Cloud Computing

Salvatore Filippone, PhD

School of Aerospace, Transport and Manufacturing
salvatore.filippone@cranfield.ac.uk

Introduction

Cloud Essentials: What is it?

Anatomy of the Cloud: How does it work?

Opportunities: What can we do with it?

Introduction

In order to be called a cloud, your infrastructure must have five essential characteristics:

1. On-demand service *Consumer, Resources, Automatically*
2. Broad network access *Resources, all platforms*
3. Resource pooling *Multitenancy, Virtualization, Abstraction, Independence*
4. Rapid elasticity *up, down, in, out, auto*
5. Measured service, or pay-per-use model *resources monitored, controlled, reported, transparent*

Cloud Essentials

What is Cloud?

- A distinct IT environment
- Remotely provisioning scalable and measured IT resources

Cloud vs Internet:

- A cloud has a finite boundary
- Many clouds are accessible via the Internet
- privately owned vs open access
- Cloud may not be Internet based
- Can be based on the use of any protocols that allow for the remote access to its IT resources (ex. LANs or private, where the cloud is deployed inside a firewall and managed by the user organization)

Cloud Essentials

Definition

A computing Cloud is a set of network-enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing infrastructures on demand, which could be accessed in a simple and pervasive way.

Wang et al, "Cloud Computing: a Perspective Study", New Generation Computing, 28(2), 137-146, 2010

Cloud Essentials

Definition

A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet

I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," 2008 Grid Computing Environments Workshop, 2008

Definitions

- Cloud Provider
- Cloud Consumer
- Cloud Service Owner
- Cloud Resource Administrator
- Additional Roles
- Organizational Boundary
- Trust Boundary

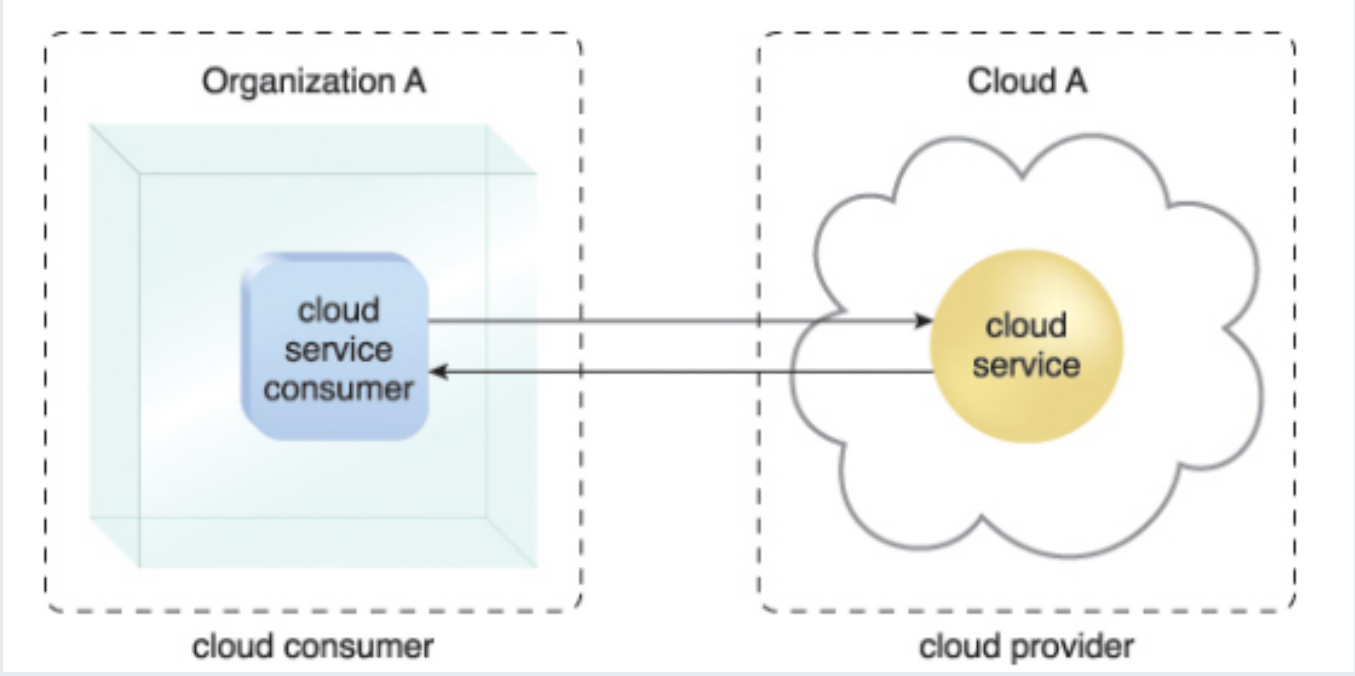
Cloud Essentials – Definitions

Cloud Provider

- Organization that provides cloud-based IT resources
- Responsible for making cloud services available to cloud consumers
- Tasked with any required management and administrative duties to ensure the on-going operation of the overall cloud infrastructure
- They normally own the IT resources that are made available for lease by cloud consumers
- Some cloud providers also "resell" IT resources leased from other cloud providers
- Amazon Web Services, Microsoft Azure, Google Drive, Vmware, Salesforce.com

Cloud Consumer

Organization (or a human) that has a formal contract or arrangement with a cloud provider to use IT resources made available by the cloud provider

