Cloud computing
– From infrastructure to applications

Clouds and datacenters: Some introductory concepts

Renaud Lachaize & Thomas Ropars
Univ. Grenoble Alpes
M2 MoSIG
September 2023

Goals and topics for these lectures (1/2)

- This class aims at providing students with an overview of **the history, the current state and the upcoming challenges** of data center / cloud computing.

- The progresses in the above domain shave help democratized the access to computing resources and shaped the way to develop and manage applications. Thus, this class should be useful for diverse types of students in computer science & engineering.

- We will **focus on the software aspects at the infrastructure level (i.e., operating systems and middleware)**, while keeping an eye on the evolution of the hardware and the applications.

Goals and topics for these lectures (2/2)

The following topics will be covered:

- An overview of the **main software building blocks** available from cloud platform providers
- The design principles and challenges of **cloud-native applications** and **microservices**
- **Resource management** and coordination services
- **Data processing** architectures and systems

Cloud computing – The roots

- Cluster and Grid computing
- Utility computing (“pay and use”)
- Service-oriented architectures (SOA)
- Virtualization technologies
- Autonomic computing
- DevOps (“you build it, you run it”)

Cloud computing – A definition from NIST (1/9)

- NIST: National Institute of Standards and Technology (USA)
- Definition coined in 2011. Probably the most commonly used definition (although some of its parts have become incomplete today).
- Available from: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
- We will cite it extensively. (emphasis added)

> "Cloud computing is a model for enabling ubiquitous, 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.
> This cloud model is composed of **five essential characteristics**, three service models, and four deployment models."

Cloud computing – A definition from NIST (2/9)

5 essential characteristics (1/3)

1) On-demand self-service:

"A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically **without requiring human interaction with each service provider**."

2) Broad network access:

"**Capabilities are available over the network** and accessed **through standard mechanisms** that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations)."

Cloud computing – A definition from NIST (3/9)

5 essential characteristics (2/3)

3) Resource pooling:

"The provider's computing resources are pooled to serve multiple consumers using **a multi-tenant model**, with **different physical and virtual resources dynamically assigned** and reassigned according to consumer demand.

There is a sense of location independence in that the customer generally has **no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter).**

Examples of resources include storage, processing, memory, and network bandwidth."

Cloud computing – A definition from NIST (4/9)

5 essential characteristics (3/3)

4) Rapid elasticity:
"Capabilities can be elastically provisioned and released, in some cases automatically, to **scale rapidly outward and inward** commensurate with demand.
To the consumer, the **capabilities available for provisioning often appear to be unlimited** and can be appropriated in any quantity at any time."

5) Measured service:
"Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts).
Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service."

Cloud computing – A definition from NIST (5/9)

3 service models (1/3)

1) Software as a service (SaaS):

"The **capability provided to the consumer is to use the provider's applications running on a cloud infrastructure**.

The applications are **accessible from various client devices** through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface.

The **consumer does not manage or control the underlying cloud infrastructure** including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. "

Cloud computing – A definition from NIST (6/9)

3 service models (2/3)

2) Platform as a Service (PaaS):

"**The capability provided to the consumer is to deploy** onto the cloud infrastructure **consumer-created or acquired applications** created **using programming languages, libraries, services, and tools supported by the provider.**

The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."

Cloud computing – A definition from NIST (7/9)

3 service models (3/3)

3) Infrastructure as a Service (IaaS):

"The capability provided to the consumer is to **provision processing, storage, networks, and other fundamental computing resources**
where **the consumer is able to deploy and run arbitrary software**, which can include operating systems and applications.

The consumer does not manage or control the underlying cloud infrastructure
but **has control over operating systems, storage, and deployed applications**;
and possibly limited control of select networking components (e.g., host firewalls)."

Cloud computing – A definition from NIST (8/9)

4 deployment models (1/2)

1) Private cloud:

"The cloud infrastructure is provisioned for **exclusive use by a single organization comprising multiple consumers** (e.g., business units).

It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises."

2) Community cloud:

"The cloud infrastructure is provisioned for **exclusive use by a specific community of consumers from organizations that have shared concerns**

(e.g., mission, security requirements, policy, and compliance considerations).

It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises."

Cloud computing – A definition from NIST (9/9)

4 deployment models (2/2)

3) Public cloud:

"The cloud infrastructure is **provisioned for open use by the general public.**
It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. **It exists on the premises of the cloud provider**."

4) Hybrid cloud:

"The cloud infrastructure is a **composition of two or more distinct cloud infrastructures (private, community, or public)** that remain unique entities, but are bound together by standardized or proprietary technology that **enables data and application portability** (e.g., cloud bursting for load balancing between clouds)."

