11 – Cloud Computing

Cloud computing

- Computing technology in which data and/or computation are outsourced to a massive scale, multi-user infrastructure that is managed by a third-party.
- Appeared gradually due to two important challenges facing the Web:
  - Scaling
  - Management

Internet Scaling Problem

- The Web and e-commerce were gaining traction
  - Ebay, amazon
  - Yahoo, altavista
- Big Chalange
  - How to scale services
  - 1996 to 1997: eBay grew from 41,000 to 341,000 users

Scaling solutions

- Pre-1995
  - Big, Expensive Computers
- eBay used Sun E-10000 “supermini”
  - 64 processors @250MHz, 64GB RAM, 20TB disk,
  - ~$1M

- Advantages
  - Easy to manage
  - Easy to program
  - Simple failure mode

- Disadvantages
  - Expensive
  - Single point of failure
  - No incremental scalability

Berkeley Network of Workstations

- NOW
  - Leverage many interconnected small, cheap, general-purpose machines for incremental scalability and reliability
  - Typical PC: 200 MHz CPU, 32MB RAM, 4GB disk

Berkley Now

- 1994
  - 4 HP 735's
- 1995
  - 32 Sun SPARC stations
- 1997
  - 60 Sun SPARC-2's
  - Inktomi search engine (bought by yahoo)

NOW adoption

- Everybody builds their own clusters and grows them to handle more and more load
- Examples
  - eBay, Amazon, Google, all .com bubble companies
- Similar to early days of electricity when everyone built their own generator

- (Dis) Advantages
  - Easy to manage
  - Easy to program
  - Simple failure mode

- (Dis)advantages
  - Expensive
  - Single point of failure
  - No incremental scalability

The power plant analogy

- 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…
- Then people started to build large, centralized power plants with very large capacity...
- Power plants are connected to customers by a network
- Usage is metered, and everyone (basically) pays only for what they actually use

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

The Manageability Challenge

- Hard to manage and program large clusters
  - How to write scalable distributed programs?
  - How to debug large scale programs?
  - How to make services reliable?
  - How to architect the network infrastructure?
  - How to provision a cluster to handle peak load?
  - How to administer a huge number of computers?
- Each company had to build own complex software
  - Like each of us building an OS from scratch!

Scalable Clusters

- Very few technically strong companies create powerful scalable and reliable primitives for cluster management and programming
- Examples:
  - Google’s Map/Reduce
  - The Google File System (GFS)
  - Google’s Bigtable
  - Amazon’s Dynamo
  - Distributed debugging and tracing tools
  - Datacenter temperature regulators
  - Scalable distributed communication mechanisms

Cloud computing

- AWS sells resources, expertise, and access to cloud primitives in a pay-for-what-you-use model
  - Resources: CPU, network bandwidth, persistent storage
  - Cloud primitives: Amazon S3, EC2, SQS, Map/Reduce, ...
- Google launches Google Apps for Your Domain
  - Customizable Gmail, Google Docs, Google Calendar under a custom domain
- Google then launches the App Engine
  - Web hosting infrastructure (such infrastructures existed before, but never enjoyed popularity)
- Microsoft launchesAzurein 2009

Three Valuable Commodities

- Giant scale clusters with enormous excess capacity
  - Everybody provisioned for peak
- Expertise for managing and operating clusters at low cost
  - “Economies of scale”
- Complex software to help program/manage clusters
  - Even full applications (e.g., Gmail, Google Calendar, etc.)

Cloud computing

- According to NIST:
  - Cloud computing is a model for enabling
    - convenient, on-demand network access to a shared pool of configurable computing resources
      - 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