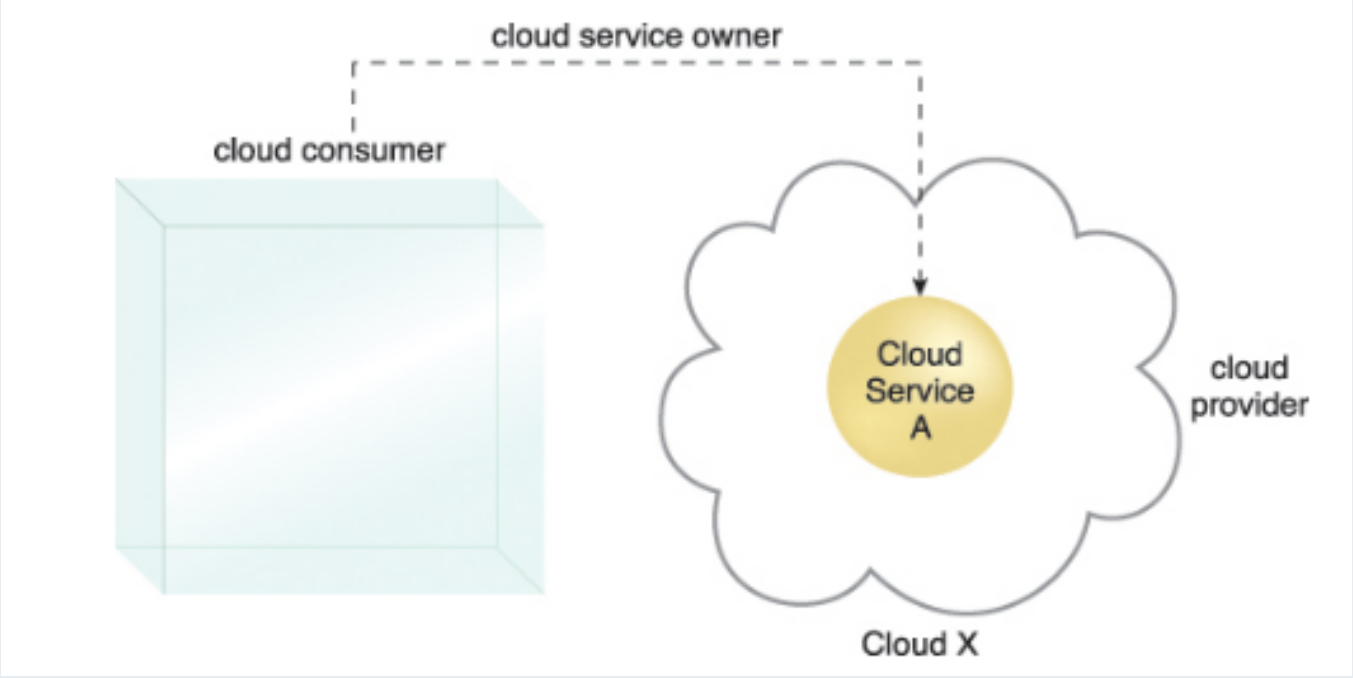


Erl Thomas

Cloud Essentials – Definitions

Cloud Service Owner

The person or organization that legally owns a cloud service. A cloud consumer can be a cloud service owner when it deploys its own service in a cloud. A cloud provider becomes a cloud service owner if it deploys its own cloud service, typically for other cloud consumers to use.



Erl Thomas

Cloud Resource Administrator

- Person/ organization responsible for administering a cloud-based IT resource
- Can be with a cloud consumer organization and administer remotely accessible IT resources that belong to the cloud consumer.
- OR can be with a cloud provider organization where can administer internally and externally available IT resources.

Additional Roles

Cloud Auditor A third-party

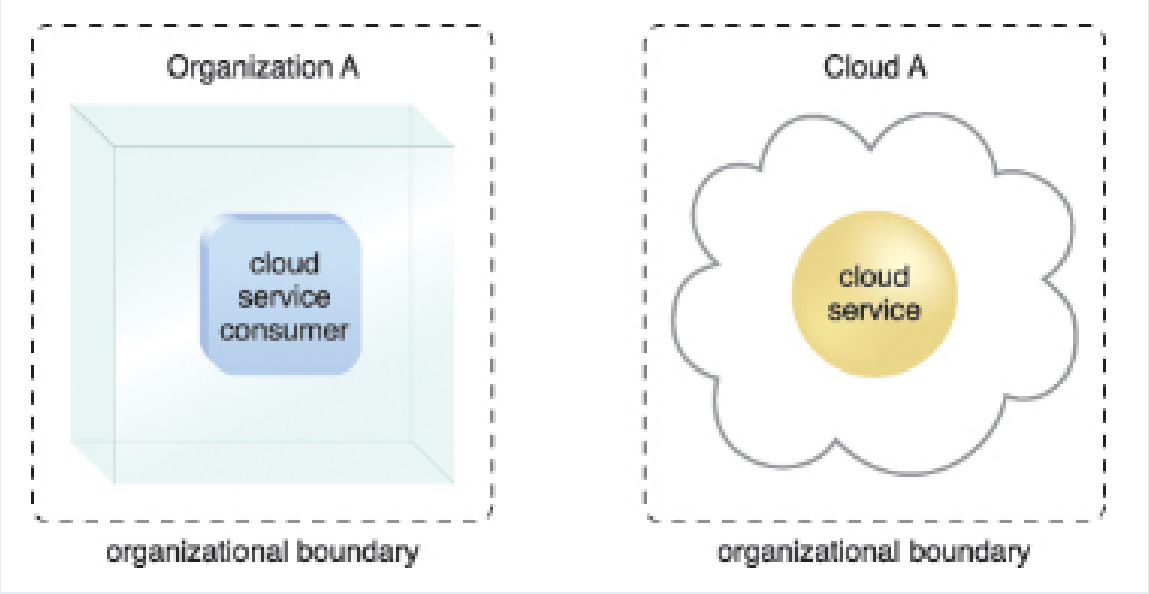
Cloud Broker A party with the responsibility of managing and negotiating the usage of cloud services between consumers and providers

Cloud Carrier The party responsible for providing the wire-level connectivity between consumers and providers

Erl Thomas

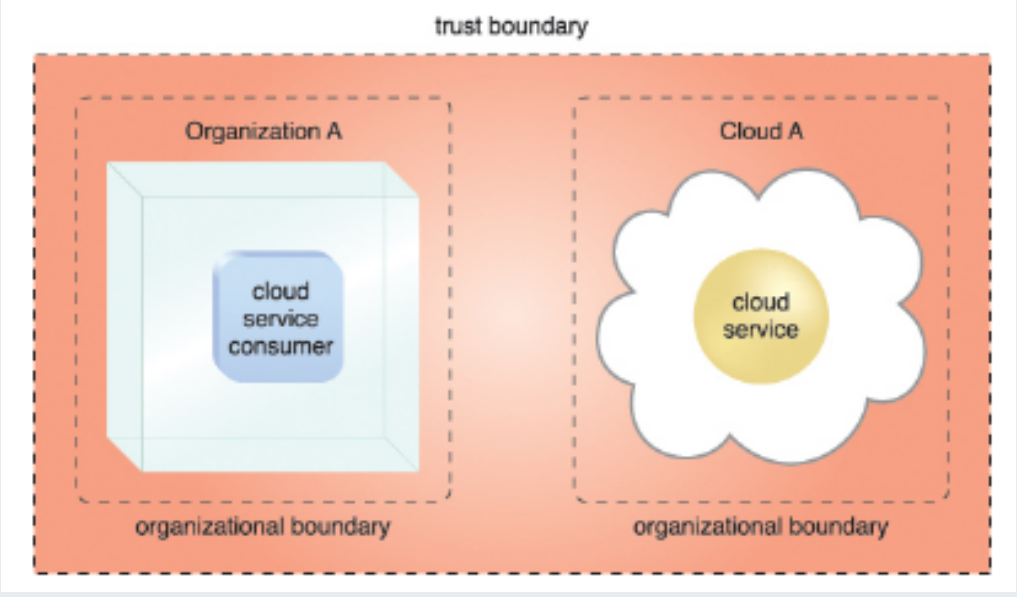
Organizational boundary

- Represents the physical perimeter that surrounds a set of IT resources that are owned and governed by an organization
- Does not represent the boundary of an actual organization, only an organizational set of IT assets and IT resources
- Similarly, clouds have an organizational boundary