Beyond IaaS, PaaS and SaaS: Other service models

- “Anything/Everything as a Service” (also sometimes named “XaaS”).

- A few examples:
  - Hardware as a Service
  - Monitoring as a Service
  - Various compute-level abstractions that sit between IaaS and PaaS. For example (discussed later) :
    - Containers as a Service (CaaS)
    - Functions as a Service (FaaS)

Data centers

- A data center (DC) is a room/building hosting a set of servers and their network fabric (+ cooling facilities + possibly power generators).
- Typical scales range from 100s to ~10,000s or up to ~100,000 servers per DC.
- The basic hierarchical unit is the rack (a column of severs), which contains a few 10s of server boards (a.k.a. “blades”) + network switches.
- Machines within the same rack:
  - Have faster network communications
  - Have more correlated failures (due to power, networking, heating or hardware issues)

For a detailed introduction on data center design and challenges,
see the following book (freely available),
written by some of the lead scientists and executives at Google:

The Datacenter as a Computer. Designing Warehouse-Scale Machines. 3rd edition. Morgan & Claypool publishers. 2018. https://www.morganclaypool.com/doi/abs/10.2200/S00874ED3V01Y201809CAC046

Data center communications (1/2)

- **Intra**-DC latencies

Round-Trip Times (RTT)
in microseconds
for communications between two machines in the same DC
for various cloud providers and various DCs.
(May 2017).

(Source: D.A. Popescu.
"Latency-driven performance in data centres".
PhD Thesis. U. of Cambridge. 2019.
https://www.repository.cam.ac.uk/handle/1810/291685)