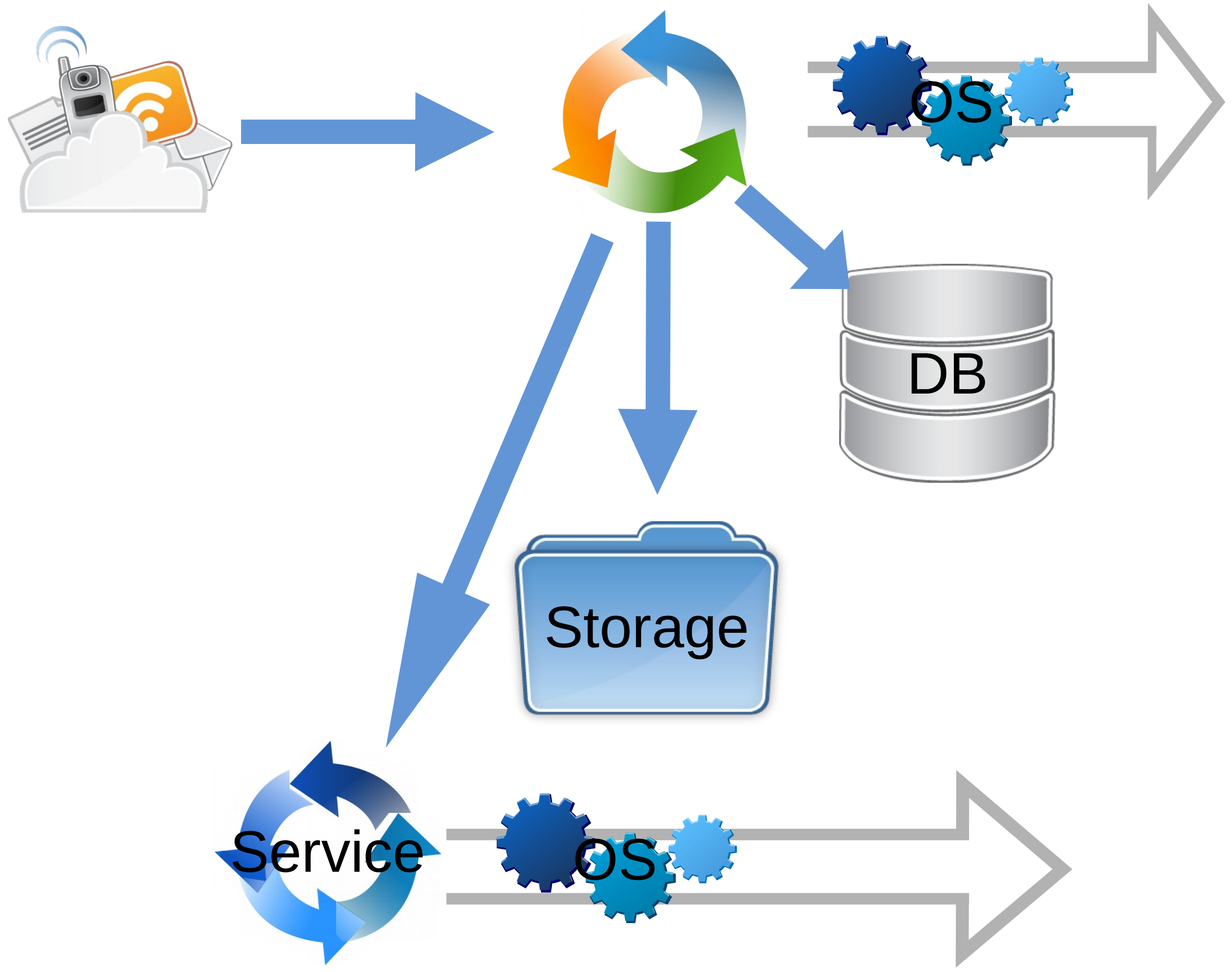
Cloud computing

- 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

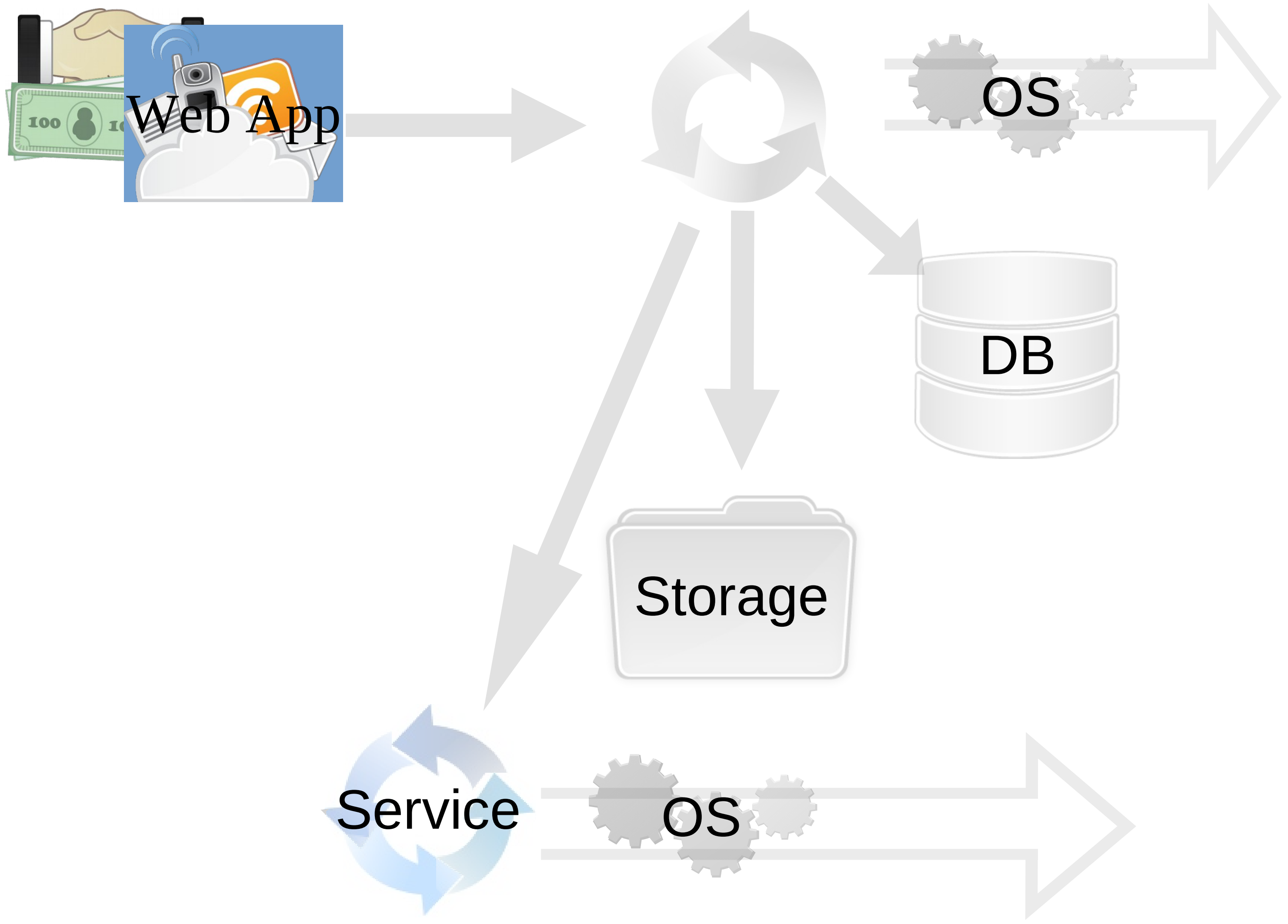
Cloud computing

- 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)
  - Platform as a service (PaaS)
  - Infrastructure as a service (IaaS)
- Other *aaS types have been defined, but are less common
  - Desktop, Backend, Communication, Network, Monitoring, ...

