Chapter 26 Architectures for the Cloud

Cloud Computing - Definition

- **Cloud computing** is an information technology (IT) paradigm that enables ubiquitous access to shared pools of system resources and higher-level services that can be rapidly provisioned with minimal management effort, often over the Internet.

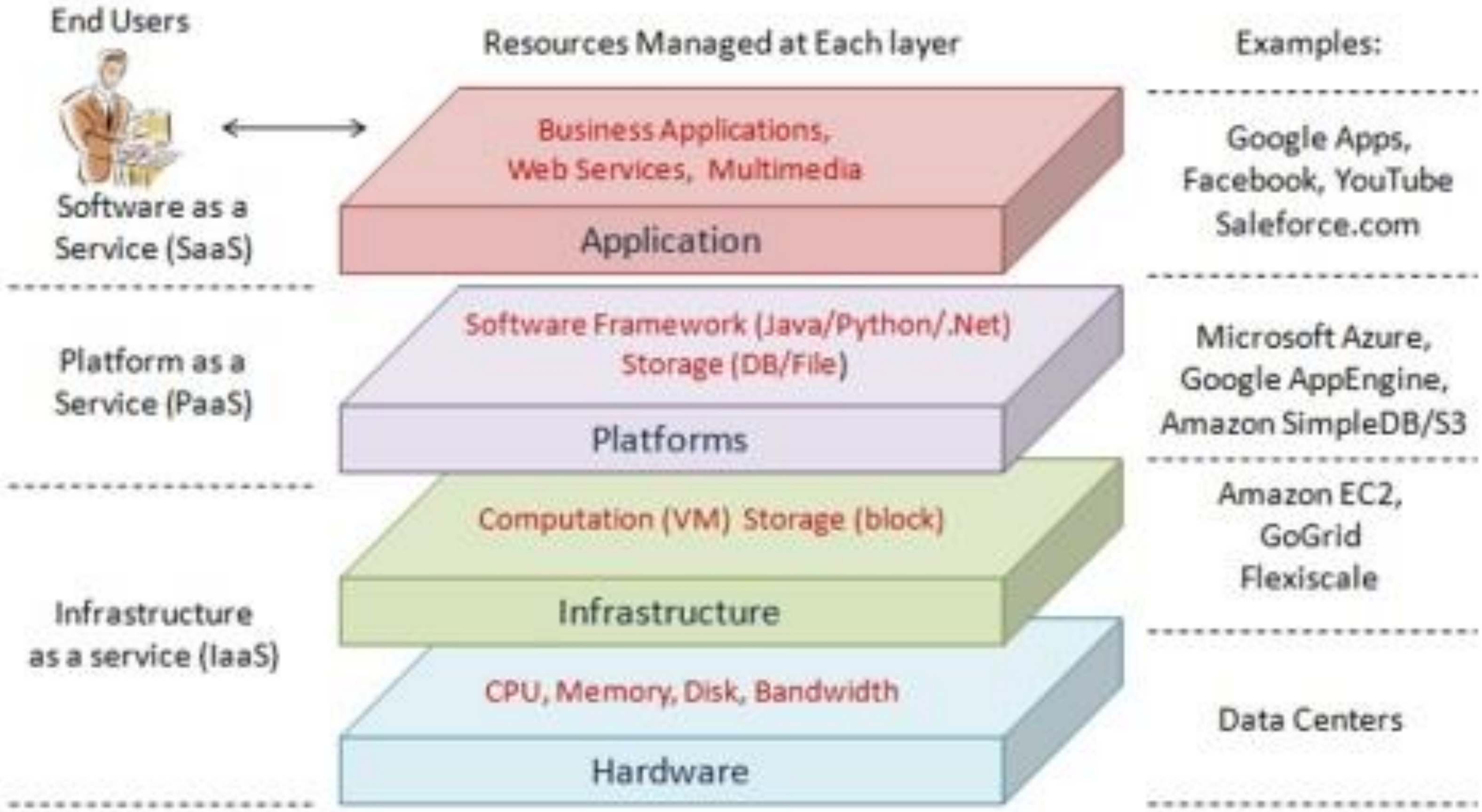
Basic Properties

- ***On-demand.*** A resource consumer can unilaterally acquire computing services as needed
- ***Resource pooling***. The cloud provider's computing resources are pooled.
- ***Ubiquitous network access***. Cloud services and resources are available over heterogeneous network access
- ***Location independence***. The location of the resources need not be of concern to the consumer of the resources.

Basic Properties

- ***Rapid elasticity***. Capabilities can be rapidly and elastically provisioned.
- ***Pay-as-you-go***. Consumers of the services are billed only for what they use.
- ***Multi-tenancy***. Applications and resources can be shared among multiple consumers who are unaware of each other.