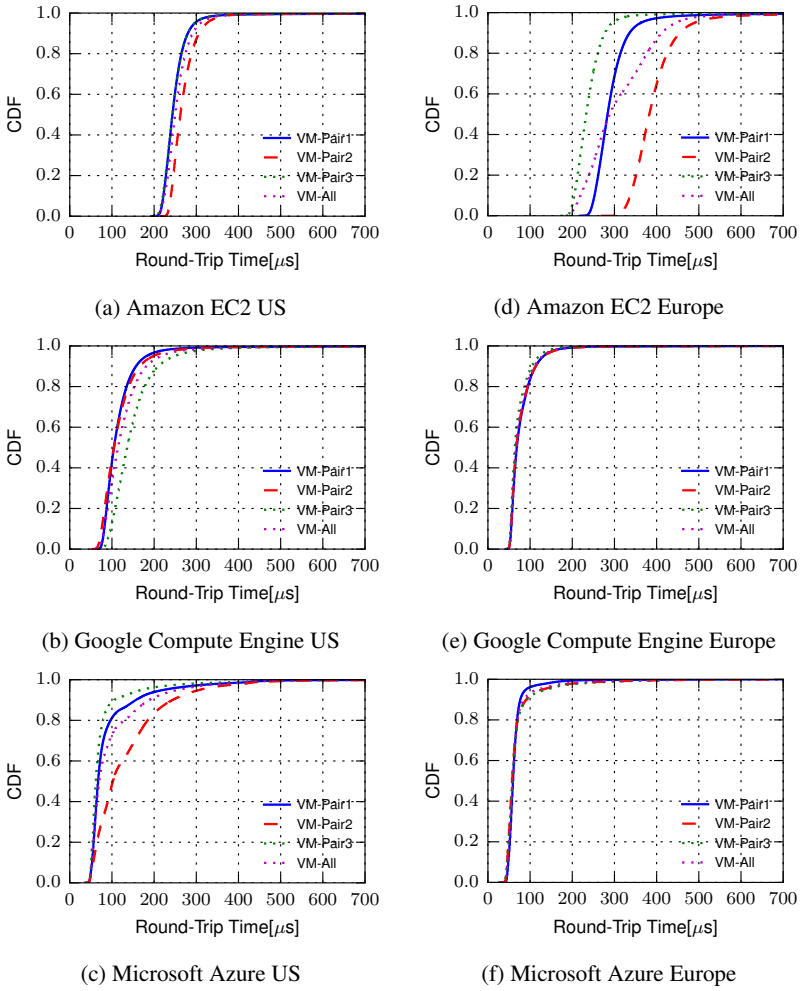
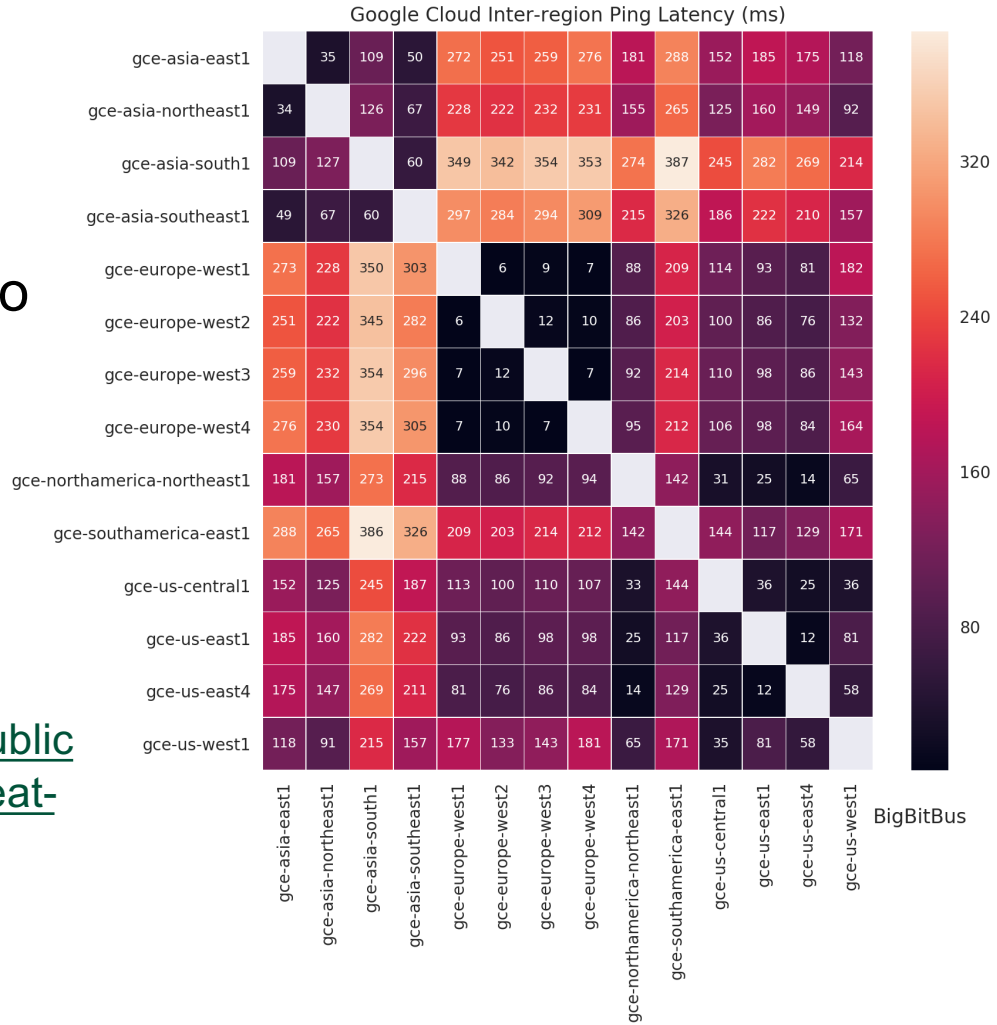


Figure 2.6: Measured RTTs within data centres for Amazon EC2, Google Compute Engine, and Microsoft Azure in May 2017.

Data center communications (2/2)

- **Inter**-DC latencies

Round-Trip Times (RTT)
in milliseconds
for communications between two machines in two distinct DCs of Google Cloud Platform (August 2018)



(Source: S. Agarwal https://medium.com/@sachinkagarwal/public-cloud-inter-region-network-latency-as-heat-maps-134e22a5ff19)

Scalability

- (Also known as **"scaling"**)
- *A generic term related to **the expected behavior of a system/application when the input load and/or the available resources evolve.***
- Two types of considerations: "weak scaling" and "strong scaling"
- **"Weak scaling":**
  - "**Can I achieve more work per unit of time** if I increase the quantity of resources in my system?" *(And if so in which proportions?)*
  - Here, "more work" can mean "more requests" or "more/larger data items" or "more complex tasks" …
- **"Strong scaling':**
  - "**Can I complete the same work in a shorter amount of time** if I increase the quantity of resources in my system?" *(And if so in which proportions?)*

Elasticity

- A notion related to scaling (mainly “weak scaling”) and to the flexible resource reservation/billing model of cloud computing.
- Sometimes also called **“autoscaling”**
- Corresponds to **the ability of a system to dynamically adjust (i.e., grow or shrink) the allocated resources according to the current input load.**
- Example:
  - “My Web-based e-commerce application should handle any request in less than 500ms, regardless of the total number of requests that it is currently handling.”
  - However, you do not want to dimension your system for the worst case: wasted resources would lead to high costs.
- Ideally, an elasticity manager must be:
  - Quick to react (can be predictive and/or reactive)
  - Accurate
  - Fully automated

Vertical vs. horizontal scaling

- In practice, there are two main approaches to achieve (weak or strong) scaling: vertical vs. horizontal scaling.