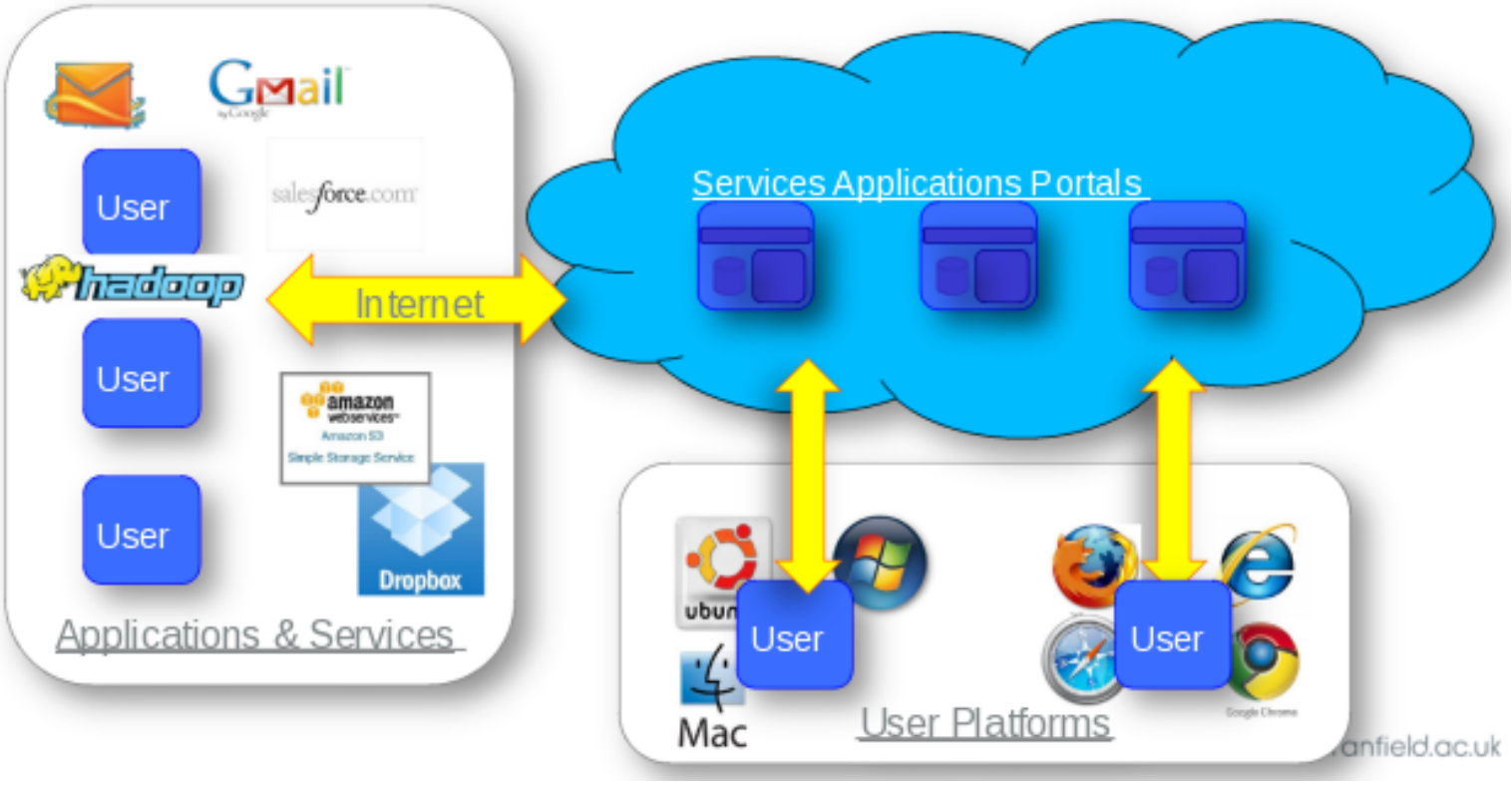


Trust boundary

- Consumer needs to extend its trust beyond the physical boundary of the organization to include parts of the cloud environment
- It is a logical perimeter — Spans beyond physical boundaries to represent the extent to which IT resources are trusted



What does a Cloud look like?



Motivations for Cloud Computing

- Capital Expenditure $\Rightarrow$ initial purchasing costs, rolling upgrade costs
- On-going maintenance $\Rightarrow$ labor and parts costs for existing local resources (departmental and enterprise wide)
- Environmental costs $\Rightarrow$ power, cooling
- Moving costs from capital to operational expenditure
- Access to large scale compute resources $\Rightarrow$ small engineering departments without significant compute resources to perform model simulations
- Rapid turn-around of results $\Rightarrow$ cost of 1000 servers for 1 hour is the same as 1 server for 1000 hours

Applications

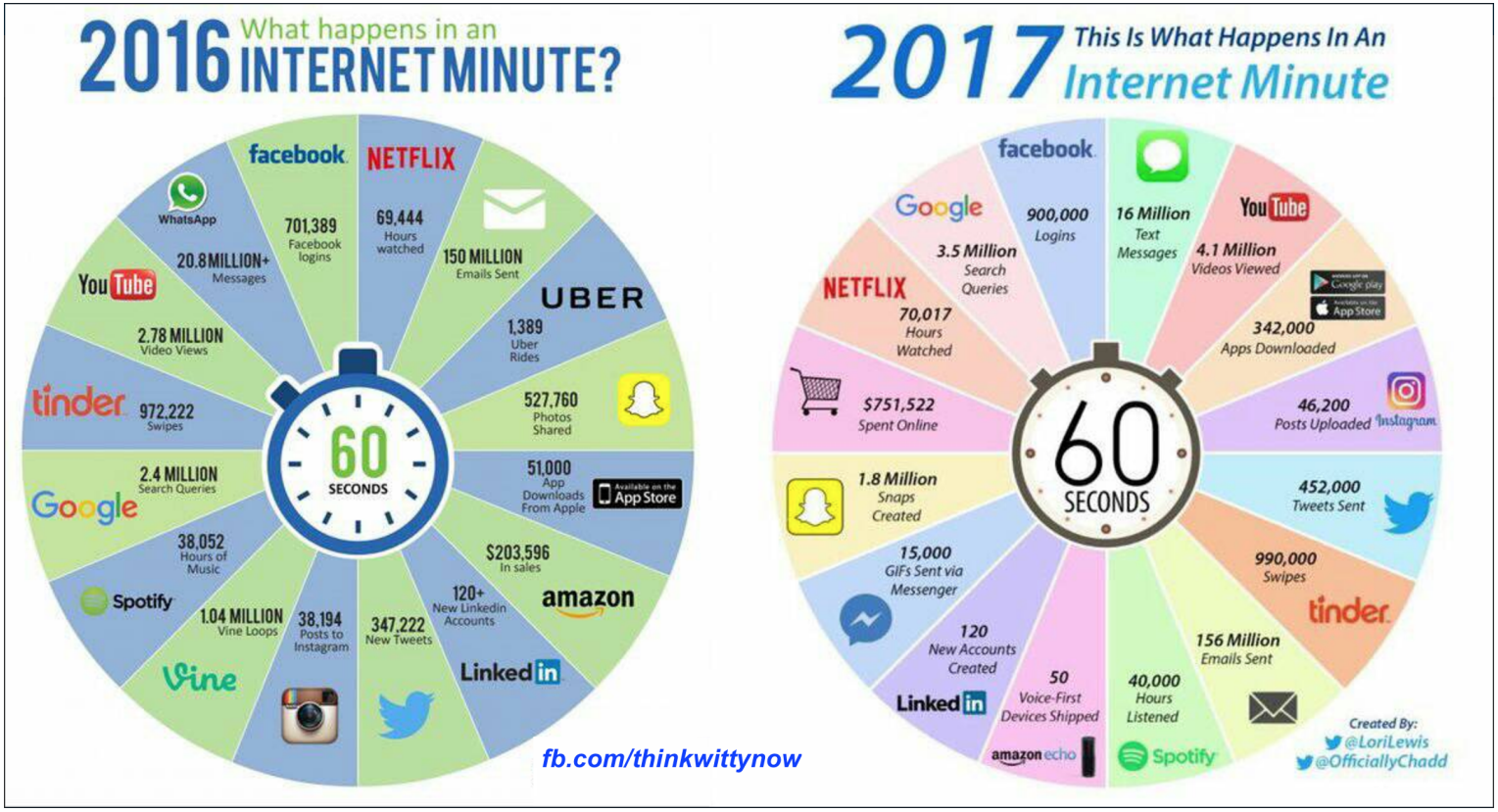
What can we do with Cloud Computing?