Basic Service Models



End Users

Resources Managed at Each layer

Examples:

Software as a Service (SaaS)

Business Applications, Web Services, Multimedia

Application

Google Apps, Facebook, YouTube Saleforce.com

Platform as a Service (PaaS)

Software Framework (Java/Python/.Net) Storage (DB/File)

Platforms

Microsoft Azure, Google AppEngine, Amazon SimpleDB/S3

Infrastructure as a service (IaaS)

Computation (VM) Storage (block)

Infrastructure

Amazon EC2, GoGrid Flexiscale

CPU, Memory, Disk, Bandwidth

Hardware

Data Centers

Basic Service Models

- **Software as a Service (SaaS).**
- The consumer in this case is an end user.
- The consumer uses applications that happen to be running on a cloud.
- E.g. e-mail services

Basic Service Models

- **Platform as a Service (PaaS).**
- To provide the programming languages and tools for the users to develop and deploy applications on the cloud
- The consumer in this case is a developer.
- E.g., Google App Engine, Microsoft Azure,

Basic Service Models

- **Infrastructure as a Service (IaaS).**
- To provision processing, storage, networks, and other fundamental computing resources
- The consumer is able to deploy and run arbitrary software, which can include operating systems and applications
- The consumer in this case is a developer or system administrator.
- E.g., Amazon EC2

Deployment Models

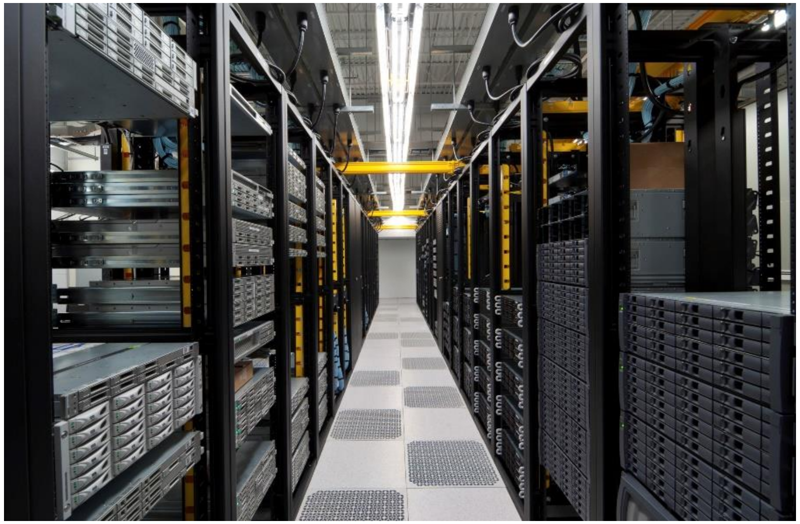
- ***Public cloud.*** The cloud infrastructure is made available to the general public and is owned by an organization selling cloud services.
- ***Private cloud.*** The cloud infrastructure is owned solely by a single organization and operated solely for applications owned by that organization.
- ***Community cloud.*** The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns
- ***Hybrid cloud.*** The cloud infrastructure is a composition of two or more clouds (private, community, or public)

Economic Justification

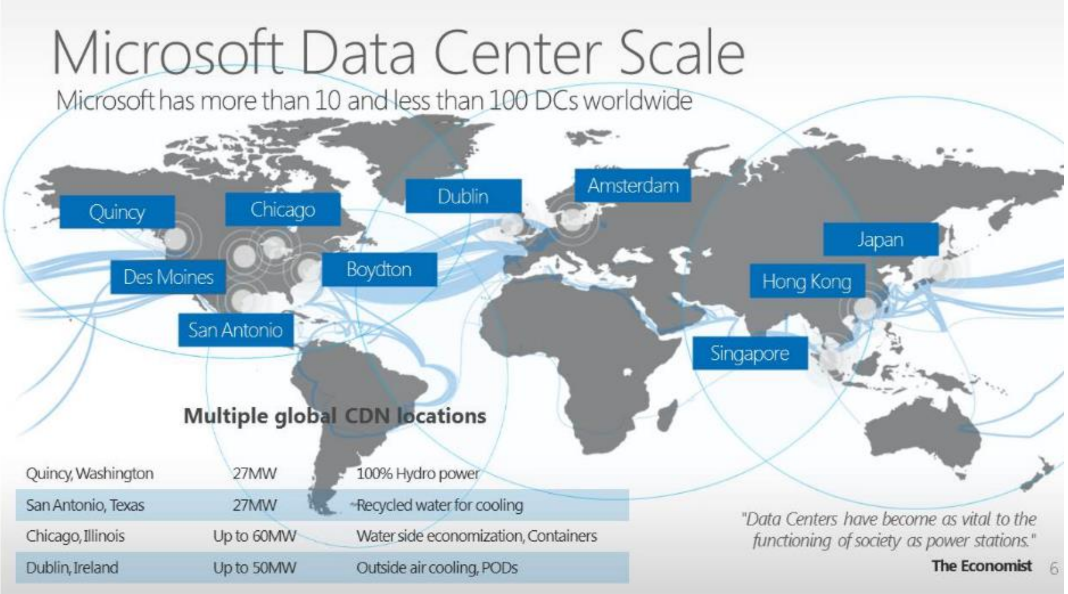
- Economies of scale
- Utilization of equipment
- Multi-tenancy

Economies of Scale

- Large data centers are cheaper to operate (per unit measure) than small data centers.
- ***Large*** in this context means 100,000+ servers
- ***Small*** in this context means <10,000 servers.