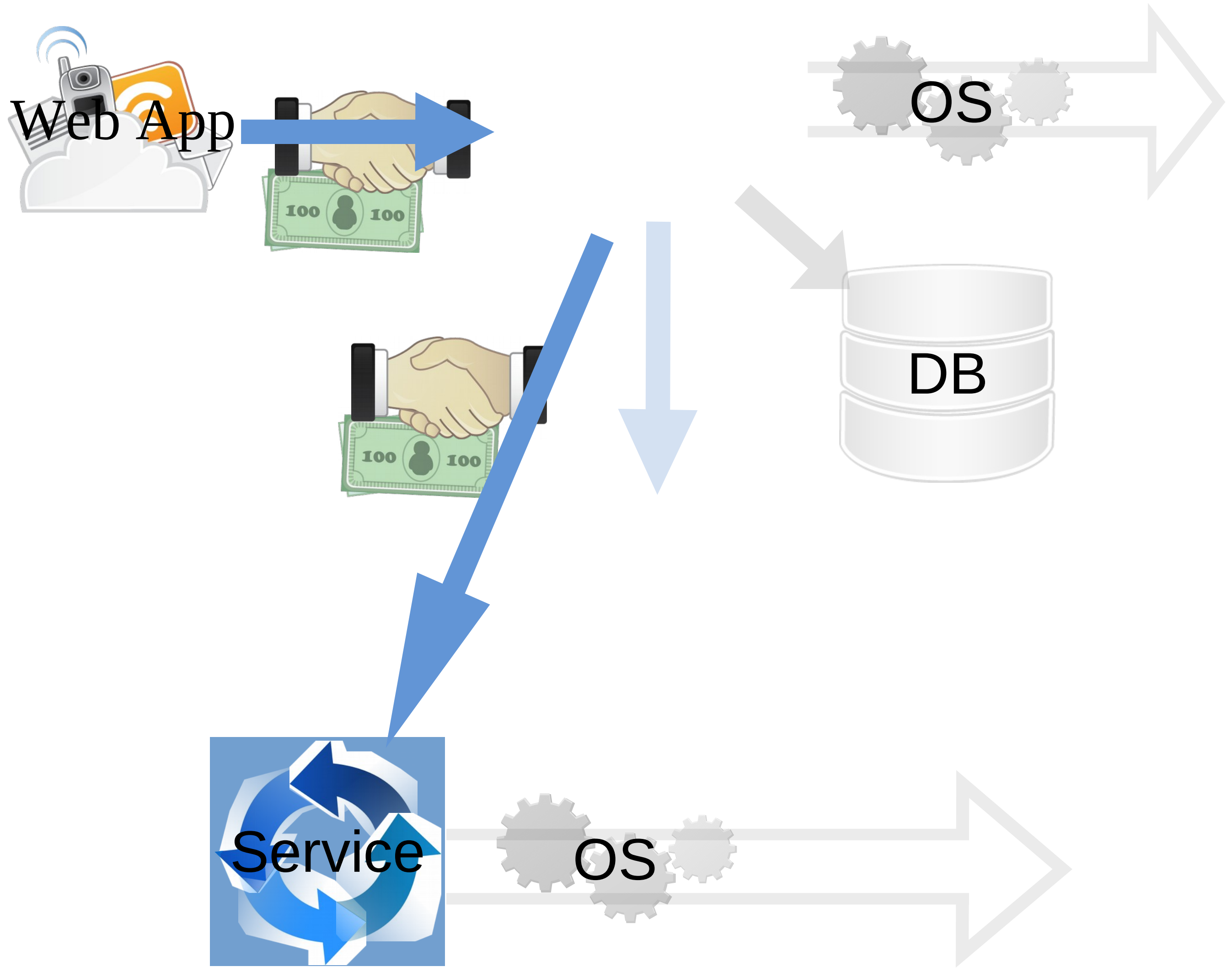
Web architecture



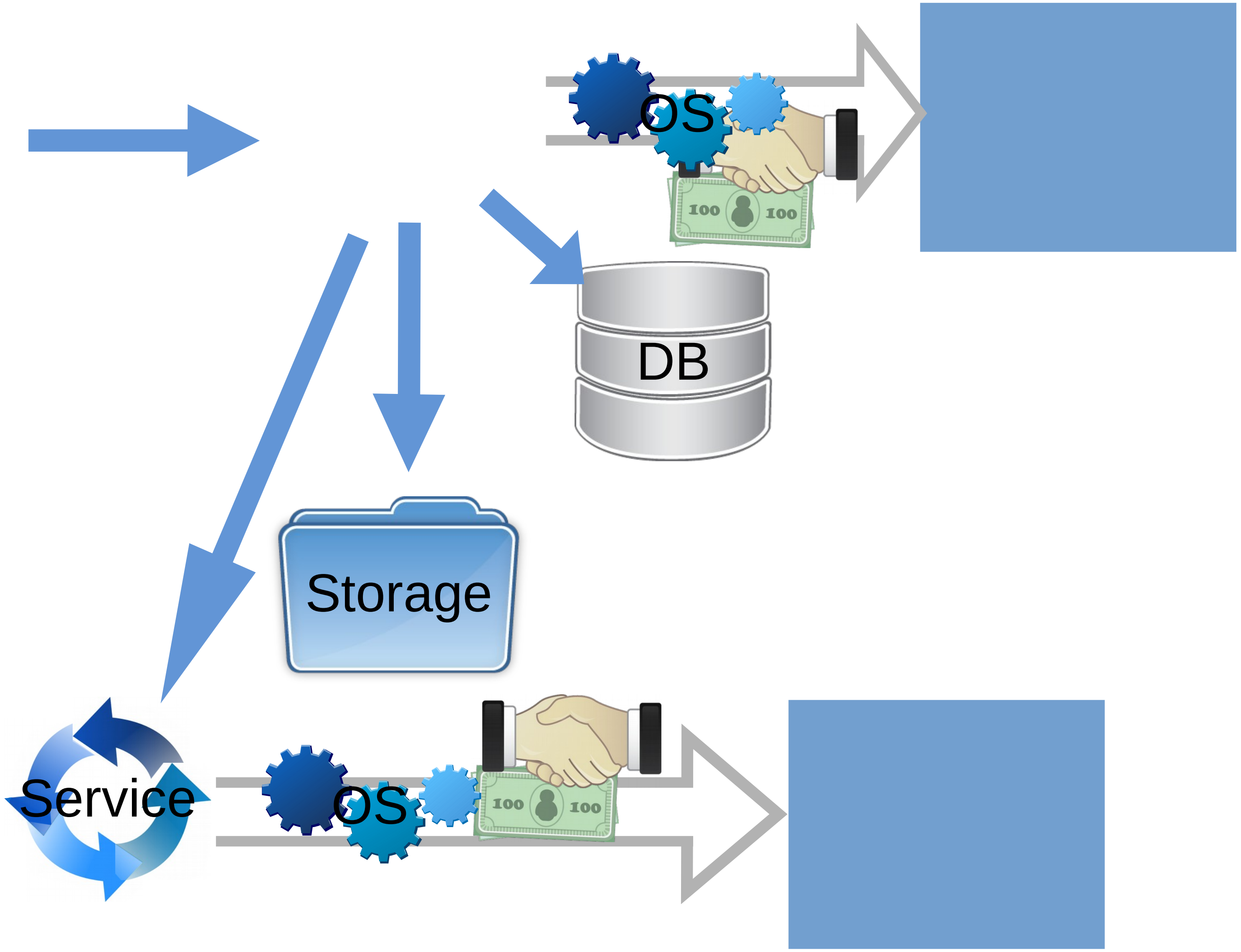
Software as a service



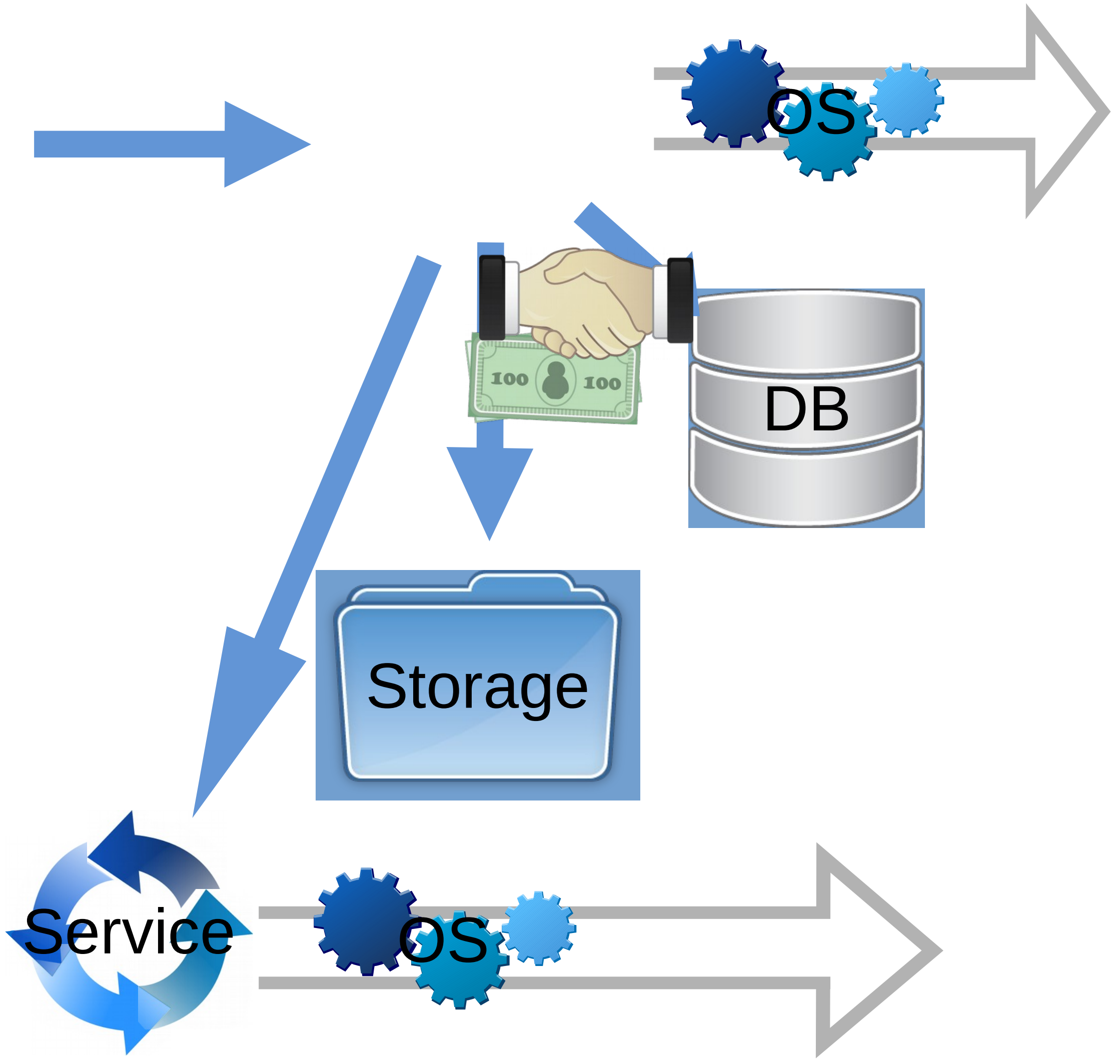
Platform as a service



Infrastructure as a service



Infrastructure as a service



Examples

- Payroll company – monthly variation
  - Has peak loads on its web applications on the last working day of the month
  - Traffic tails off the rest of the month
- Weather company – special events
  - Fairly steady state load most of the time
  - Extreme peak loads when there is a weather event (hurricane, ice storm, etc.)

Examples

- Short-Term Campaign
  - You have a campaign (e.g. product releasel) that needs a short-term increased capacity to manage.
- Small company need generic software
    - Accounting software
  - Buys SW package
  - Hires programmer
  - Rents online version

Examples

- Simulate the interaction of each of millions of compounds with a cancer-related protein.
  - Estimated 341,700 hours of computingI.e. 39 years!
- Solution:
  - Use 10,600 cloud-based compute instances
  - Each of which was multi-core
  - 2 hour setup
  - 9 hours use
  - Peak cost $549.72/hour
  - Total cost $4,362!
- Physical equivalent:
  - 12,000 sq feet data center
  - Costing $44 million

Example

- Startup company - scalability
- Minimal capital available for equipment
- Minimal capital available to hire tech support staff
- No way to predict when their new service will go viral.
- Animoto made its service available via Facebook
  - Demand surge growing from 50 servers to 3,500 servers in three days.
  - Resource needs would doubled every 12 hours for three days.
  - After the peak subsided, traffic fell to a lower level.

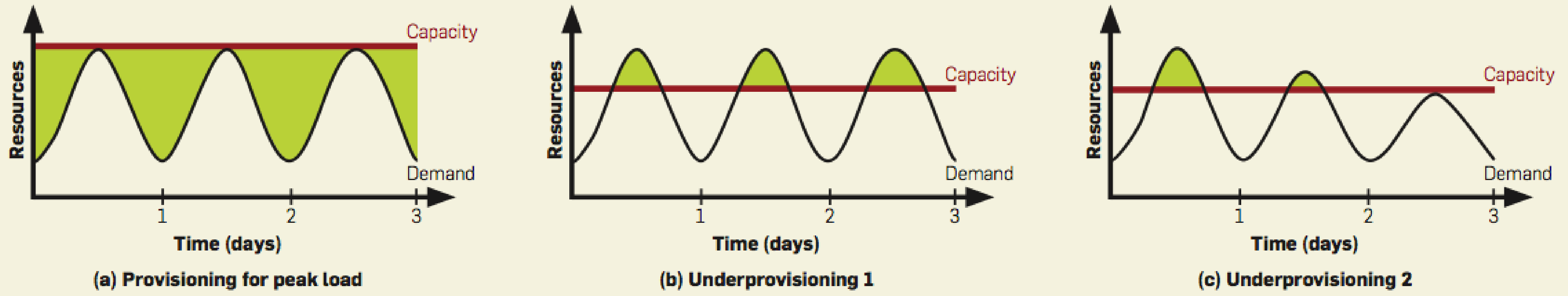
Cloud computing

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

Cloud opportunities

- Over / Under Provisioning
  -

Figure 2. (a) Even if peak load can be correctly anticipated, without elasticity we waste resources (shaded area) during nonpeak times. (b) Underprovisioning case 1: potential revenue from users not served (shaded area) is sacrificed. (c) Underprovisioning case 2: some users desert the site permanently after experiencing poor service; this attrition and possible negative press result in a permanent loss of a portion of the revenue stream.



Cloud opportunities

- Scaling up is difficult
  - Need to order new machines, install them, integrate with existing cluster - can take weeks
  - 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
  - Energy is consumed even when no work is being done
- Many fixed costs, such as construction