- Storage Backup; Work on Applications online
- Large scale data processing problems
- Web based applications
- Mobile applications — Google's Gmail and Google Voice for iPhone, Google Drive, Google Docs, Google Apps, Chrome
- Business analytics — IBM, SAP, SAS, Tableau
- Desktop application extensions
- Microsoft Live (Outlook.com, OneDrive, Windows Phone, or Xbox LIVE), Cloud Print (Xerox) , Drop Box, 4Shared, Picasa, Flickr, Etsy, Siri

Applications

Source: `http://www.visualcapitalist.com/internet-minute-2018/`

Applications



Source: `http://www.dalendesign.com/webpress-blog/social-media/happens-internet-minute/`

Cloud Essentials

What are the technical roots of Cloud Computing?

Grid Computing: Distributed, geographically displaced computing

Utility Computing: Payment on demand (cf. gas, water, electricity)

Service Oriented Architectures: componentized, reusable applications using common communication protocols to provide interfacing

Cluster/HPC computing

Cloud Essentials

Cluster

- Simple (min 2 computers)
- It is essentially smashing up many machines to make a really big and powerful one
- (Nearly) Identical computers; may run special operating system versions to keep them synchronized.
- 100 % of the resource may be given to a single problem.
- On-demand Self-service? NO
- Rapid Elasticity? NO
- Measured Service? Not always

Cloud

- Clouds are not conceptually one big machine
- You may *NOT* allocate the full capacity of your data centre to a single request
- On-demand Self-service? YES
- Rapid Elasticity? YES
- Measured Service? YES

Clusters

Table 1: Grid vs. HPC Cluster vs. Cloud

Category	Grid	HPC Cluster	Cloud
SLA Requirement	High	Very High	Low
Unit of work type	Repetitive	Very Repetitive – MPP	Not MPP – workflow repetitive
Data Dependency	Medium	High	High
I/O Dependency	Medium	High	Low
External Dependency	Medium	None	None
Integration type	Aware	Tight with the H/W	Loose coupling
Integration time	Long	Long and on-going	Short
Statefull?	Not desirable but possible	No!	No
Typical unit of work	Seconds to minutes	Vey short – low minutes max	Long – minutes to hours

source: `http://www.devx.com/architect/Article/45626`

Grid Computing

collection of technologies that allow the consumption of compute resources on demand

Key technology driver: standardization of protocols used to request extra compute resources

- Creation of utility style resources (water, electricity, gas)
- Large, federated systems were created
- Commercial interest was slow to appear
- Little interest in creating a commercial Grid
- Easier to offer local cluster resources on-demand

Grid vs. Clouds

Common factors between the two:

Scalability: system's ability to handle increasing amounts of work, or to improve performance. Both are scalable, resources are directed on demand, have fluctuating storage capacity depending on number of users amount of data.

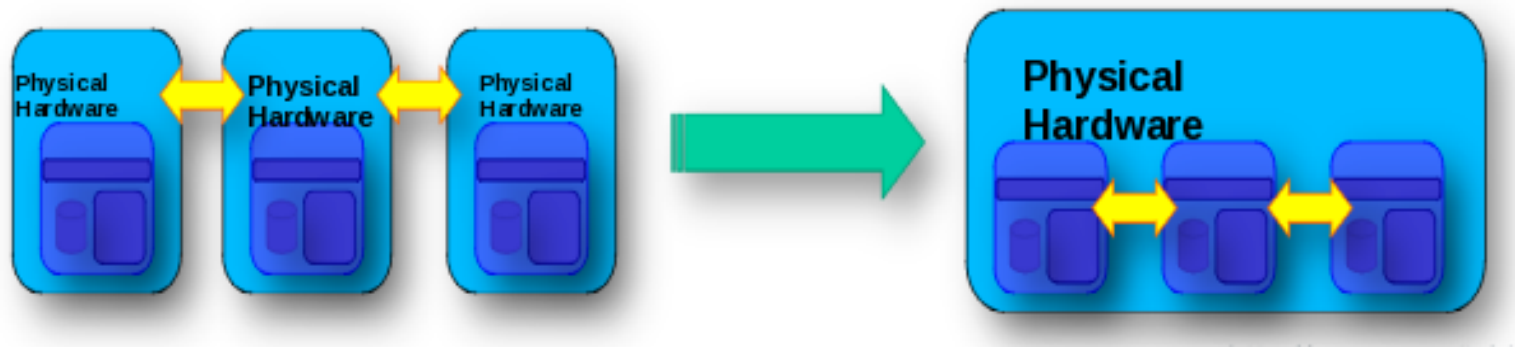
Multitenancy serve multiple clients

Multitasking many tasks share processing resources

Grid vs. Clouds

Clouds sit on top of virtualised resources, whilst Grid nodes sit on bare metal/clusters

- Advances in technologies allow more efficient and more flexible use of hardware — virtualisation
- Most advances in Grid computing came before large scale availability of virtualisation



Grid vs. Clouds

Role of middleware differs between Cloud and Grid

Grid middleware

- discovers resource and provides secure communications and usage (resource virtualization, resource management)
- Has complex interface
- Toolkits can be used to discover resources from many different sources

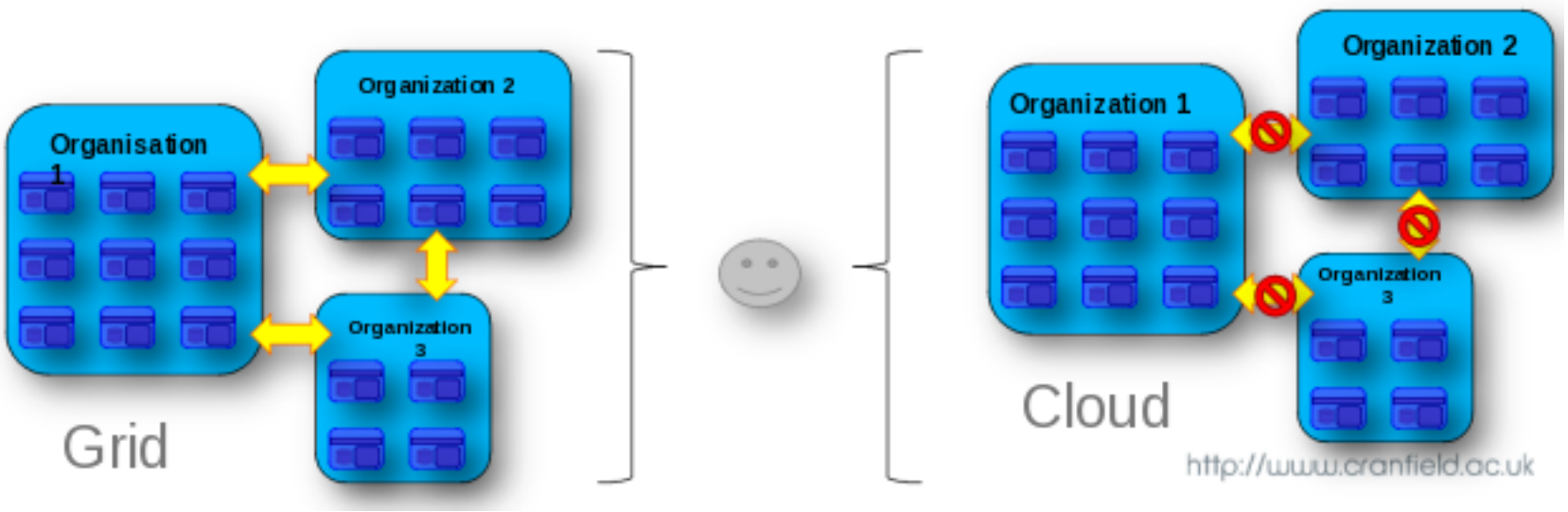
Cloud middleware

- Resource management solution by itself
- Provides an interface to manage resources
- Each vendor supplies its own API — no standards at the moment
- No need for resource discovery — assume infinite resources
- Security is usually implied — rely on the vendor to deliver as promised