Geographically Distributed DCs



Microsoft Data Center Scale

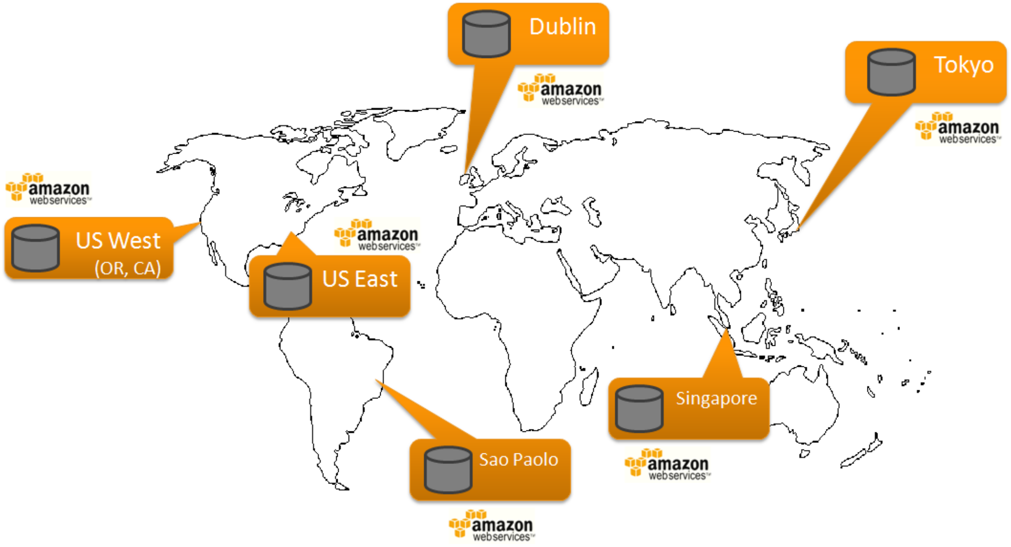
Microsoft has more than 10 and less than 100 DCs worldwide

Multiple global CDN locations

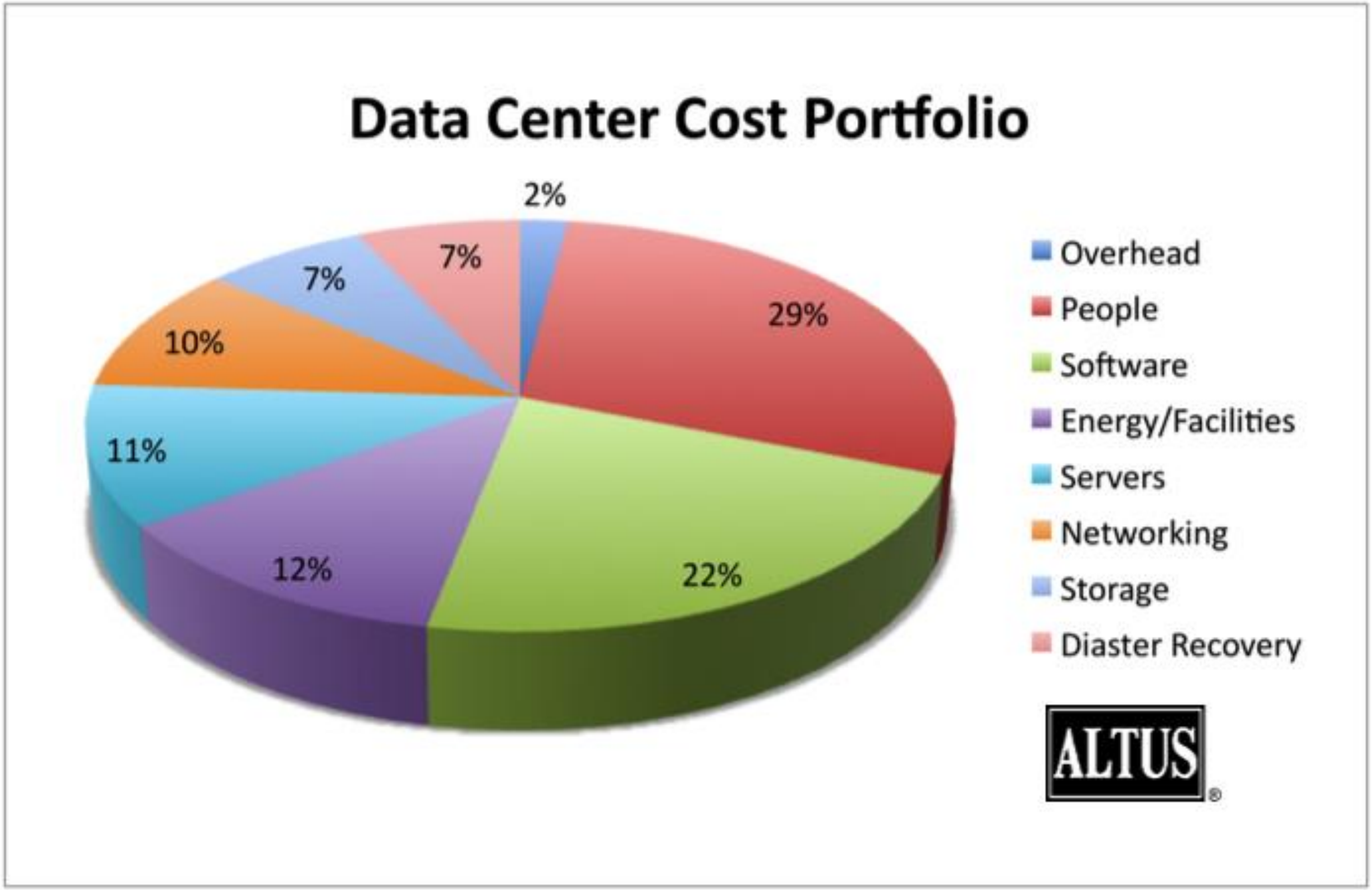
Quincy, Washington	27MW	100% Hydro power
San Antonio, Texas	27MW	Recycled water for cooling
Chicago, Illinois	Up to 60MW	Water side economization, Containers
Dublin, Ireland	Up to 50MW	Outside air cooling, PODs

"Data Centers have become as vital to the functioning of society as power stations."

