Distributed Systems: Introduction to cloud computing

Bert Lagaisse, Stefan Walraven, Wouter Joosen, Tom Van Cutsem

DistriNet, KU Leuven

October, 2023

Overview

- Cloud computing ?
  - = business model + technical architectures

- Case studies:
  - Google AppEngine
  - Microsoft Azure,

Definitions of cloud computing?

Cloud computing: essentials

- Outsourcing, pay per use model
- For on-demand, web-based access
- to shared pool of computing resources
  - Virtual machines: Storage, cpu, network
  - Applications

- 3 types of services in cloud computing model
  - Infrastructure as a service (IaaS):
    - virtual machine , storage, network
  - Platform as a service (PaaS):
    - middleware and web-hosting platform (php, .net, java)
  - Software as a service (SaaS):
    - Zero-install, online applications (CRM, Google Mail, Google Apps)

amazon web services

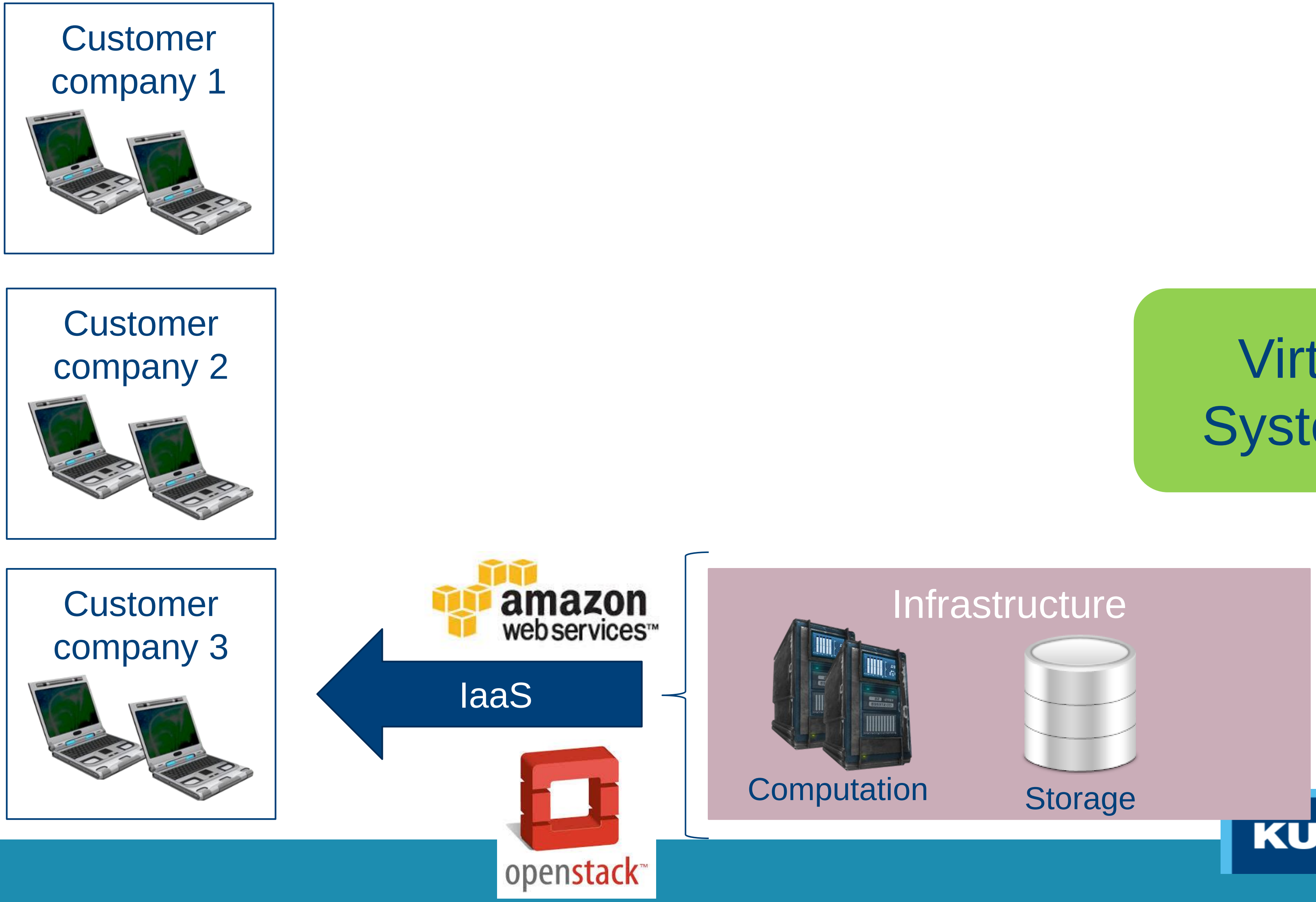
Azure

Google App Engine

IaaS: private or public ?

- IaaS: infrastructure as a service
  - Virtual machines on shared hardware !
    - With operating system of choice
  - Virtual networks between machines
  - Storage (Blob storage, structured storage)
- Public cloud: Amazon, MS Azure
  - Accessible by any one from anywhere
  - (When you have a credit card.☺)
  - Many data centers around the world
    - Choose where you want your resources in advance.
- Private cloud: OpenStack, Xen Cloud platform
  - private data centers of companies (in Flanders: integrators such as Cegeka, Telecom operators)
  - Self-service creation of virtual machines etc.