- **Vertical scaling**, also known as **"scaling up"**:
  - In essence, consists in **using more powerful machines**.
- **Horizontal scaling**, also known as **"scaling out"**:
  - In essence, consists in **using a larger number of machines**.

- Both approaches can be combined.
  - But in the design of a large-scale system, one of the two will generally dominate.
  - Resorting to horizontal scaling is almost unavoidable for some requirements because it introduces distribution, which enables replication & geo-placement.

Vertical scaling

- **Idea:** **Replace existing machine(s) with more powerful one(s)**
- **Examples:**
  - Faster CPUs (better microarchitecture, higher clock frequency)
  - More CPU cores
  - Faster RAM
  - Greater RAM capacity
  - Faster disks
  - Faster networking
  - Better hardware accelerators
- **Pros:**
  - Relatively simple
  - May work well in some cases
- **Cons:**
  - Limited scalability potential
  - Non-linear increase of hardware costs

Horizontal scaling

- **Idea: Increase the number of machines in the system**
  - Typically using commodity servers (i.e., no specific/expensive hardware required)
  - Coordination between machines is performed at the software level, using a conventional network
  - Warning: the network must be dimensioned accordingly to avoid bottlenecks
- **Pros:**
  - **Better scalability potential** (esp. for very large scales)
  - **Better cost-effectiveness** (esp. for very large scales)
  - Distribution **enables fault-tolerance (replication) and performance optimizations**
- **Cons:**
  - Using a large number of machines **increases the probability of failures.**
  - **More efforts required** from humans to achieve good performance and reliability:
    - Middleware/infrastructure designers
    - System administrators
    - Application developers (often)

Vertical vs. horizontal scaling in the cloud

- Historically, over the last 20 years, cloud computing platforms have mainly been been built according to the horizontal scaling approach.
- **Our lectures will mostly focus on techniques related to horizontal scaling.**
- **However:**
  - **Today, many cloud computing platforms (especially public cloud providers) provide solutions for both models.**
    - Greater diversity of available hardware (CPUs, RAM, disks, networking, accelerators).
    - Possibility to reserve dedicated machines.
  - **In some circumstances, vertical scaling is just simpler and more cost-effective.**
    - See some of the next slides for examples.
  - For some types of cloud services, the internal design is completely abstracted for the end-user.

Distributed systems – Main techniques

- Software infrastructures for the Cloud are **inherently distributed systems**.
- Distributed systems extensively rely on **two techniques**: replication and partitioning.
  - These two approaches are often combined.
  - These two approaches can be applied to processing and storage.
- **Replication**
  - Keeping a copy of the same service/data on multiple nodes, potentially in different geographical locations.
  - Provides redundancy, which is useful for fault-tolerance and can also help improve performance.
- **Partitioning** (also known as "**sharding**")
  - Splitting a data set into multiple partitions, and routing a given request to the service in charge of the appropriate partition.
  - Helpful for performance (load balancing), and possibly for fault tolerance (partial availability).

Deployment archetypes for Cloud applications (1/7)

- There are **six main types of deployments** for cloud applications:

  zonal, **regional**, **multi-regional**, **global**, **hybrid**, **multi-cloud**

- The deployment type has a **major impact on the overall availability of an application**, which includes the following aspects:
  - the time to access the application
  - the time to get a response with valid results
  - the assurance that data is stored and maintained with integrity
  - the ability to scale and handle peak traffic demands

- We will briefly review each type, with quotes from the **following reference**:

  A. Berenberg and B. Calder.

  Deployment Archetypes for Cloud Applications

  ACM Computing Surveys. Vo. 55., No. 3, February 2022.

  https://dl.acm.org/doi/full/10.1145/3498336

Deployment archetypes for Cloud applications (2/7)

- **Zonal**:
  - "All components of an application run **within a single zone**. A zone provides **a set of clusters** with the infrastructure needed to run services (compute, storage, networking, data, etc.) within that zone."
  - "Should a zone go down, what is running within the zone is either restarted in another zone from the last checkpointed state, or a failover occurs to a standby instance of the application in another zone."