The Economist



Data Center Cost



Source: Gartner (October 2009)

Reasons for Economies of Scale

- ***Cost of power***. The cost of electricity to operate a data center currently is 10 to 20 percent of the total cost of operation.
- Per-server power costs are lower in large data centers
  - Sharing of items such as racks and switches.
  - Negotiated prices. Large power users can negotiate significant discounts.
  - Geographic choice. Large data centers can be located where power costs are lowest.
  - Acquisition of cheaper power sources such as wind farms and rooftop solar energy.

Reasons for Economies of Scale

- ***Hardware costs***. Operators of large data centers can get discounts on hardware purchases of up to 30 percent over smaller buyers.
- ***Infrastructure labor costs***. More efficient utilization of system administrators
  - Small data center administrators service ~150 servers.
  - Large data center administrators service >1000 servers.

More Reasons for Economies of Scale

- ***Security and reliability***. Maintaining a given level of security, redundancy, and essentially disaster recovery requires a fixed level of investment.
- Larger data centers can amortize that investment over their larger number of servers.

Utilization of Equipment

- How to increase the utilization of data centers?
- **Virtualization** allows for co-location of distinct application
- Take use of **variations in workload** to increase utilization.
  - ***Random access***. End users access applications randomly. More users are more likely to impose a uniform load
  - ***Time of day.***
    - Co-locate those services that are workplace related with those that are consumer related.
    - Time differences among geographically distinct locations.

Utilization of Equipment

- ***Resource usage patterns***. Co-locate heavier CPU services with heavier I/O services
- ***Uncertainty***. Consider spikes in usage
  - news events, marketing events, sporting events
  - Leverage public cloud to maintain sufficient capacity to support spikes in usage

The key technology is to analyze the load variance pattern, and leverage the pattern to allocate the load on the resources

Multi-tenancy

- **Multi-tenancy** is a software architecture in which a single instance of software runs on a server and serves multiple tenants
- In **multi-instance architectures**, separate software instances operate on behalf of different tenants
- Take advantage of multi-tenancy in SaaS
- This reduces costs
  - Upgrade once, simultaneously, for all consumers
  - Single version of the software from a development and maintenance perspective.

Basic Mechanisms

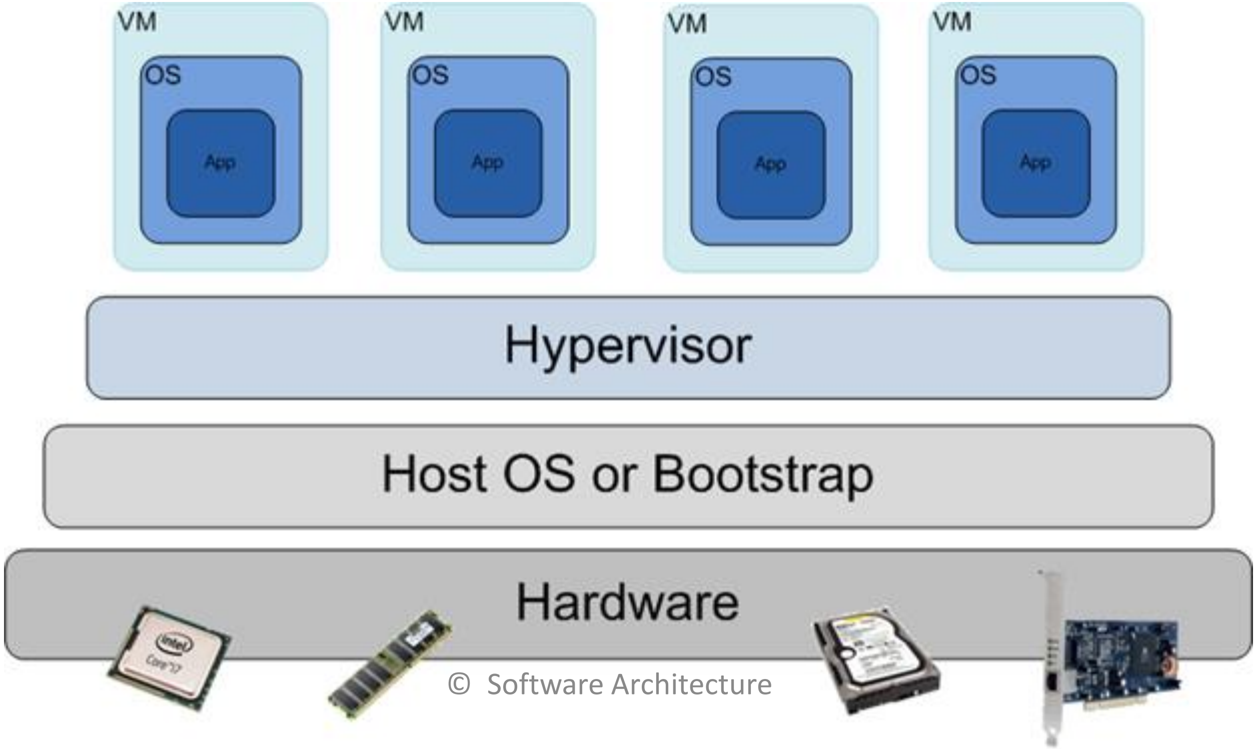
- Virtual Machine
- Hypervisor
- File system