Infrastructure as a Service



Customer company 1

Customer company 2

Customer company 3

IaaS

amazon web services™

openstack™

Infrastructure

Computation

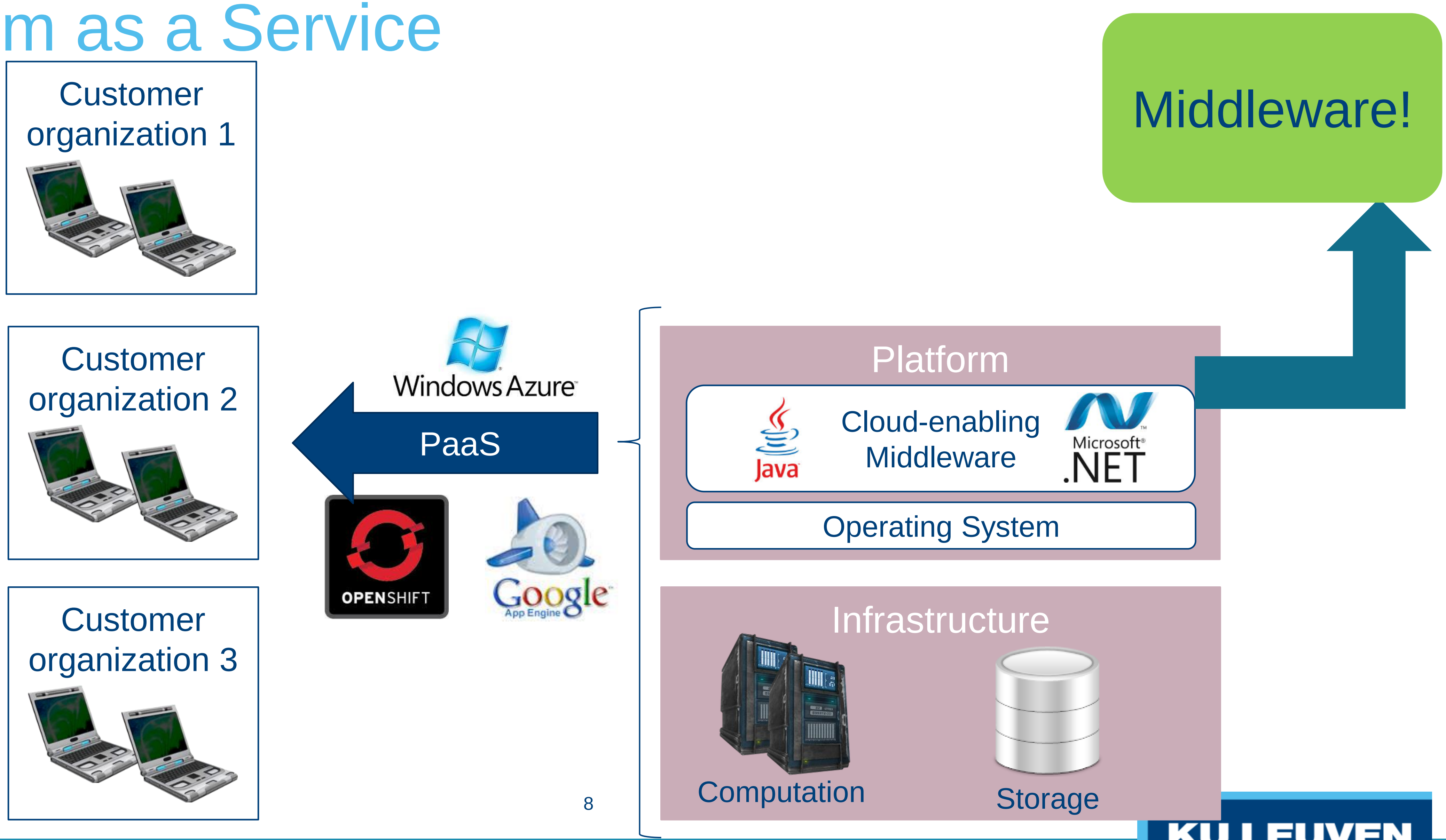
Storage

Virtual Systems!

PaaS: focused on scaling-out and resource sharing

- Middleware as a service
  - scalable web server, web container, app server, background workers with triggers,…
  - without the burden of setting up the whole VM underneath
- Google AppEngine, MS Azure, OpenShift.
- Software architecture properties: **scalability** first !
  - Different way of thinking about software entities and their interactions
    - From: object-based, synchronous method invocations
    - To: Task-based, background workers, with asynchronous message passing (not locking resources)
  - Different way of thinking about persistence
    - Due to scalability requirements: must scale-out
    - Need for replicated data storage with simple access (e.g. key based retrieval of simple entities, blobs)
- **Resource sharing** has impact on performance and failure isolation: multiple applications run on the same web server, VM.

Platform as a Service



Customer organization 1

Customer organization 2

Customer organization 3

PaaS

Windows Azure

OPENSHIFT

Google App Engine

Platform

Cloud-enabling Middleware

Java

Microsoft .NET

Operating System

Infrastructure

Computation

Storage

Middleware!

Next-gen PaaS:
IaaS → VM-based PaaS → serverless model

- Initial cloud hype & popularity
  - IaaS !
  - Easily create VMs anywhere, when you need them
- The systems need to be managed☹ - operational challenges remain, e.g.
  - Installation and maintenance of software: Web server, App server, web frameworks, etc.
  - Configuration of the network between the VMs
  - → Devops = automate the deployment of VMs and software
    - Requires relatively complex scripts for automation, risks to be error prone.
    - Example: robust management of Kubernetes Clusters
- "Serverless" model:
  - Don't manage the whole (generic) server software stack
    - Focus on application logic, and deploy application software "as smoothly as possible".
  - Let cloud provider handle automatic scaling of infrastructure
  - Let cloud provider handle the management of the software stack
  - Examples: AWS lambda, (Google AppEngine, Azure Web Apps)

CACM September 2023

- *“Serverless Computing: What It Is, and What It Is Not?”, pp 80-92*
- → allow to develop deploy and run applications