Cloud opportunities

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

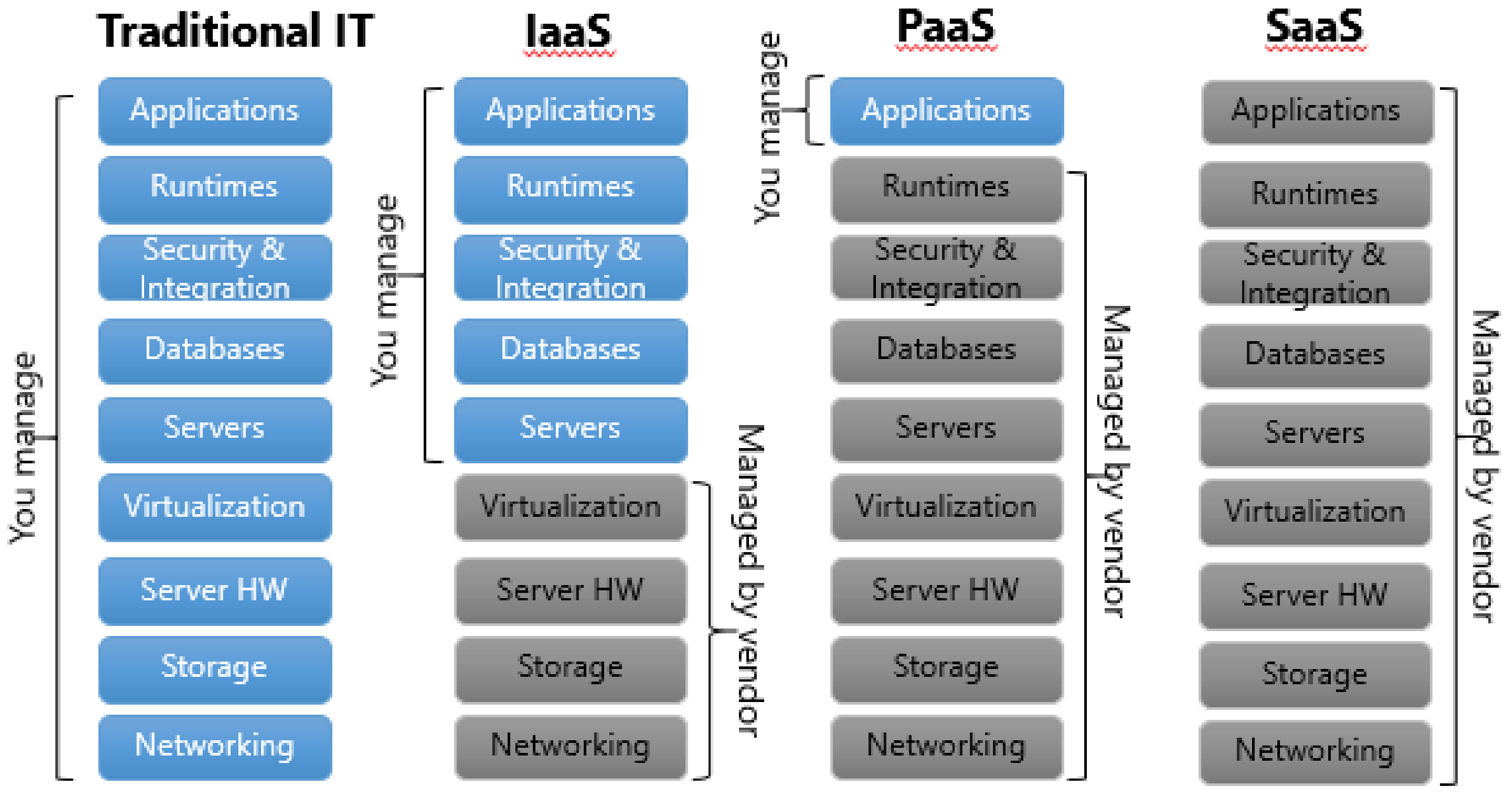
Cloud opportunities

- Infrastructure is Not Core Business
  - You have a need for an extensible software application that
    - scales indefinitely (from your perspective)
    - is available 24/7 worldwide.
- And your core business is not (distributed) software development

Cloud computing

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

Iaas, PaaS, SaaS



Software as a Service

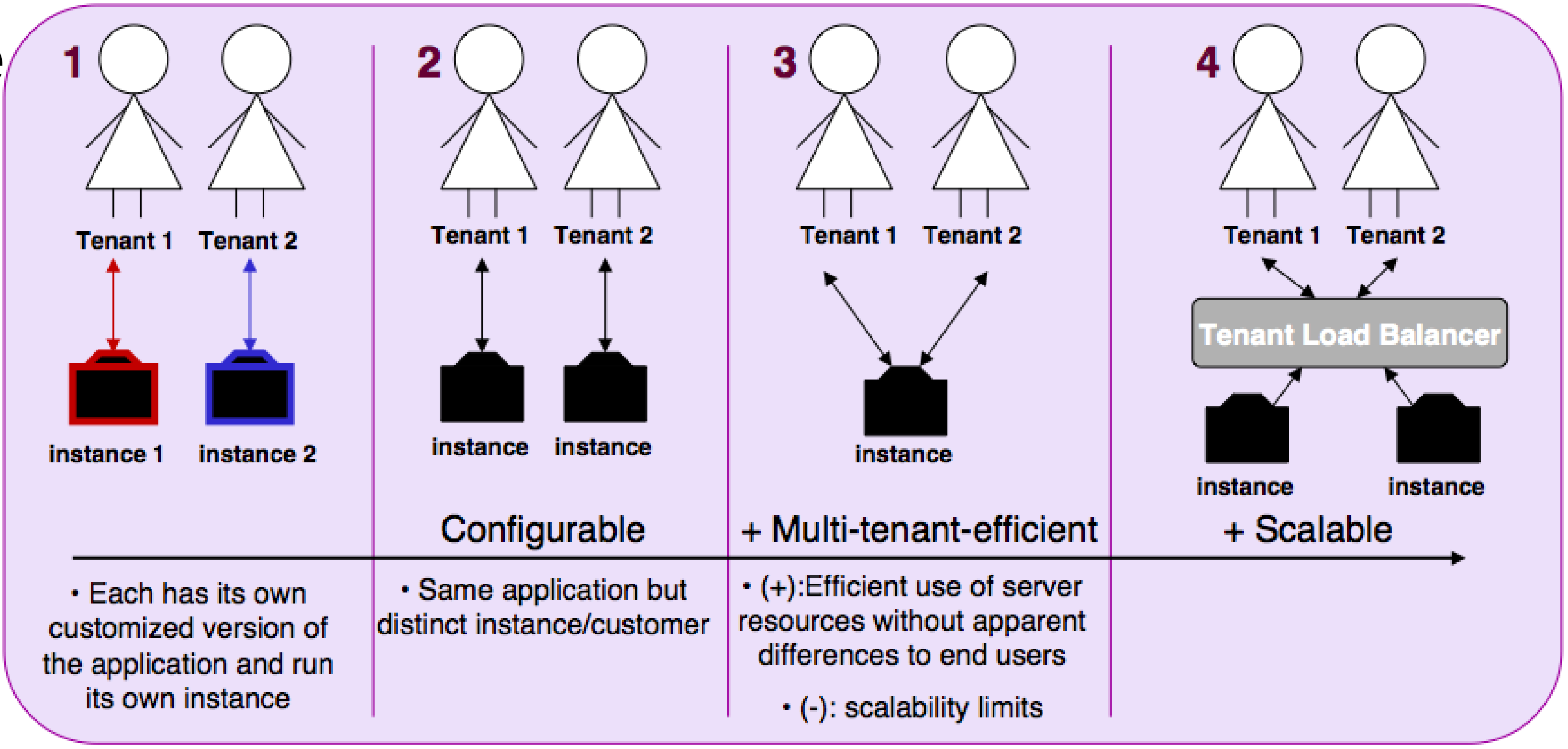
- Started around 1999
- Application is licensed to a customer as a service on demand
- Software Delivery Model:
  - Hosted on The vendor’s Web servers
  - Downloaded at the consumer’s device and disabled when on-demand contract is over
  - Executed on the browser
-

Software as a Service

- Models
  - Accessed through the Web
  - SaaS Vendor Support
  - SaaS Subscription Pricing
  - SaaS Low Customization
  - SaaS Managed Upgrades
  - SaaS Shift to service-based mentality
  - SaaS Success based revenue model

Software as a Service

- SaaS Architectural Maturity Levels
  - Ad-Hoc/Custom (no maturity
  - Configurability
  - Multitenant Efficiency
  - Scalable
-


1
Tenant 1
Tenant 2
instance 1
instance 2
2
Tenant 1
Tenant 2
instance
instance
Configurable
3
Tenant 1
Tenant 2
instance
+ Multi-tenant-efficient
4
Tenant 1
Tenant 2
Tenant Load Balancer
instance
instance
+ Scalable
• Each has its own customized version of the application and run its own instance
• Same application but distinct instance/customer
• (+):Efficient use of server resources without apparent differences to end users
• (-): scalability limits

Consumer

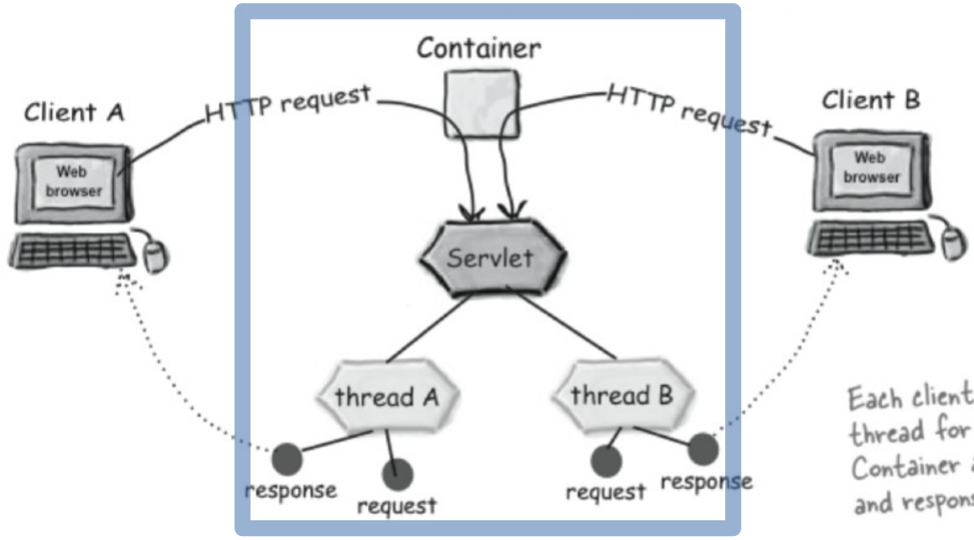
- SaaS
  - End user (mail, project, accountting...)
- PaaS
  - Programmer
- IAAS
  - System Administrator

