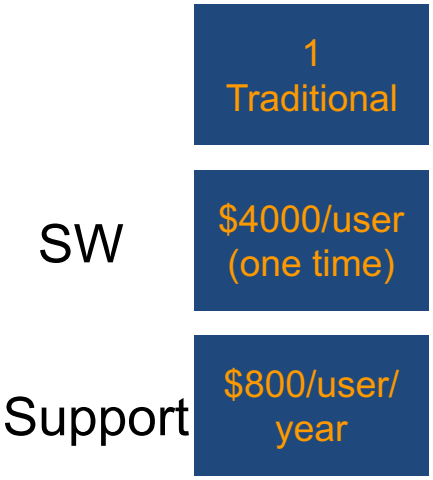
- Multiple VMs can time-share resources -> high utilization
- VMs can be migrated -> balance load, handle planned maintenance, cope with failures

Key enabler of Cloud computing!

Today's outline

1. Cloud computing
2. Virtualization
3. **What's really cloud computing?**

Model 1



- For large software companies (MS, SAP, Oracle) is a good model
- Buy perpetual license, and upgrades, bug fixes, phone support

Model 2

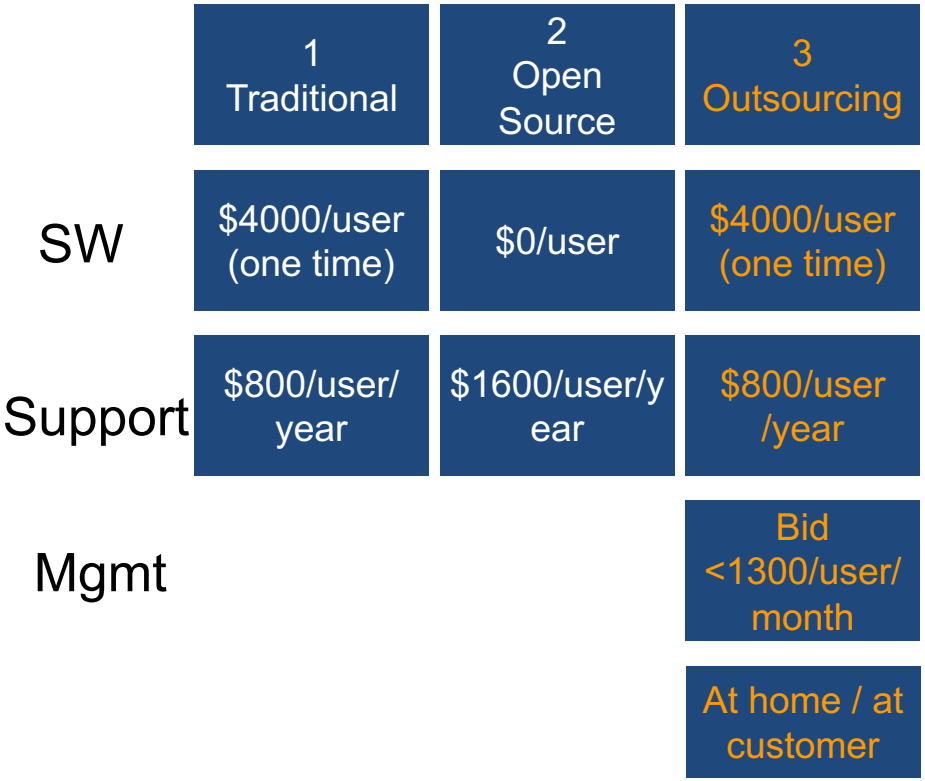
	1 Traditional	2 Open Source
SW	$4000/user (one time)	$0/user
Support	$800/user/ year	$1600/user/year

- Not as many success stories (biggest one probably RedHat)
- But open source is fueling much success for cloud computing

Management Costs

- Cost to manage software is 4x the purchase price per year

Model 3



- Outsourcing became popular in the ’90s as Internet got widespread
- Cost reduction through low-cost labor is hard to sustain
- Human error is #1 cause of computer system failures

Model 4

	1 Traditional	2 Open Source	3 Outsourcing	4 Hybrid
SW	$4000/user (one time)	$0/user	$4000/user (one time)	$4000/user (one time)
Support	$800/user/ year	$1600/user/y ear	$800/user/ year	$800/user/ year
Mgmt			Bid <1300/user/ month	$150/user/ month
			At home / at customer	At home / at customer

- How to decrease costs by 10x?
- Standardization, specialization, repetition → in time automate

Model 5

	1 Traditional	2 Open Source	3 Outsourcing	4 Hybrid	5 Hybrid+
SW	$4000/user (one time)	$0/user	$4000/user (one time)	$4000/user (one time)	$300/user/ month
Support	$800/user/ year	$1600/user/year	$800/user/ year	$800/user/ year	
Mgmt			Bid <1300/user/ month	$150/user/ month	
			At home / at customer	At home / at customer	At home / at customer

- Pure subscription model
- Simply a change of the payment term compared to Model 4

Model 6

	1 Traditional	2 Open Source	3 Outsourcing	4 Hybrid	5 Hybrid+	6 Cloud
SW	$4000/user (one time)	$0/user	$4000/user (one time)	$4000/user (one time)	$300/user/ month	<$100/user/ month
Support	$800/user/ year	$1600/user/year	$800/user/ year	$800/user/ year		
Mgmt			Bid <1300/user/ month	$150/user/ month		
			At home / at customer	At home / at customer	At home / at customer	

- Software companies gone public since 2000 deliver Cloud services
- 10x cost reduction in exchange for higher degree of standardization

Model 7

	1 Traditional	2 Open Source	3 Outsourcing	4 Hybrid	5 Hybrid+	6 Cloud	7 Internet
SW	$4000/user (one time)	$0/user	$4000/user (one time)	$4000/user (one time)	$300/user/ month	<$100/user/ month	Ads Transactions Embedded (<$10/user/ month)
Support	$800/user/ year	$1600/user/year	$800/user/ year	$800/user/ year			
Mgmt			Bid <1300/user/ month	$150/user/ month			
			At home / at customer	At home / at customer	At home / at customer		

- No direct charge; monetization model is asymmetric
- Software is paid through ads, transactions fees, …
- Consumer apps have dramatically lower costs than enterprise apps

Cloud computing is a business model

AWS
us-west-2
prices

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
t2.micro	1	Variable	1	EBS Only	$0.013 per Hour
t2.small	1	Variable	2	EBS Only	$0.026 per Hour
t2.medium	2	Variable	4	EBS Only	$0.052 per Hour
t2.large	2	Variable	8	EBS Only	$0.104 per Hour
m4.large	2	6.5	8	EBS Only	$0.126 per Hour
m4.xlarge	4	13	16	EBS Only	$0.252 per Hour
m4.2xlarge	8	26	32	EBS Only	$0.504 per Hour

Compute
Per service hour
$0.16/hour
(averaged on instance sizes)

Storage
Per GB stored
SSD: $0.10 HDD: $0.05
GB/month

Serverless computing

- Enter post VM and container based clouds
- Cloud execution model in which cloud provider dynamically manages the allocation of machine resources
- Developers don't have to worry about managing, provisioning and maintaining servers
  - Just write and deploy code on the cloud

- It's a form of PaaS but ricing is based on the actual amount of resources consumed by an application
  - AWS charges for compute power in 100 ms increments

- Ultimately a new business model for the cloud

Next lecture topic:

Network file systems