Virtual Machine

- A **virtual machine** has an address space isolated from any other virtual machine.
- Looks like a bare metal machine from the application perspective.
- Assigned an IP address and has network capability.
- Can be loaded with any operating system or applications that can execute on the processor of the host machine.

Hypervisor

- A **hypervisor** is the operating system used to create and manage virtual machine
- E.g., VMWare, Xen, KVM



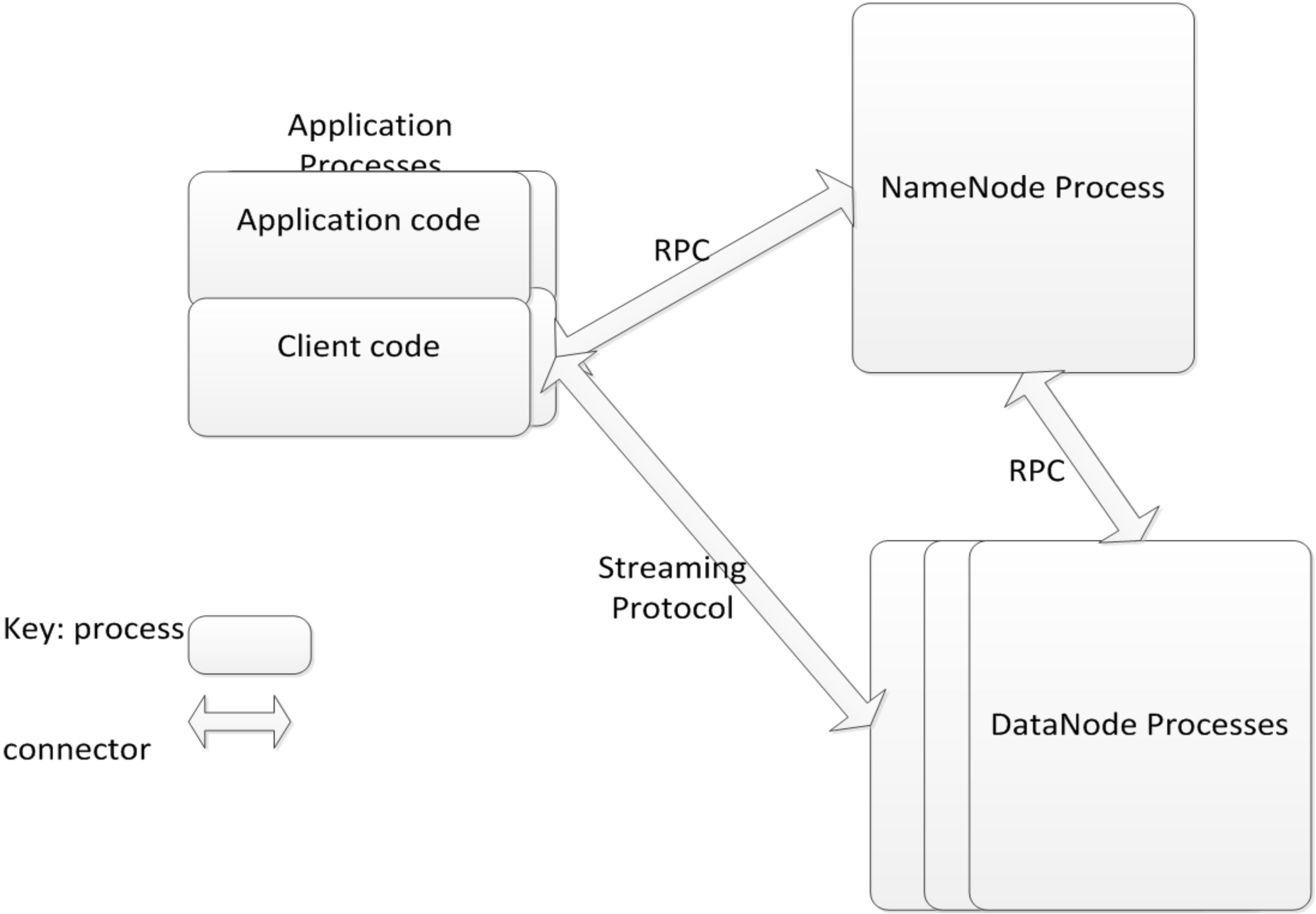
Techniques on Virtual Machine

- **Virtual machine consolidation**
  - How to place the VMs onto the physical machine
- **Virtual machine migration**
  - Happen when machine failure occurs
  - To save energy
  - For load balancing among the physical machine
- **Virtual machine startup storm**
  - How to avoid it?

File System

- Each virtual machine has access to a file system.
- We will present HDFS (Hadoop Distributed File System) – a widely used open source cloud file system.
- We describe how HDFS uses redundancy to ensure availability.