Platform as a service

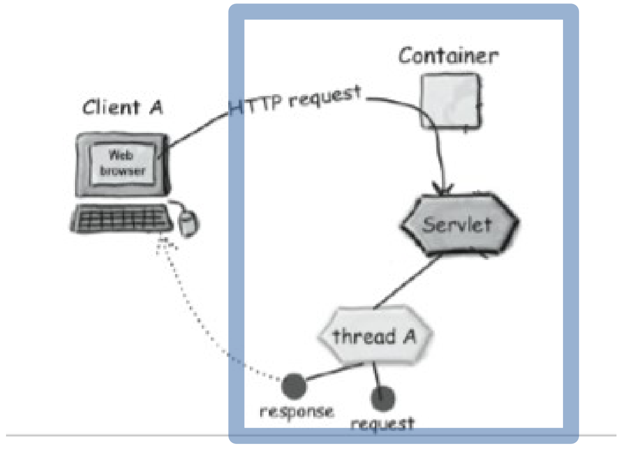
- Elastic on-demand platform for applications
  - that completely abstracts away the underlying infrastructure
  - with the application scaling seamlessly with the platform.
- Application scales with the platform that offers “infinite” scalability
- No human intervention
- No hardware in the discussion

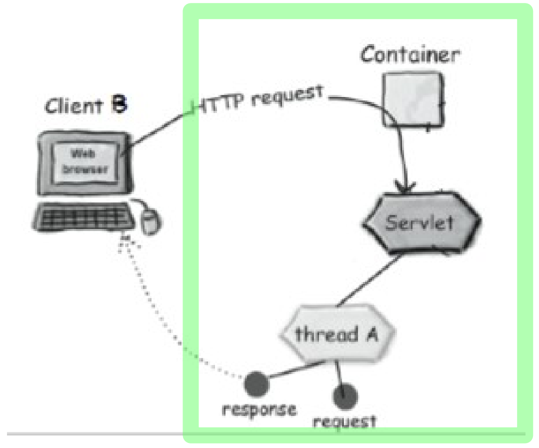
Platform as a service

- Programming languages, tools, software systems
  - Defined by the system
  - provide a platform upon which a customer can build applications.
- The consumer does not manage or control the underlying computing infrastructure, but has control over the deployed applications



Each client gets a separate thread for each request, and the Container allocates new request and response objects.





Google App Engine (GAE)

- Platform as a Service (PaaS)
  - for developing and hosting web applications in Google-managed data centers.
  - lets you run web applications on Google's infrastructure.
- Easy to build.
- Easy to maintain.
- Easy to scale as the traffic and storage needs grow.
-

Google App Engine

- Jav Programming languages support
  - App Engine runs JAVA apps on a JAVA 7 virtual machine
- Uses JAVA Servlet standard for web applications:
  - Servlet classes
  - Java Server Pages (JSP)
  - Static and data files
  - Deployment descriptor (web.xml)
  - Other configuration files

Google App Engine

- Python Programming languages support
  - Uses WSGI (Web Server Gateway Interface) standard.
- Python applications can be written using:
  - Webapp2 framework
  - Django framework
  - Any python code that uses the CGI (Common Gateway Interface) standard.

Google App Engine

- Othe languages support
  - PhP Programming languages support
  - Google’s Go Programming languages support
  - Node.js
  - Ruby
  -

Google App Engine (Data storage)

- App Engine Datastore:
  - NoSQL schema-less object based data storage, with a query engine and atomic transactions.
  - Data object is called a “Entity” that has a kind (~ table name) and a set of   properties (~ column names).
  - JAVA JDO/ JPA interfaces and Python datastore interfaces.
- Google cloud SQL:
  - Provides a relational SQL database service.
  - Similar to MySQL RDBMS.

Google App Engine (Data storage)

- Google cloud store:
  - RESTful service for storing and querying data.
  - Fast, scalable and highly available solution.
  - Provides Multiple layers of redundancy. All data is replicated to multiple data centers.
  - Provides different levels of access control.
  - HTTP based APIs.

Google App Engine (services)

- App Engine also provides a variety of services to perform common operations when managing your application.
  - URL Fetch: Facilitates the application’s access to resources on the internet, such as web services or data.
  - Mail: Facilitates the application to send e-mail messages using Google infrastructure.
  - Memcache: High performance in-memory key-value storage.
    - Can be used to store temporary data which doesn’t need to be persisted.

Google App Engine (Security)

- The sandbox:
  - All hosted applications run in a secure environment that provides limited access to the underlying operating system.
  - Sandbox isolates the application in its own secure, reliable environment that is independent of hardware, operating system and physical location of a web server.
  - Limitations imposed by sandbox (for security):
    - An application can only access other computers over internet using the provided URL fetch and email services.
    - Other computers can only connect to the application through HTTP/ HTTPS requests on the standard ports (80/ 443).
    - Applications cannot write to local file system in any of the runtime environments.
    - Application code runs only in response to a web request, a queued task or a scheduled task and must return the response data within 60 seconds.
    - A request handler cannot spawn a sub-process or execute code after the response has been sent.

Google App Engine

- Use App Engine when:
  - You don’t want to get troubled for setting up a server.
  - You want instant for-free nearly infinite scalability support.
  - Your application’s traffic is spiky and rather unpredictable.
  - You don't feel like taking care of your own server monitoring tools.
  - You need pricing that fits your actual usage and isn't time-slot based (App engine provides pay-per-drink cost model).
  - You are able to chunk long tasks into 60 second pieces.
  - You are able to work without direct access to local file system.

Infrastructure as a Service

- Processing, storage, networks, and other fundamental computing resources
- The consumer can run arbitrary software
  - Including operating systems and applications
- Cloud provides raw computing resources
  - Virtual machine, blade server, hard disk, ...
  - Customer pays SaaS provider for the service;
  - Examples: Amazon Web Services, Rackspace Cloud, GoGrid

Amazon Cloud: EC2

- Amazon Elastic Compute Cloud (Amazon EC2)
  - web service that provides resizeable computing capacity—literally, servers in Amazon's data centers—that you use to build and host your software systems.
  - You can access the components and features that EC2 provides using a web-based GUI, command line tools, and APIs.
- Use and pay for only the capacity that you need.
  - Eliminates the need to make large and expensive hardware purchases, reduces the need to forecast traffic, and enables you to automatically scale your IT resources to deal with changes in requirements or spikes in popularity related to your application or service.
- Components of EC2:
  - Amazon Machine Images and Instances,Storage, Databases, Networking and Security, Monitoring, Auto-Scaling and Load Balancing

Amazon Cloud EC2: AMI

- An Amazon Machine Image (AMI)
  - template that contains a software configuration (operating system, application server, and applications).
- Instances, which are running copies of the AMI.
- Your instances keep running until you stop or you terminate them, or until they fail.
- An instance type is essentially a hardware archetype. As illustrated in the following figure, you select a particular instance type based on the amount of memory and computing power you need for the application or software that you plan to run on the instance.
- Amazon publishes many AMIs that contain common software configurations for public use. In addition, members of the AWS developer community have published their own custom AMIs

Amazon Cloud EC2: Storage

- To store data, Amazon EC2 offers the following storage options:
  - Amazon Elastic Block Store (Amazon EBS)
  - Amazon EC2 Instance Store
  - Amazon Simple Storage Service (Amazon S3)
- Amazon EBS
  - instances with persistent, block-level storage. Amazon EBS volumes are essentially hard disks that you can attach to a running instance.
  - Amazon EBS is particularly suited for applications that require a database, file system, or access to raw block-level storage.
  - You can also detach a volume from an instance and attach it to a different one, as illustrated in the following figure.
- Instance Store
  - storage that doesn't persist if the instance is stopped or terminated. Instance store is an option for inexpensive temporary storage

Amazon Cloud EC2: Storage

- You can also detach a volume from an instance and attach it to a different one, as illustrated in the following figure.
- Instance Store
- All instance types, with the exception of Micro instances, offer instance store. This is storage that doesn't persist if the instance is stopped or terminated. Instance store is an option for inexpensive temporary storage. You can use instance store volumes if you don't require data persistence.
- Amazon S3
- Amazon S3 is storage for the Internet. It provides a simple web service interface that enables you to store and retrieve any amount of data from anywhere on the web.
-
- .

Amazon Cloud S3

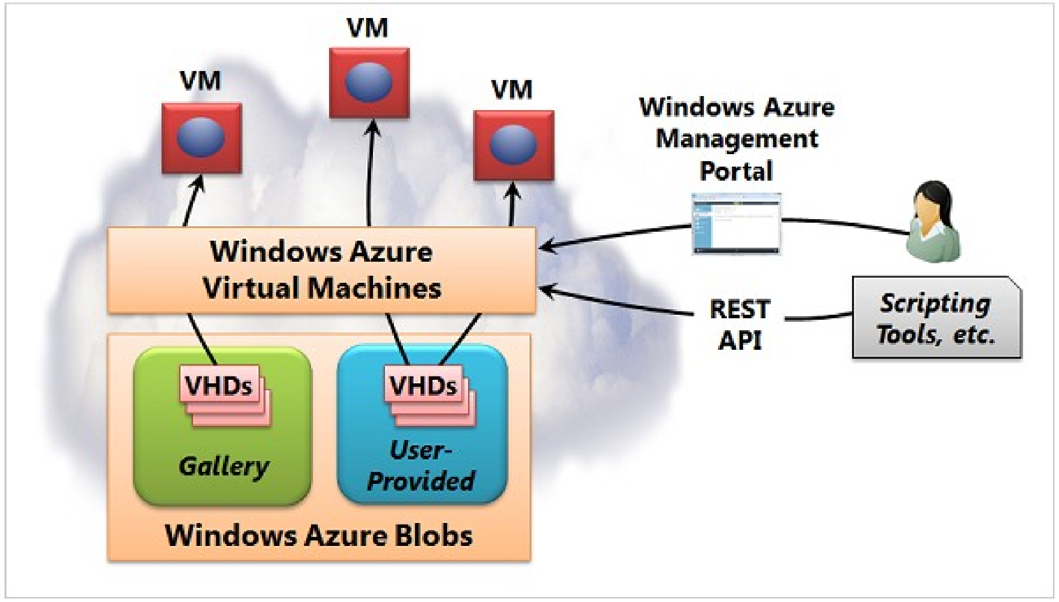
- Amazon S3 Functionality
  - Write, read, and delete objects containing from 1 byte to 5 terabytes of data each.
  - Each object is stored in a bucket and retrieved via a unique, developer-assigned key.
  - Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users.
  - Options for secure data upload/download and encryption of data at rest are provided for additional data protection.
  - Uses standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.