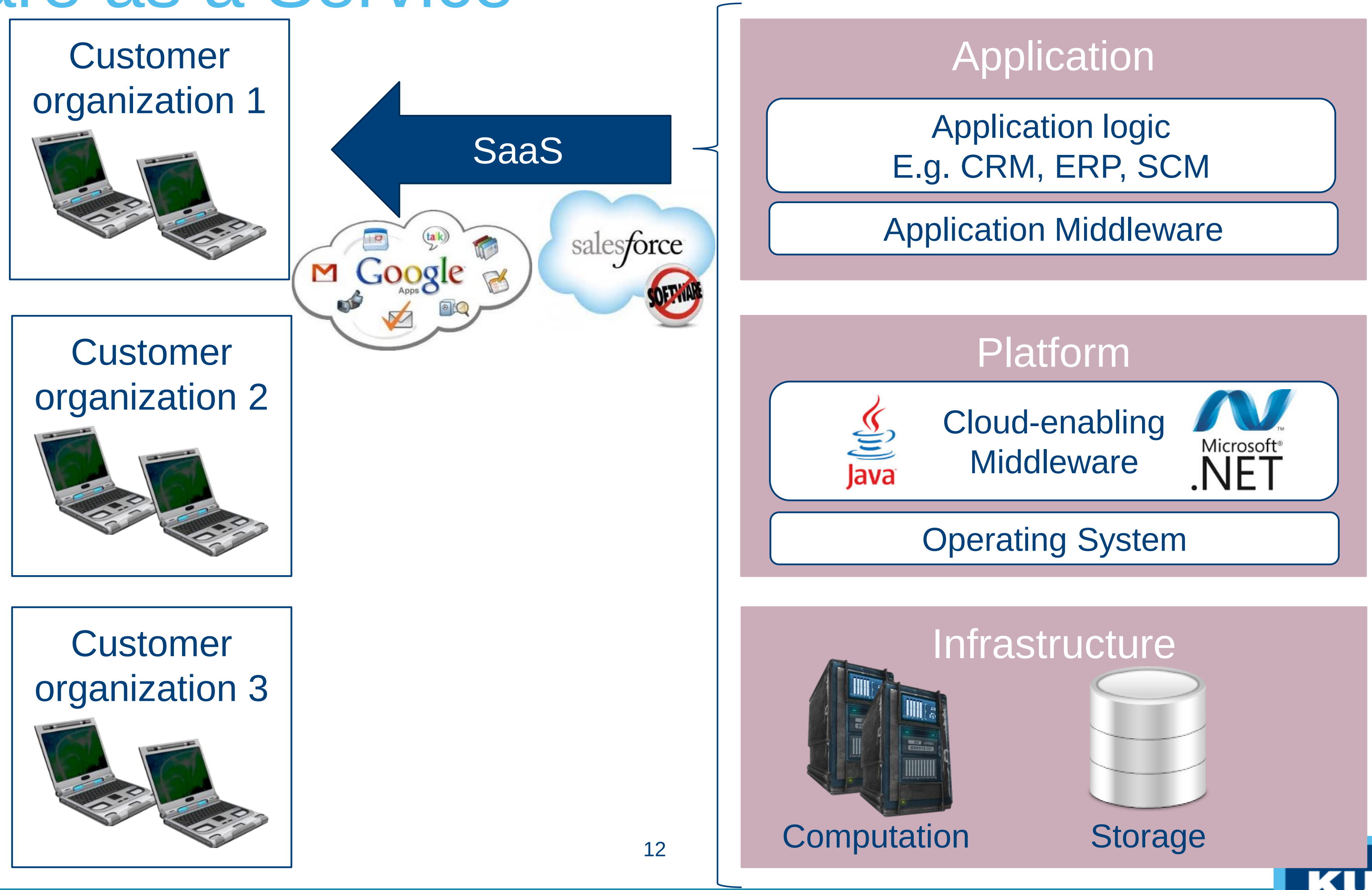
- Key insights -selected
  - Serverless computing means full automation and fine-grained utilization-based billing
  - Serverless computing supports diverse applications, *(e.g.)* from enterprise automation to scientific computing
- Note: projected market value of serverless computing $36.8 billion by the end of 2028 – Ref. 49

Is Serverless computing in 2020 +

… what cloud computing has been in 2010+?

Is serverless computing indeed the next concept/paradigm?