HDFS Components



HDFS Write – Sunny Day Scenario

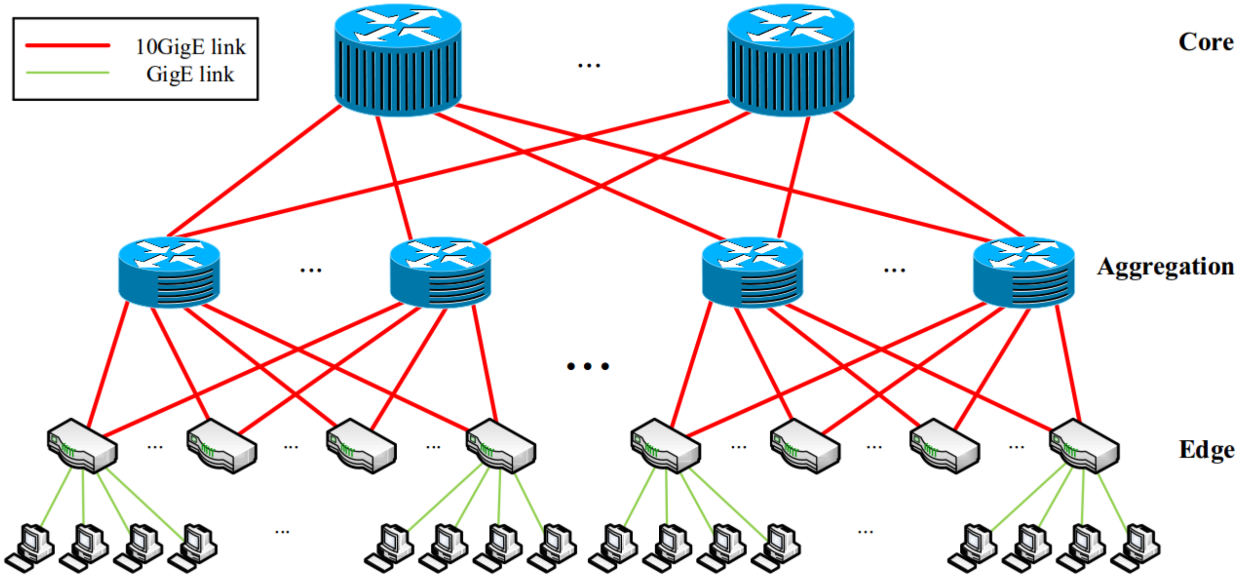
- Application writes as to any file system
- Client buffers until it gets 64K block
- Client informs *NameNode* it wishes to write a new block
- *NameNode* returns list of three *DataNodes* to hold block
- Client sends block to first *DataNode* and informs *DataNode* of other two replicas.
- First *DataNode* writes block and sends it to second *DataNode*. Second *DataNode* writes block and sends it to last *DataNode*.
- Each *DataNode* reports to client when it has completed its write
- Client commits write to *NameNode* when it has heard from all three *DataNodes*.

HDFS Write – Failure Cases

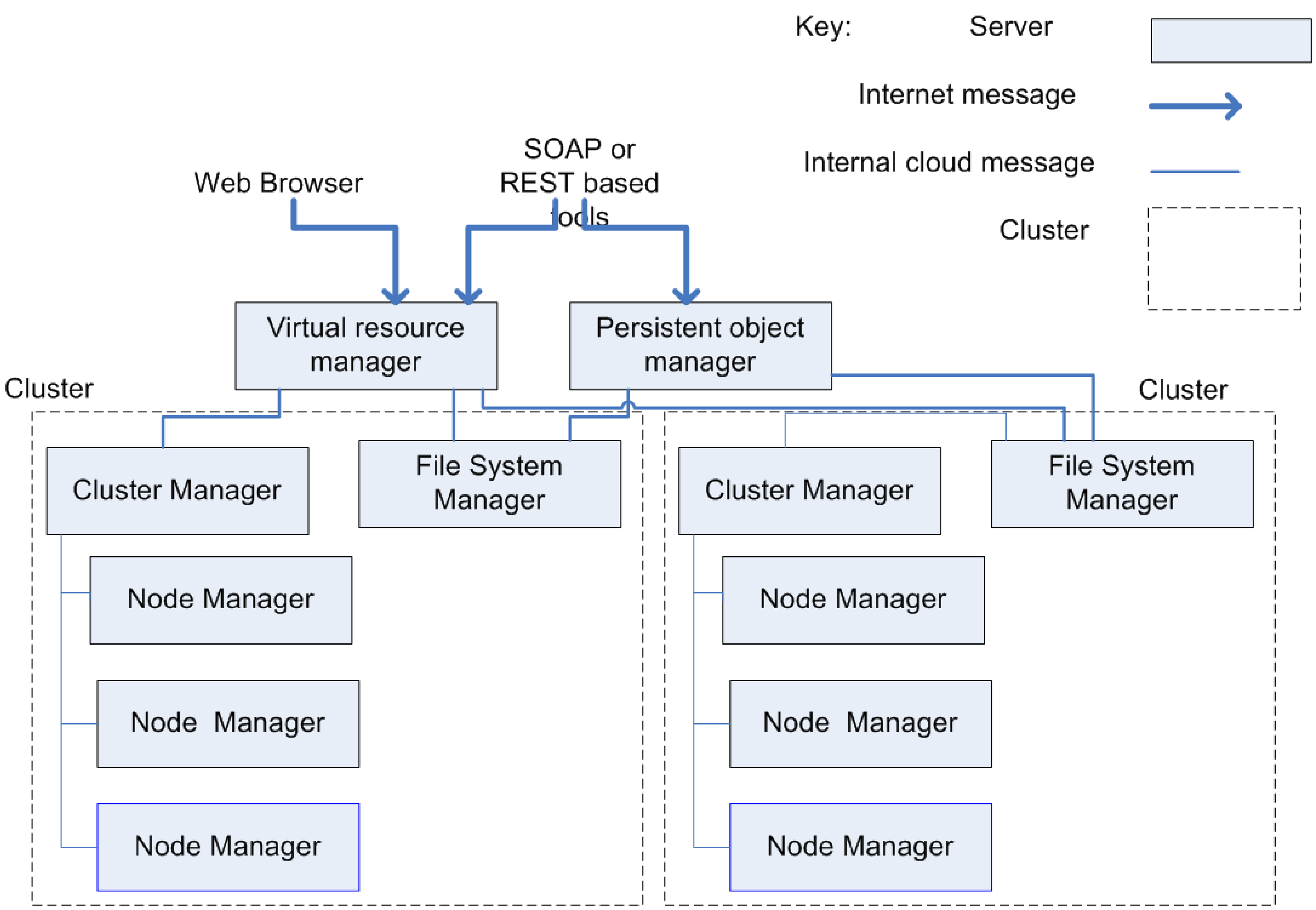
- Client fails
  - Application detects and retries
  - Write is not complete until committed by Client
- NameNode fails
  - Backup NameNode takes over
  - DataNodes maintain true list of which blocks they have
  - Client detects and retries
- DataNode fails
  - Client (or earlier DataNode in pipeline) detects and asks NameNode for different DataNode.
- Since each block is replicated three times, a failure in a DataNode does not lose any data.