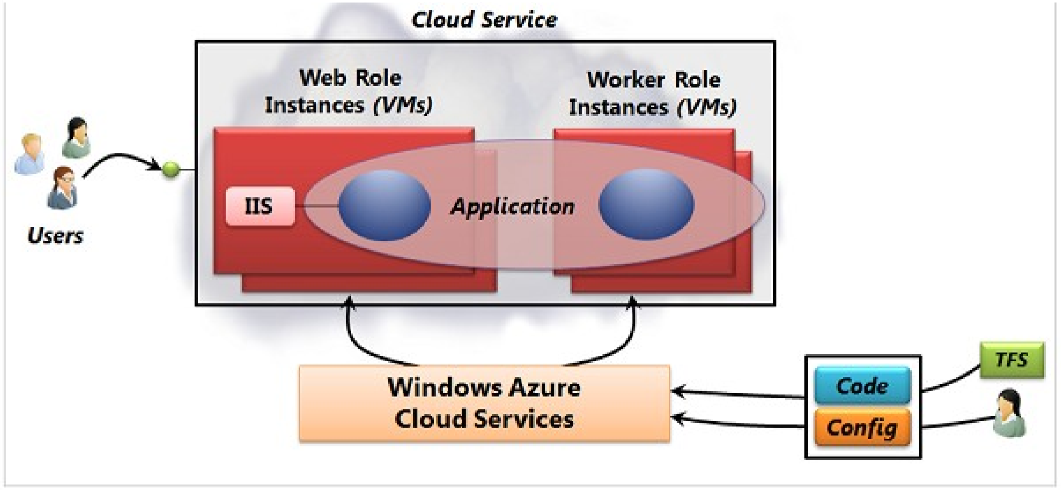
Monitoring, Auto Scaling, and Load Balancing

- Monitor basic statistics for instances and Amazon EBS volumes.
- Automatically scale EC2 capacity up or down according to conditions defined by the user.
- Automatically distribute incoming application traffic across multiple EC2 instances.
  - It detects unhealthy instances and reroutes traffic to healthy instances until the unhealthy instances have been restored.
  - Elastic Load Balancing automatically scales its request handling capacity in response to incoming traffic.
- Elastic Load Balancing provides several different interfaces that can be used to manage a user's load balancers.
  - Users can create, access, and manage their load balancers using the AWS Management Console, the command line interface (CLI), or the Query API. Users need to install the command line interface and the Query API before they can be used.