Deployment archetypes for Cloud applications (3/7)

- **Regional**:
  - "All components of an application are deployed and run out of one cloud region. **A region consists of 3 or more zones, where each zone is treated as a separate fault domain**. High availability can be achieved by replicating the application across zones within the region."
  - "These applications are typically designed to run with a data store that shares data and makes it accessible across that region. To serve application traffic, the requests are load-balanced across compute instances in multiple zones."
  - "To further increase availability and reliability, some applications may have a secondary standby region with an asynchronous copy of the data, where the application can failover to the secondary region in case the primary region is not available."

Deployment archetypes for Cloud applications (4/7)

- **Multi-regional**:
  - "The application serving stack runs and is stitched together **across multiple regions to achieve higher availability and low end-user latency through geographic distribution**."
  - "In this deployment archetype, data is typically replicated and shared across regions. This archetype is commonly used for applications that want to achieve high availability, such as user-facing applications."

Deployment archetypes for Cloud applications (5/7)

- **Global**:
  - "**The application stack is spread and replicated across cloud regions around the globe and data is available worldwide via global databases and storage**."
  - "Applications consisting of a large number of services and microservices benefit from this deployment archetype. This is the five-nines deployment model used by retail, social media and other businesses requiring always-on availability, while running large services economically."

Deployment archetypes for Cloud applications (6/7)

- **Hybrid**:
  - "Applications that have deployments **combining on-premises and public cloud**(s) are becoming increasingly common."
  - "Hybrid application availability and resilience is often achieved by:
    - (a) creating deployment archetypes that leverage failover between on-premise and Cloud,
    - and (b) coordinating the execution of parts of the application that run in the Cloud versus run on-premises."

Deployment archetypes for Cloud applications (7/7)

- **Multi-cloud**:
  - "Applications can potentially gain the highest availability by **using two or more public cloud platforms at the same time**, to protect against one cloud's unavail-ability."
  - "**In each cloud, one of the deployment archetypes listed previously is used, and then combined across clouds to create a multi-cloud deployment**."
  - "This deployment archetype is in its infancy, but applications that require the highest availability are prime targets for multi-cloud deployments as this model evolves."

A Warning about modern software infrastructures (1/2)

- Nowadays, software architects and developers are overwhelmed with technical opportunities:
  - Cloud platforms provide them with a very large supply of computing resources that they can try/consume with “a pay-as-you-go model”.
  - A very rich ecosystem of high-quality open-source software (developed by major companies/organizations) provides them with freely available software building blocks that they can leverage to tackle complex problems.

- In this context, it is sometimes easy to lose some common sense. This may result in choosing system designs that are unnecessarily complex or inefficient for a specific problem.
  - In other words: “you are not necessarily like Google (or Amazon/Microsoft/…), even though you are possibly using some of their infrastructure!”

A Warning about modern software infrastructures (2/2)

- **Example #1:**
  - A majority of real-world analytic jobs process less than 100GB of input data. For many of such jobs, a single “scale-up” server (i.e., a mid-range multicore server) can do as well or better than a cluster in terms of performance, cost, power and server density.
  - [R. Appuswamy et al. “Nobody ever got fired for buying a cluster”. 2013.]

- **Example #2:**
  - For some tasks, a single threaded program running on a decent laptop may outperform a distributed processing framework running on more than 100 cores.
  - [F. McSherry et al. “Scalability? But at what COST!”. May 2015.]

- For more examples, see also:
  - Oz Nova. “You are not Google”. June 2017. https://blog.bradfieldcs.com/you-are-not-google-84912cf44afb

Beyond clouds & datacenters: Edge and Fog computing

- There are no universally accepted definitions for these concepts. We will nonetheless try to highlight some key characteristics.
- "**Edge computing** is a distributed computing paradigm that brings computation and data storage closer to the sources of data." (source: Wikipedia)
- "**Fog computing**" is sometimes used as a synonym for "edge computing" but is also often used with a distinct meaning.
  - According to the NIST definition: "Fog computing runs applications in a multi-layer architecture that decouples and meshes the hardware and software functions, allowing for dynamic reconfigurations for different applications while performing intelligent computing and transmission services. Edge computing runs specific applications in a fixed logic location and provides a direct transmission service. Fog computing is hierarchical, where edge computing tends to be limited to a small number of peripheral devices."
- For more details:
  - https://en.wikipedia.org/wiki/Edge_computing
  - Industrial Internet Consortium. Introduction to Edge Computing. https://hub.iiconsortium.org/intro-edge-computing
  - NIST. Fog Computing Conceptual Model. 2018. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-325.pdf