Data Replication

- Trade-off between the number of replicas and availability
- How to place the replicas onto the machines to maximize the availability?



IaaS Architecture



Key: Server

Internet message

Internal cloud message

Cluster

Web Browser

SOAP or REST based tools

Virtual resource manager

Persistent object manager

Cluster

Cluster

Cluster Manager

File System Manager

Node Manager

Node Manager

Node Manager

Cluster Manager

File System Manager

Node Manager

Node Manager

Node Manager

IaaS Architecture Components

- **Cluster Manager** responsible for managing each cluster
- **Persistent Object Manager** manages persistent storage
- **Virtual Resource Manager** manages virtual machines. It acts as a gateway for messages.
- **The File System Manager** is similar to HDFS. It manages the network file system.

Services Provided by IaaS

- Automatic reallocation of IP addresses in the case of a failure of the underlying virtual machine instance.
- **Automatic Scaling**. Create or delete new virtual machines depending on load.

PaaS

- Provides an integrated stack for developer.
- E.g. LAMP stack
  - Linux, Apache, MySQL, Python
- The developer writes code in Python and the PaaS manages assignment to underlying layers of the stack.

Databases

- Why relational databases came into question
  - Massive amounts of data are collected from web systems. Much of this data is processed sequentially and so RDBMSs introduce overhead
  - The relational model is not the best model for some applications.
- Caused the introduction of new data models
  - Key-value
  - Document centric
- NoSQL: Hbase, MongoDB