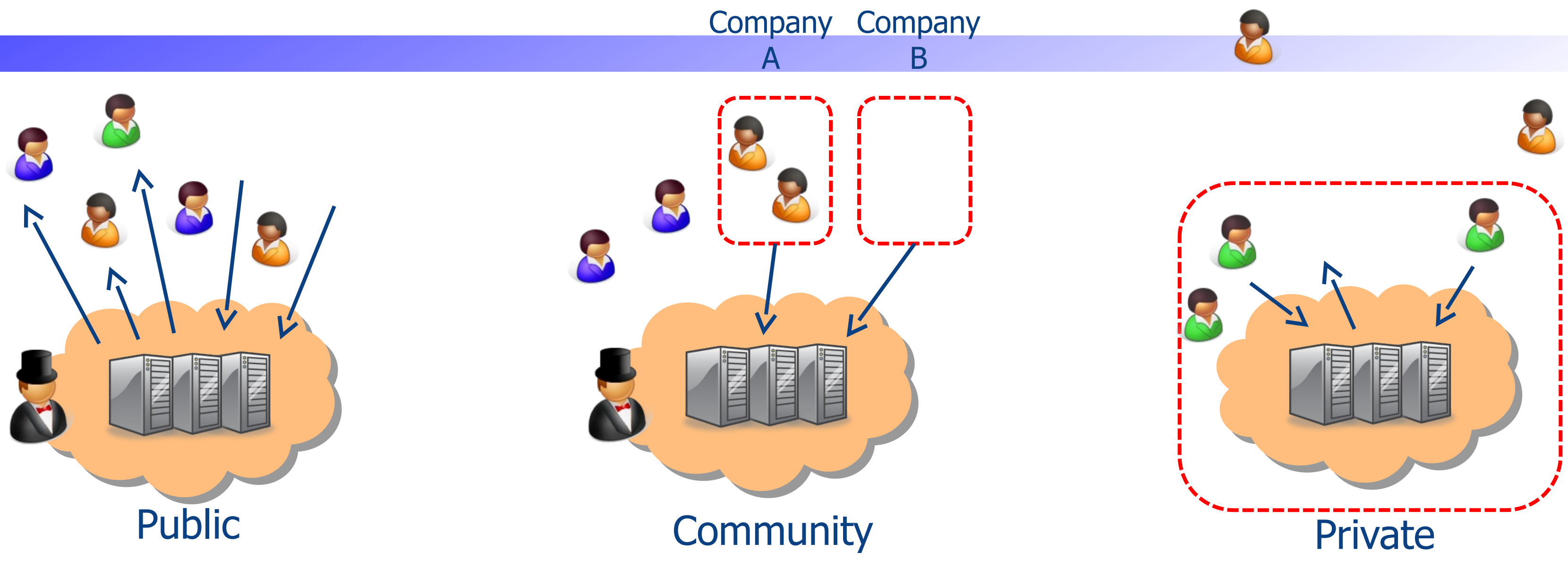
Microsoft Azure

- IAAS
  - Azure Virtual Machines
- PAAS
  - Azure Cloud Services



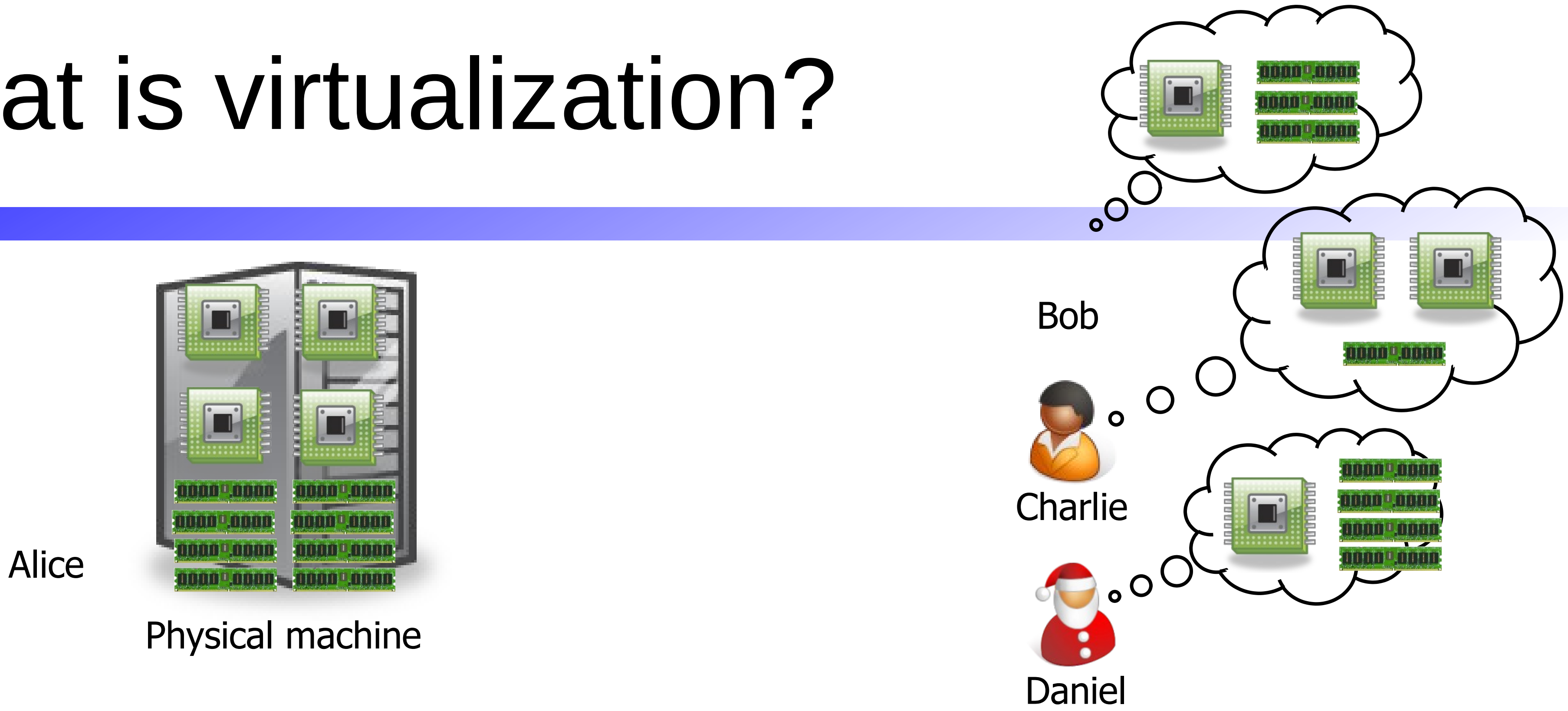


Private/hybrid/community clouds



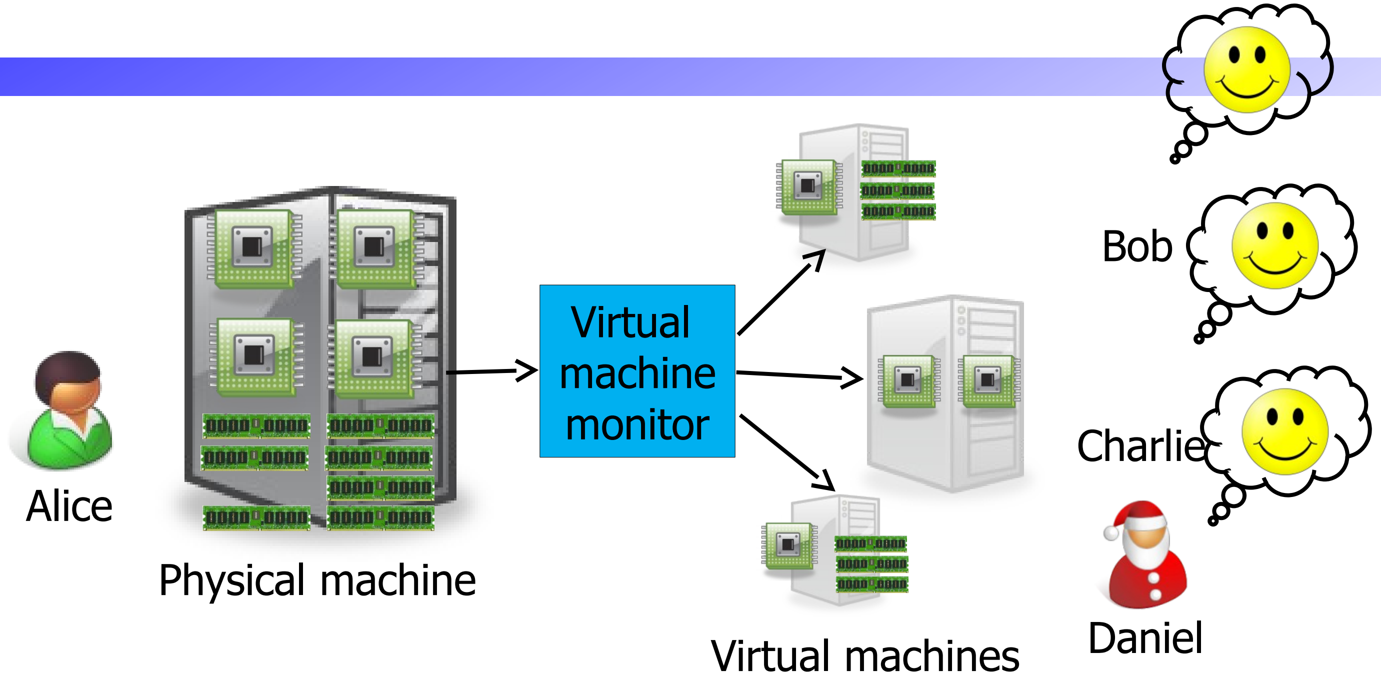
- Public cloud:
  - Commercial service; open to (almost) anyone. Example: Amazon AWS, Microsoft Azure, Google App Engine
- Community cloud:
  - Shared by several similar organizations. Example: government clouds
- Private cloud:
  - Shared within a single organization. Example: Internal datacenter of a large company.

What is virtualization?



- 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?

What is virtualization?

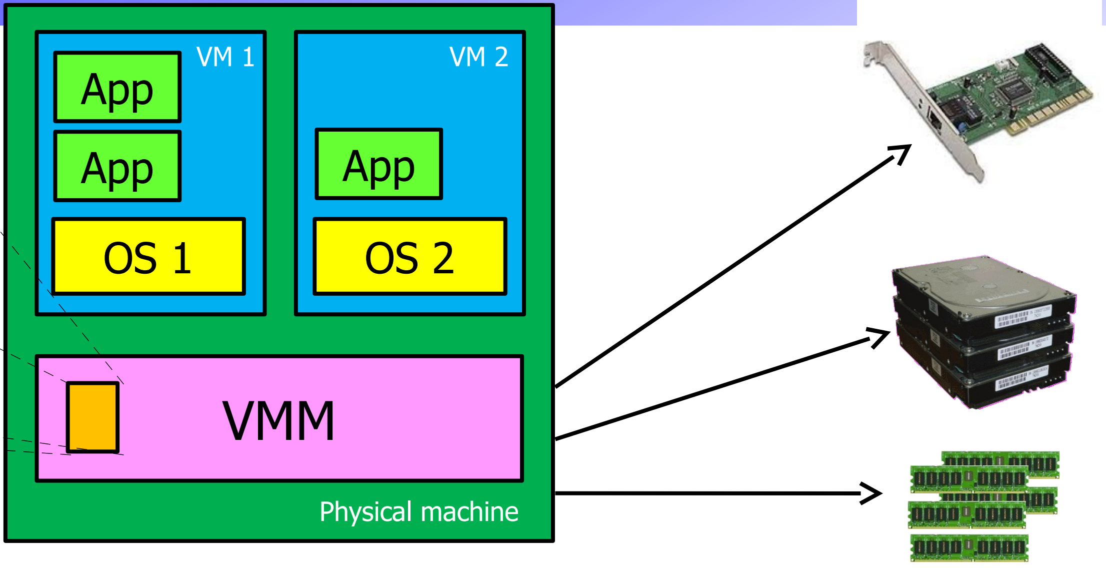


- Alice can sell each customer a virtual machine (VM) with the requested resources
- From each customer's perspective, it appears as if they had a physical machine all by themselves (isolation)

How does it work?

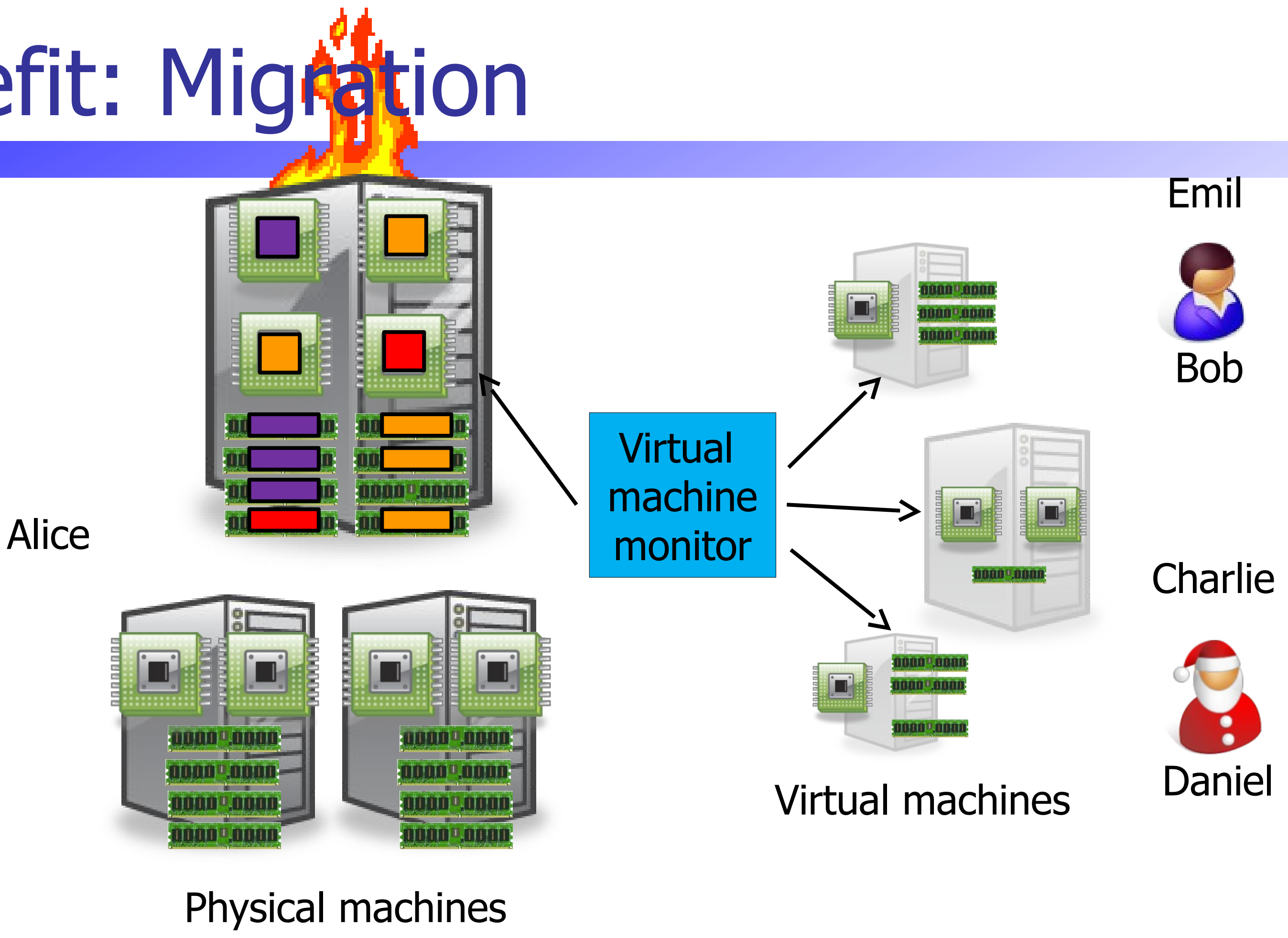
VM	Virt	Phys
1	0-99	0-99
1	299-399	100-199
2	0-99	300-399
2	200-299	500-599
2	600-699	400-499