Grid vs. Clouds

Clouds are usually offered by a single vendor, whilst Grid nodes can be offered by a number of vendors/sources

- Key technology driver was finding a mechanism to allow an application to execute on a large distributed resource
- Presently, Clouds do not interoperate very well



Grid vs. Clouds –The Concerns

Interoperability: How can vendors ensure interoperability for their clients? - it is necessary for decision-makers to examine how different cloud service vendors import or export data. Industry cloud computing standards do not exist for APIs or importing/exporting data.

Hidden Costs May include higher network charges for storage and database applications, or latency issues for users who may be located far from cloud service providers.

Surprises Experts recommend testing and pilot programs to ensure there will not be any unexpected outcomes. These may include tests checking application validation, allocating/releasing resources, etc.

Threshold Policy It is important to consider how the cloud service provider will handle sudden increases or decreases in demand. How will unused resources be allocated?

Grid vs. Clouds

Cloud infrastructure can be built into local enterprise and subsequently extended into 3rd party infrastructure

- Allows seamless scalability for end-user applications
- Ensures optimal utilization of local resources
- Cloudbursting

Some notable commercial/open source examples:

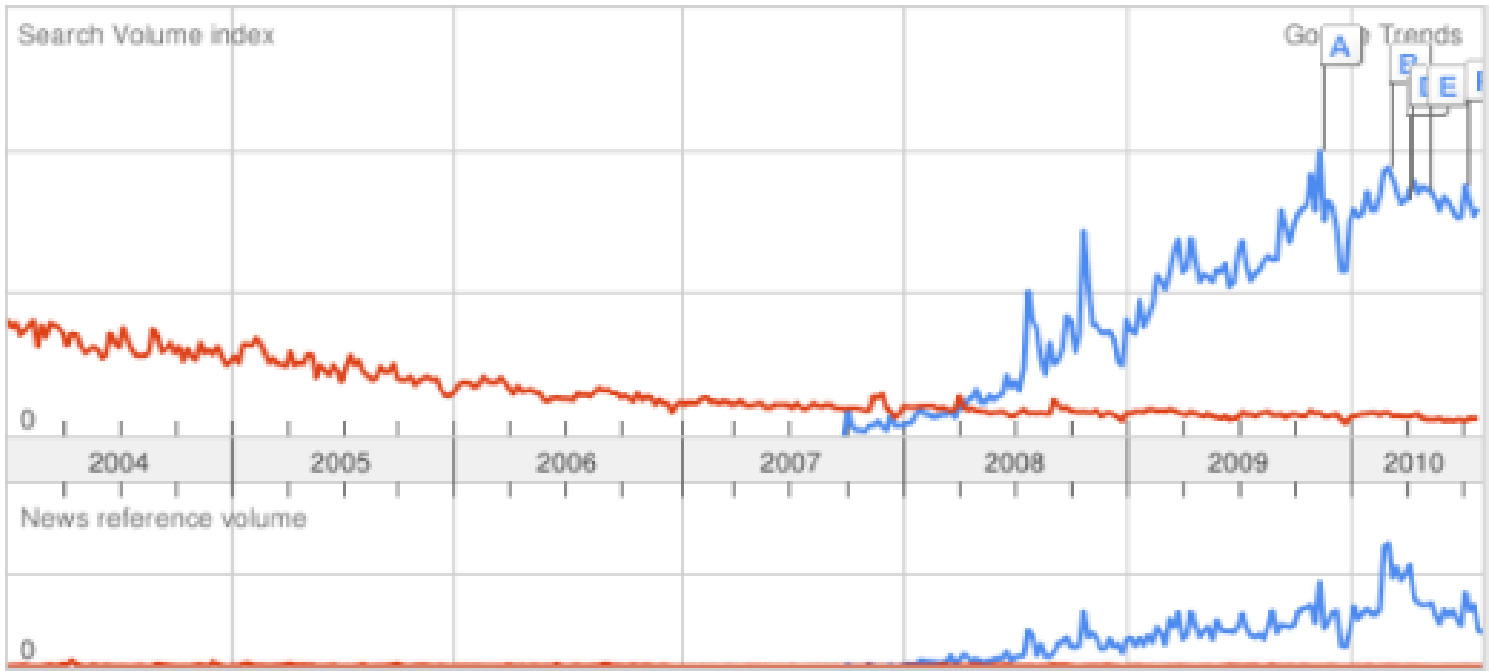
- Eucalyptus
- OpenStack, OpenCloud
- Amazon AWS
- Microsoft Azure

How do we manage large compute resources?

- What is the most effective way for potential consumers of the service to interact with these resources?
- How can we implement applications to best make use of these distributed resources?
- How to define methods to discover, request and use resources
- How to implement highly parallel computations
- Resource distribution: CC, a centralized model ; GC, a decentralized model where the computation could occur over many administrative domains.
- Ownership: A grid is a collection of computers which is owned by multiple parties in multiple locations and connected together so that users can share the combined power of resources. Whereas a cloud is a collection of computers usually owned by a single party.

Grid vs. Clouds

Rising interest in Cloud Computing: Cloud; Grid Starting at the end of 2007, convergence of grid and services into the cloud; the trend in terms of news reference volume remained constant in subsequent years



Differences between Clouds and other general environments

- User centric interfaces
- On-demand service provisioning
- QoS guaranteed offer
- Autonomous system — managed transparently to the user
- Scalability and Flexibility