Software as a Service



Customer organization 1

Customer organization 2

Customer organization 3

SaaS

Application

Application logic
E.g. CRM, ERP, SCM

Application Middleware

Platform

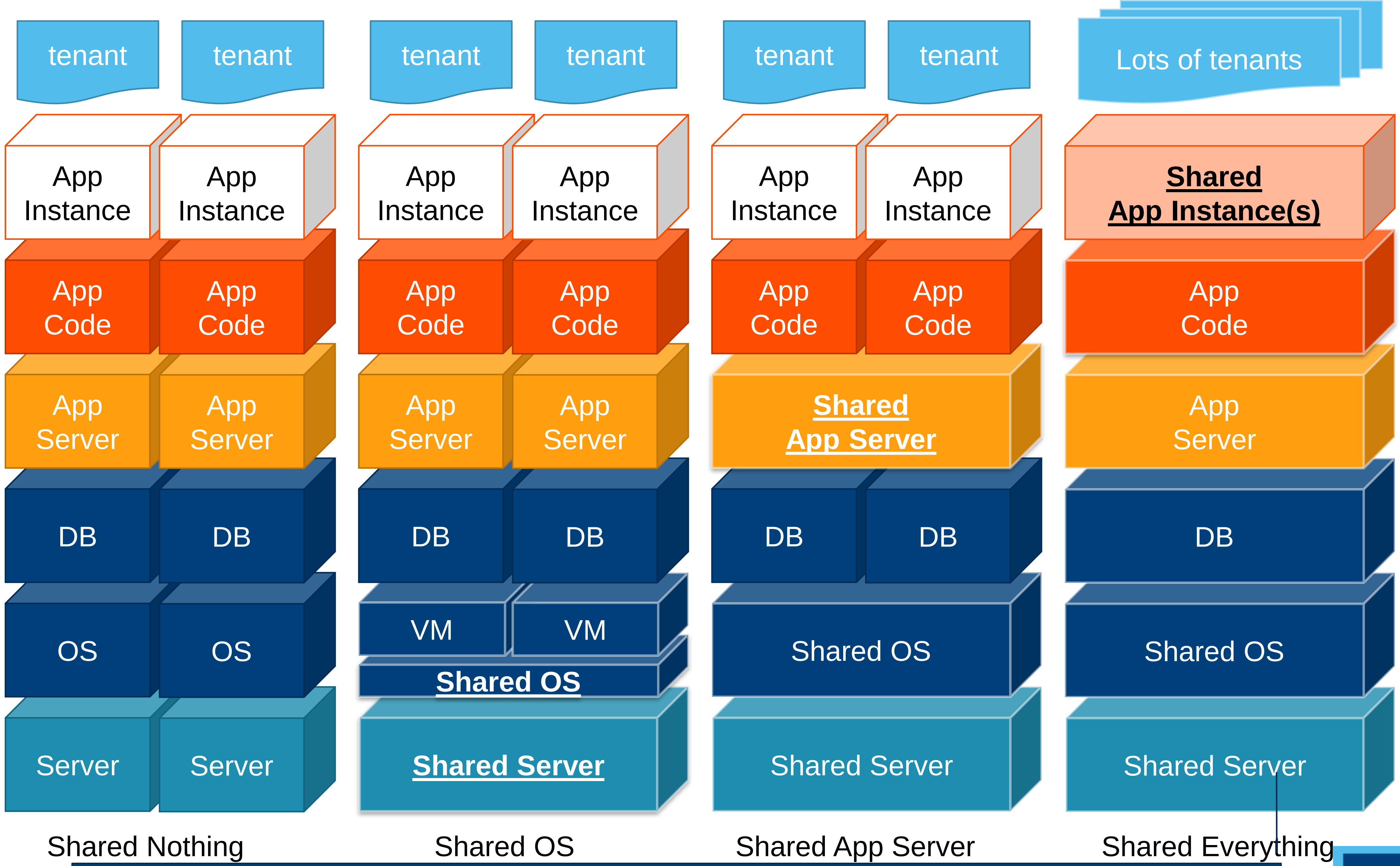
Cloud-enabling Middleware

Operating System

Infrastructure

Computation

Storage



SaaS: Application-level logic to separate tenants:
CHALLENGES: Customization, performance, lots of data, security logic,…