Translation table



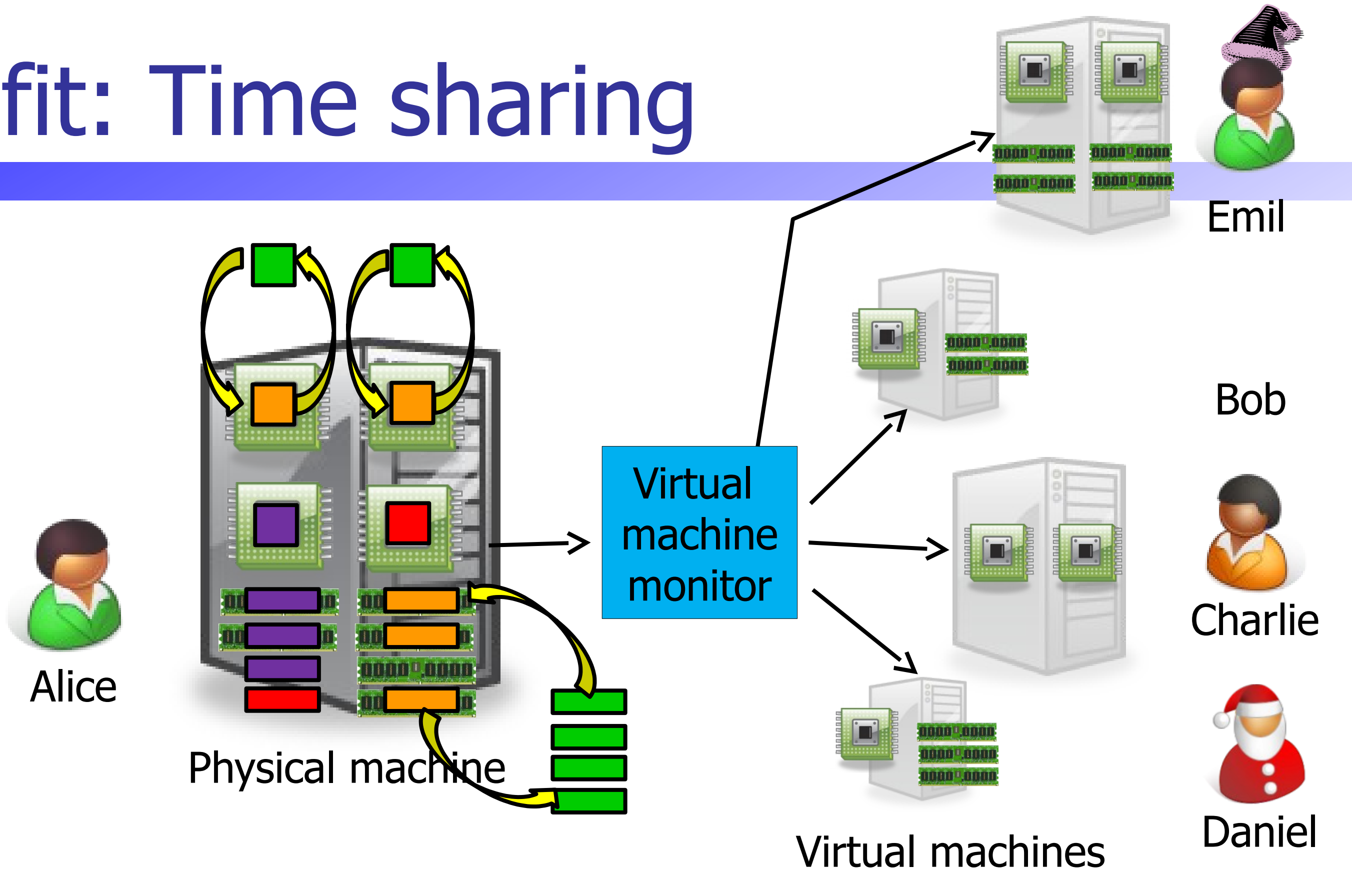
- Resources (CPU, memory, ...) are virtualized
  - VMM ("Hypervisor") has translation tables that map requests for virtual resources to physical resources
  - Example: VM 1 accesses memory cell #323; VMM maps this to memory cell 123.
  - For which resources does this (not) work?
  - How do VMMs differ from OS kernels?

Benefit: Migration



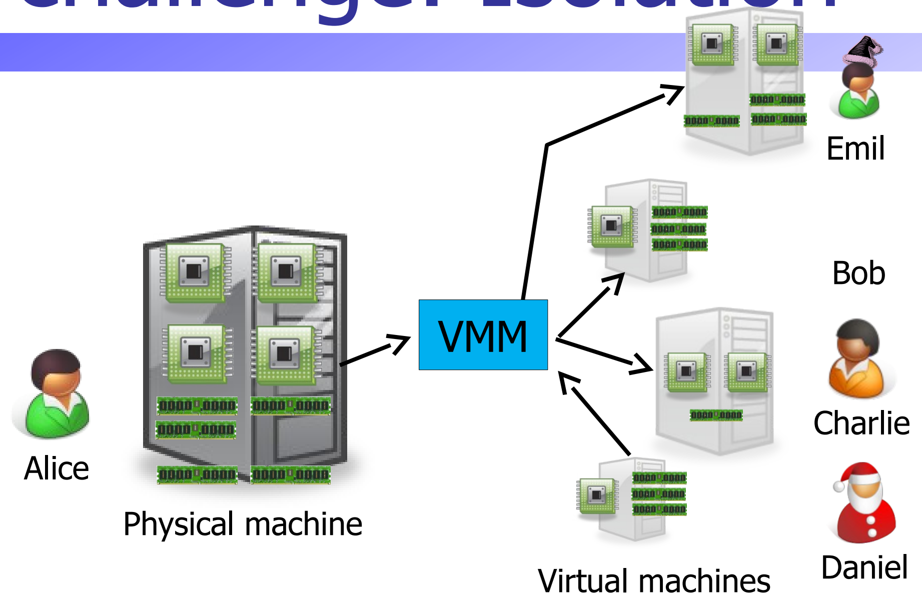
- What if the machine needs to be shut down?
  - e.g., for maintenance, consolidation, ...
  - Alice can migrate the VMs to different physical machines without any customers noticing

Benefit: Time sharing



- What if Alice gets another customer?
  - Multiple VMs can time-share the existing resources
  - Result: Alice has more virtual CPUs and virtual memory than physical resources (but not all can be active at the same time)

Benefit and challenge: Isolation



- Good: Emil can't access Charlie's data
- Bad: What if the load suddenly increases?
  - Example: Emil's VM shares CPUs with Charlie's VM, and Charlie suddenly starts a large compute job
  - Emil's performance may decrease as a result
  - VMM can move Emil's software to a different CPU, or migrate it to a different machine

Virtualization in the cloud

- Gives cloud provider a lot of flexibility
  - Can produce VMs with different capabilities
  - Can migrate VMs if necessary (e.g., for maintenance)
  - Can increase load by overcommitting resources
- Provides security and isolation
  - Programs in one VM cannot influence programs in another
- Convenient for users
  - Complete control over the virtual 'hardware' (can install own operating system own applications, ...)
- But: Performance may be hard to predict
  - Load changes in other VMs on the same physical machine may affect the performance seen by the customer

10 obstacles and opportunities

- Availability
  - What happens to my business if there is an outage in the cloud?
- Data lock-in
  - How do I move my data from one cloud to another?
- Data confidentiality and auditability
  - How do I make sure that the cloud doesn't leak my confidential data?
  - Can I comply with regulations?
-

- Data transfer bottlenecks
  - How do I copy large amounts of data from/to the cloud?
  - Example: 10 TB from UC Berkeley to Amazon in Seattle, WA
  - Motivated Import/Export feature on AWS
- Performance unpredictability
  - Example: VMs sharing the same disk -> I/O interference
  - Example: HPC tasks that require coordinated scheduling
-

- Scalable storage
  - Cloud model (short-term usage, no up-front cost, infinite capacity on demand) does not fit persistent storage well
- Bugs in large distributed systems
  - Many errors cannot be reproduced in smaller configs
- Scaling quickly
  - Problem: Boot time; idle power
  - Fine-grain accounting?

- Reputation sharing
  - One customer's bad behavior can affect the reputation of others using the same cloud
  - Example: Spam blacklisting, FBI raid after criminal activity
- Software licensing
  - What if licenses are for specific computers?
    - Example: Microsoft Windows
  - How to scale number of licenses up/down?
    - Need pay-as-you-go model as well