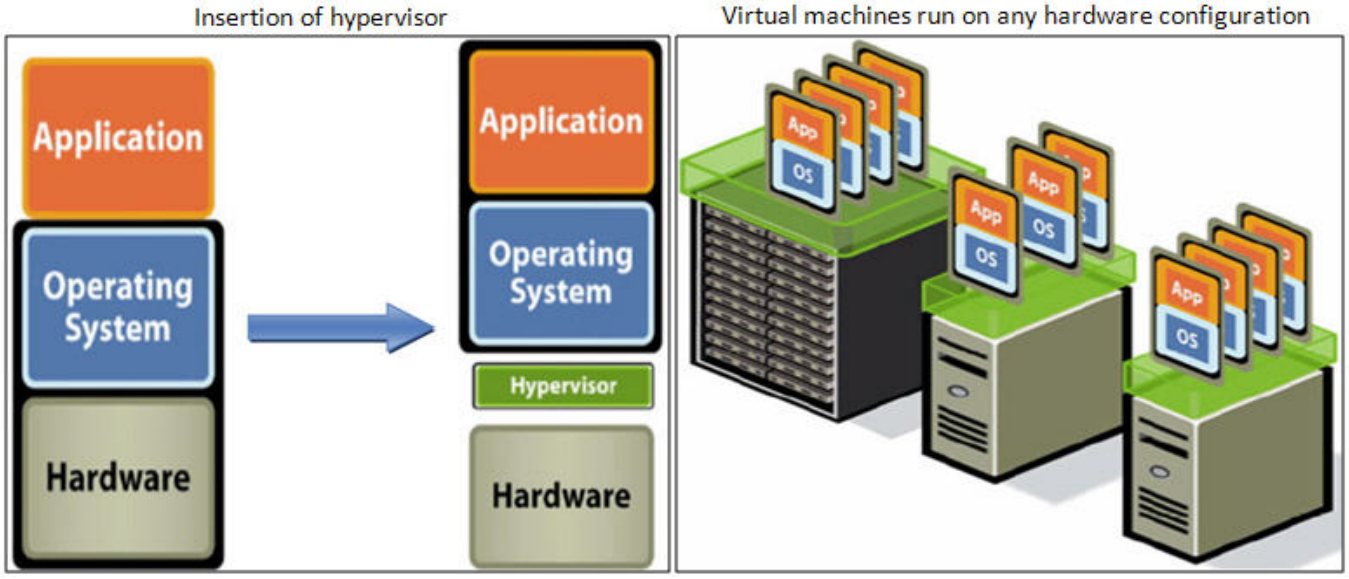
Cloud Summary

To summarise Cloud characteristics:

- Distributed compute resources
- Virtualised resources
- Resource Elasticity
- Pay As You Go pricing model
- Proprietary access mechanisms

Virtualisation

Virtualisation



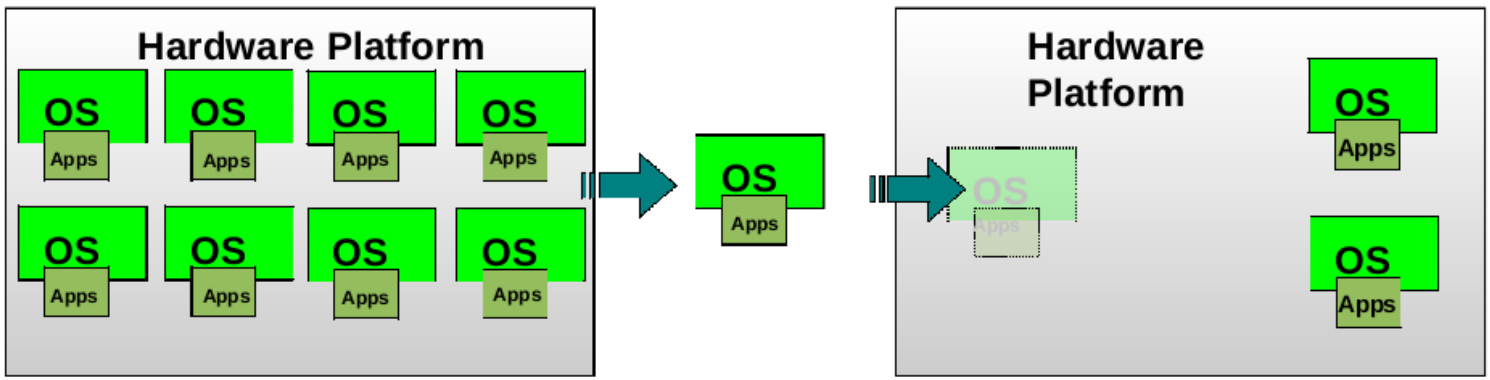
- Allow multiple operating systems and applications to run on a single unit of hardware
- Reduce energy requirements by using a single server to host several environments