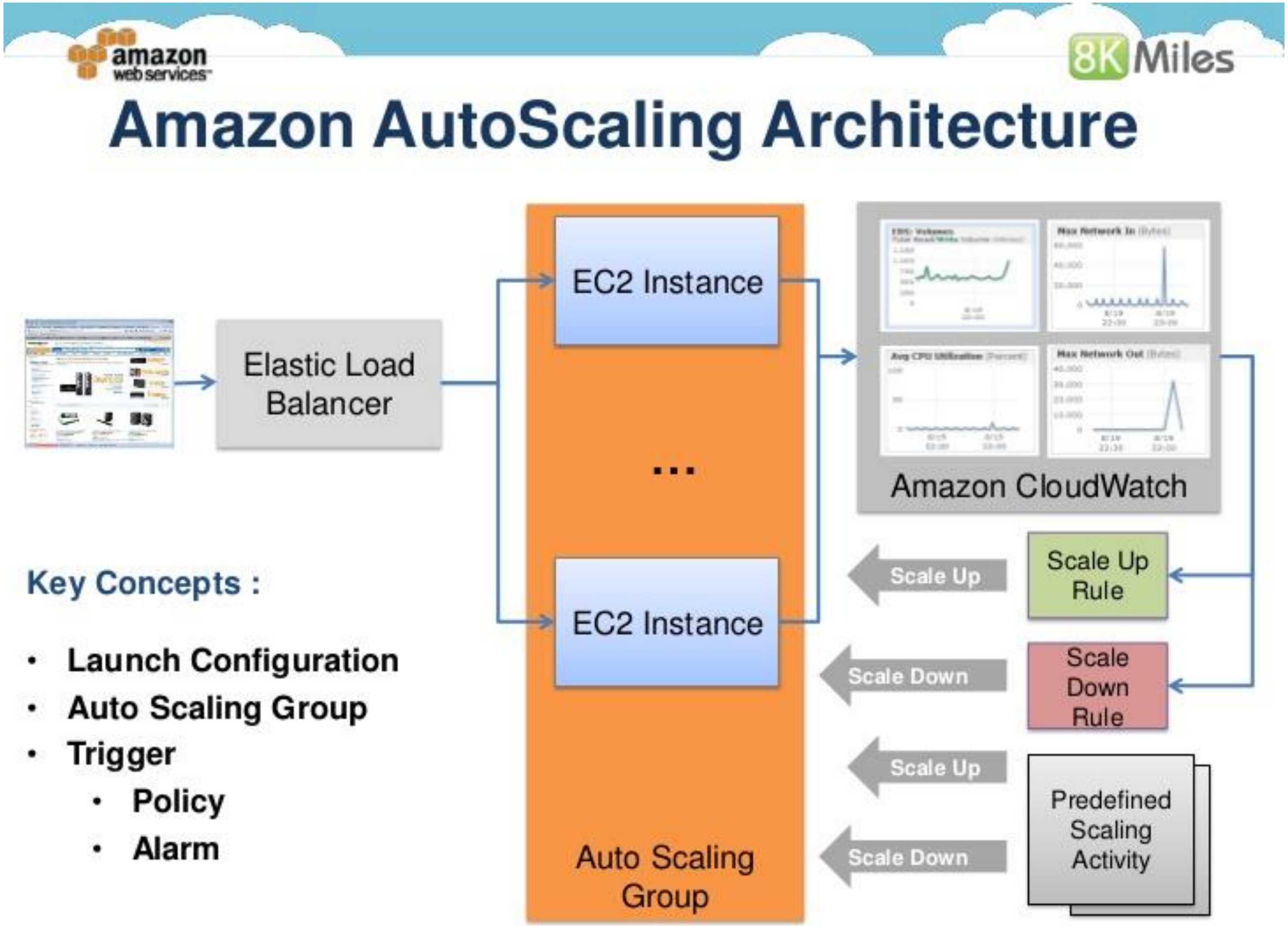
Architecting in a Cloud Environment

- Quality attributes that are different in a cloud
  - Security
  - Performance
  - Availability

Performance

- Two approaches to guarantee performance
  - **Load balancing is to** distribute workloads across multiple computing resources to avoid the overload of a single resource
  - **Auto-scaling** is a method whereby the amount of computational resources, typically measured in terms of the number of active servers, scales automatically based on the load

Auto-scaling Architecture in Amazon

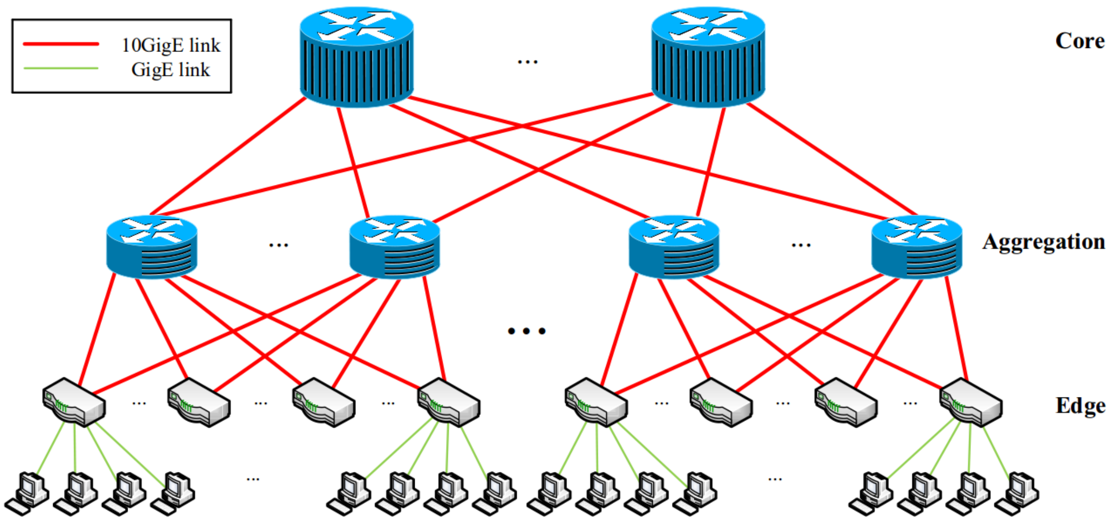


Availability

- **Failure** is a common occurrence in the cloud
  - With 1000s of servers, failure is to be expected
- Cloud providers ensure that the cloud itself will remain available with some notable exceptions.
- Application developers must assume instances will fail and build in detection and correction mechanisms in case of failure.

Strategies

- Make use of redundancy to deploy two of everything: two servers, two load balancers, two switches, two firewalls …



- Replication

Security

- Multi-tenancy introduces additional concerns over non-cloud environments.
  - **Inadvertent information sharing**. Possible that information may be shared because of shared use of resources. E.g. information on a disk may remain if the disk is reallocated.
  - **A virtual machine escape** is the process of breaking out of a virtual machine (hypervisor) and interacting with the host operating system
  - **Denial of Service attacks**. One users can consume resources of host server and deny them to other users.

Summary

- The cloud provides a new platform for applications with some different characteristics.
- Architect needs to know how a cloud cluster works and pay special attention to
  - Security
  - Performance
  - Availability