Image source: dolcera.com

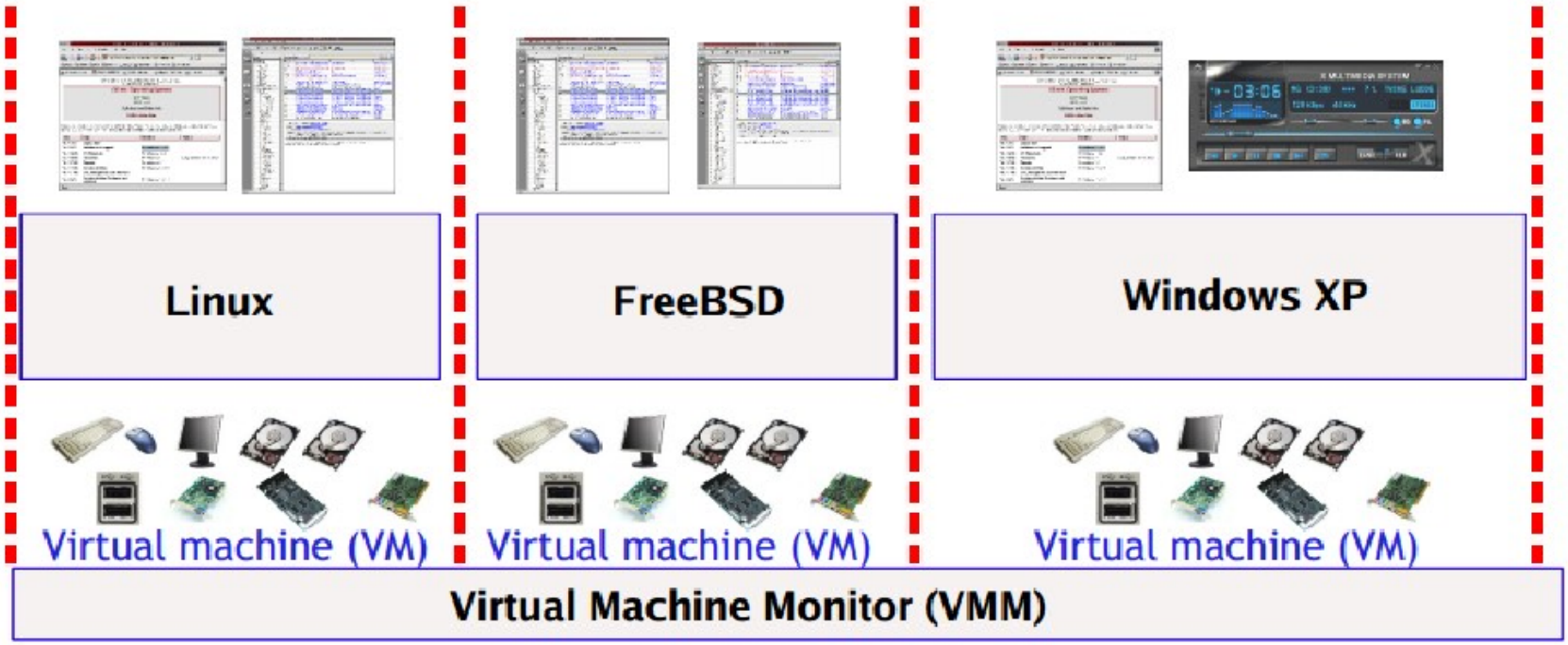
Virtualisation

Move operating systems from an overloaded server to an underused one

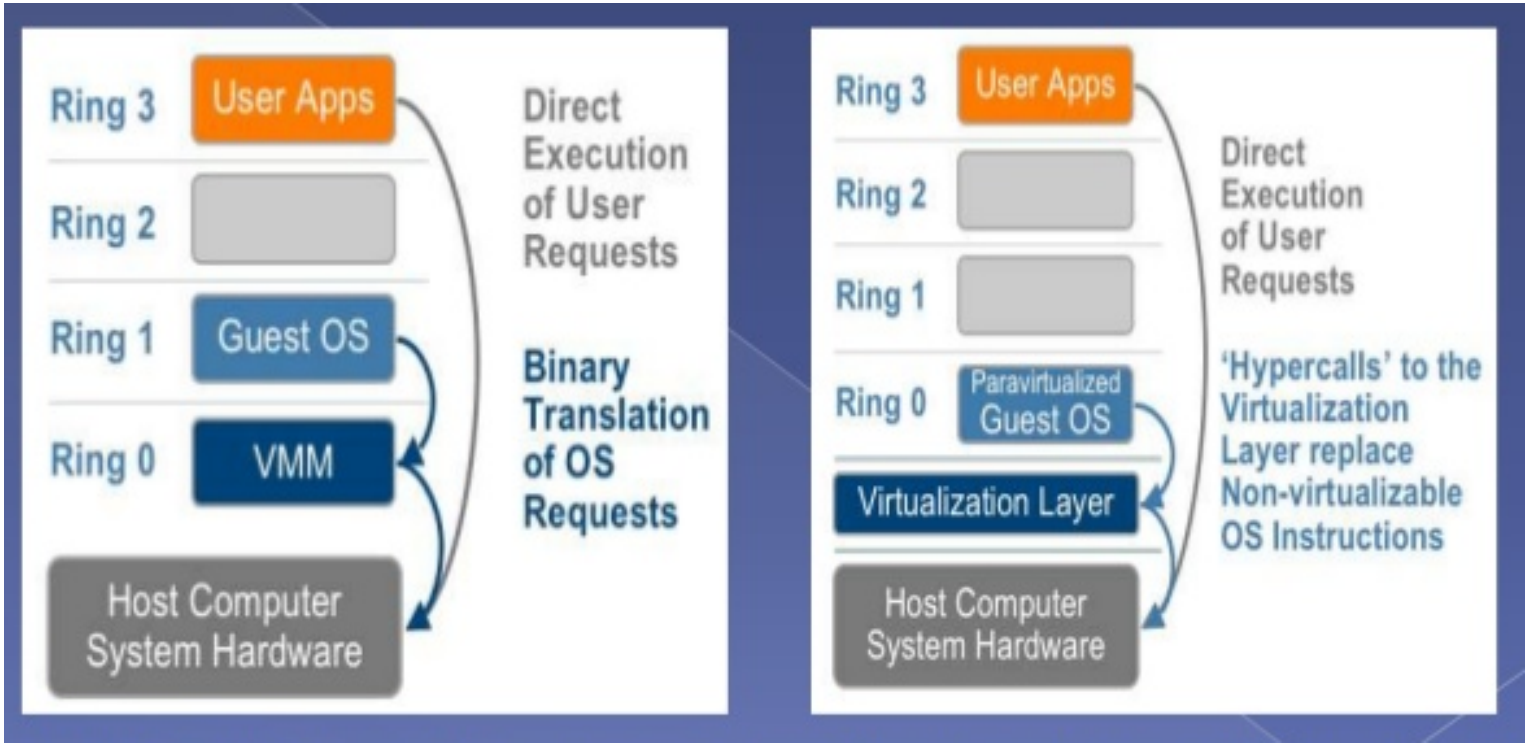
- Balance loads
- Dynamically allocate resources



Multiple guest OS coexistence



Full vs. Paravirtualisation



Paravirtualisation involves modifying the OS kernel.

- Ring 0 operating system kernel;
- Ring 1 device drivers;
- Ring 2 more device drivers;
- Ring 3 user programs.

Source: Gaurav Suri, RedHat inc.

Levels of Virtualisation

Full Virtualisation

- Simulation of entire h/w platform
- Guest OS is unaware that it is in virtualized environment, commands routed to simulated h/w and virtualization is done by host OS
- Examples: VirtualBox, VMWare

Para-virtualisation

- Presents a software implementation for some hardware functions
- Can deliver very high performance but OS must be modified
- Examples: Xen, KVM, VMWare

Hardware-assisted virtualisation

- Virtualisation technologies introduced in the CPU (Intel-VT, AMD-V)
- Increases performance of virtualised OS

Virtualisation Components

- Host and Guest OS
- Virtual Machine Monitors (VMM) or Hypervisors: provide the virtualization capabilities.
  - Act as intermediary between the physical system, also called the host, and the virtualized system, also called the guest.
  - Different hypervisors require different components be installed on the host system to provide virtualization.
  - Different hypervisors provide different options for guest OS.

Types of Hypervisor

Native/Bare metal/Type 1 sits/runs directly on top of physical host's bare-bones h/w. Does not need additional OS, acts as its own OS; hence they can efficiently use physical system's resources; XEN, KVM

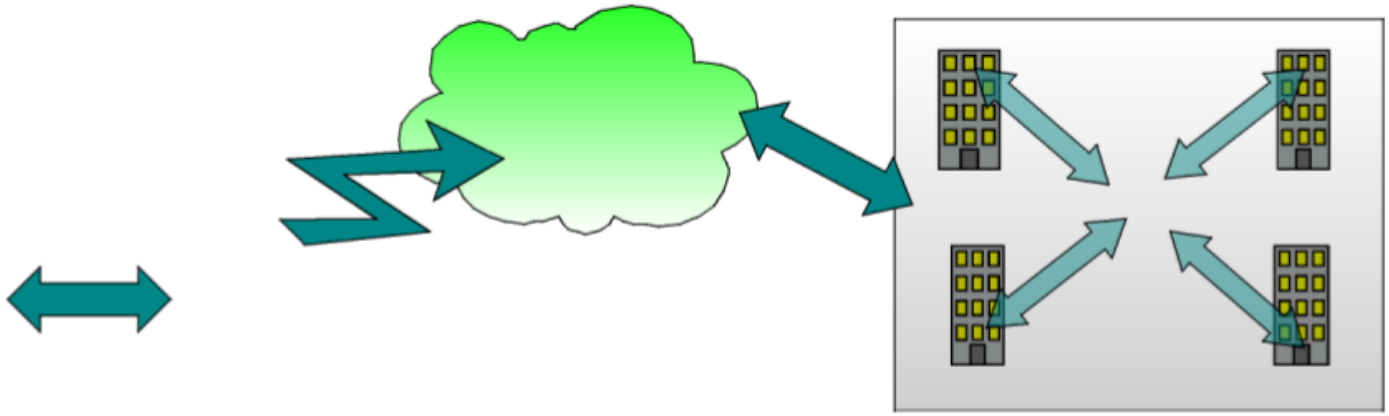
Hosted/Type 2 Sit on top of another OS. The OS controls access to the physical h/w. Act as a control system between host OS and guest OS; we can generally install them on our regular systems. Support the guest VM by coordinating calls for CPU, memory, disk, n/w etc through physical host's OS; VMware, VirtualBox

Top Virtualization Technology Companies

- VMware- Cloud Computing Virtualization OS vSphere
- Citrix- cloud vendor s/w Xen
- Oracle- server virtualization Oracle VM
- Microsoft- Hyper V
- Red Hat
- Amazon- Amazon's Elastic Compute Cloud (EC2)
- Google
- Virtual Bridges
- Proxmox
- Parallels

Centralisation

Move back to centralised resource provision and management



End-user perspective

appears as a single resource

Reality

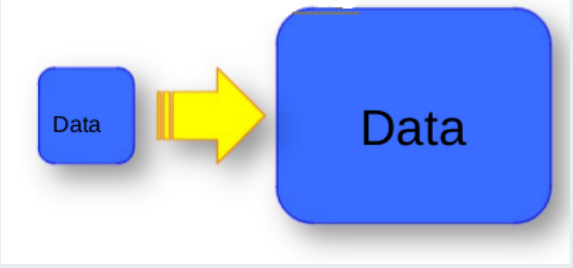
resources may be dispersed throughout vendors' infrastructure

Scalability and Elasticity

Scalability is an important consideration in HPC

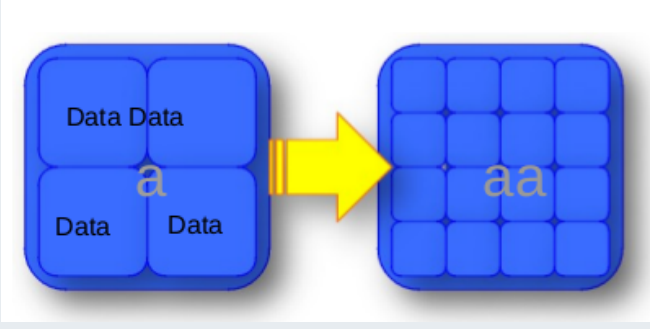
Increase the problem size?

- Increase Time, or
- Increase Resources

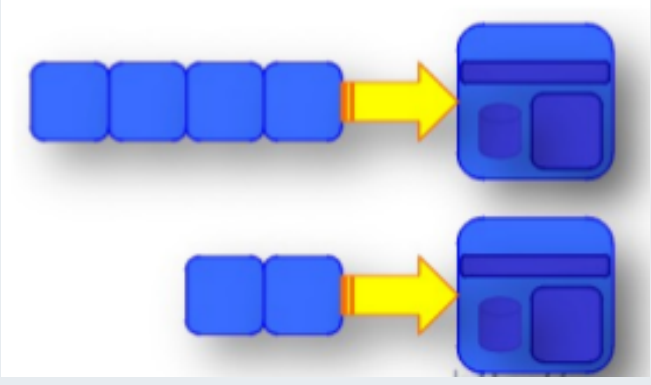


Connected issues

Data/Task Granularity



Load balancing



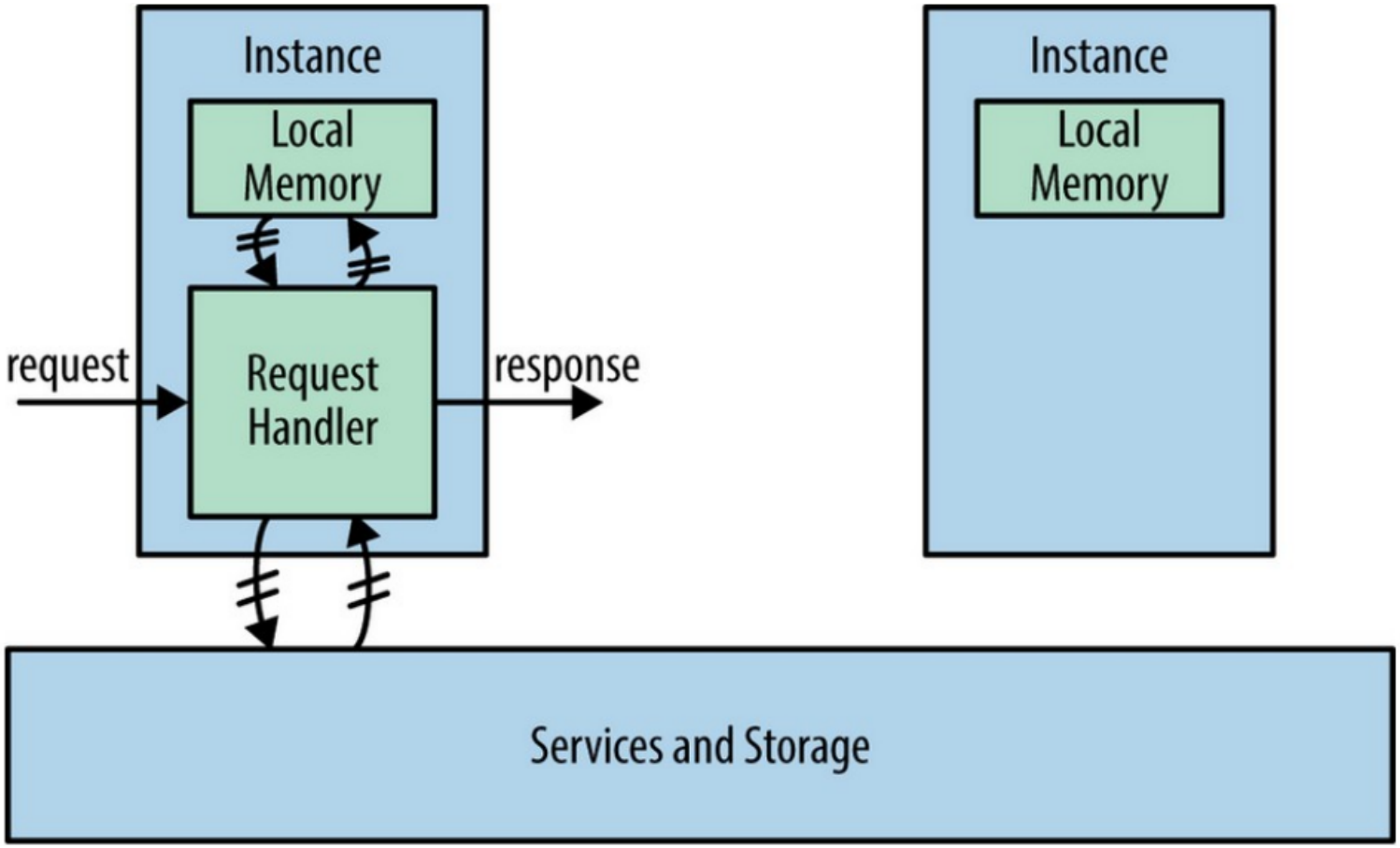
Instances

- The *instance* is your app's unit of computing power.
- It provides memory and a processor, isolated from other instances for both data security and performance. Application code and data stay in the instance's memory until it is shut down, hence opportunity for persistent local storage;
- Within the instance, your application code runs in a *runtime environment* (Python, Java, PHP, Go)
- You never write code to start or stop instances, load balance requests, or monitor resource utilization: this is provided for you.
- Instance uptime is App Engine's billable unit for computation

The Runtime Environment Instance

- App Engine initializes new runtime environments prior to using them to execute *request handlers* (Application code often needs to perform its own initialization that App Engine can't do ahead of time.)
- App Engine's solution: *instances*, long-lived containers for request handlers that retain local memory that are created and destroyed dynamically
- At any given moment, an application has a *pool* of zero or more instances allocated for handling requests;
- App Engine *routes* new requests to available instances based on availability
- It *creates* new instances as needed, and *shuts down* instances that are excessively idle.
- When a request arrives at an instance that has already handled previous requests, the instance is likely to have already done the necessary preparatory work;
- The request handler still only lives as long as it takes to return the response, but its actions can now affect instance memory; *Instance memory is cached only locally*
- Instance memory remains available to the next request handler that executes inside the instance.

Instances


Instance
Local Memory
request
Request Handler
response
Instance
Local Memory
Services and Storage