KU LEUVEN

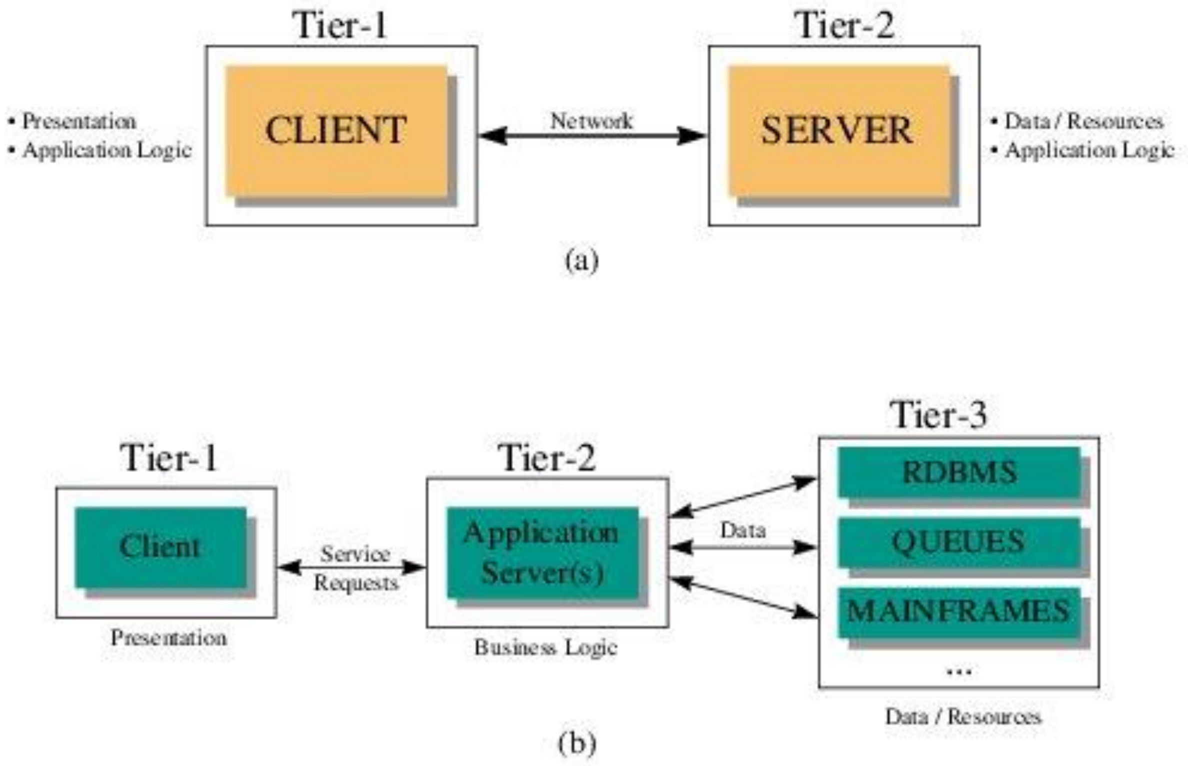
Architectural paradigm shift

- Paradigm shift in business model
  - As discussed before: towards pay per use
- But also at the architectural level: next distributed architecture
  - Scientific and enterprise computing have evolved to a common architecture … ?

Mainframe	1970s	Centralized bulk data processing, multi-user access on terminals (keyboard-screen)
Client-server	1980s	Fat clients on workstations, centralized database server
Web	1990s	Multi-tier: web tier, application servers, database .com your business
SOA	2000s	Expose automated business process as a web service B2B communication
Cloud	2010+	Centralized provision of dynamically scalable, multi-tenant software as a service over the Internet Builds further upon SOA (→ REST)

Two Tier architecture

- Two tier architecture
  - Client:
    - Fat client on desktop
    - can contain application logic
  - Server:
    - Data base server with app logic
      - E.g. stored procedures & transactions
- 3 tier architecture
  - user interface (presentation), functional
  - process logic (“business rules”),
  - computer data storage and data access

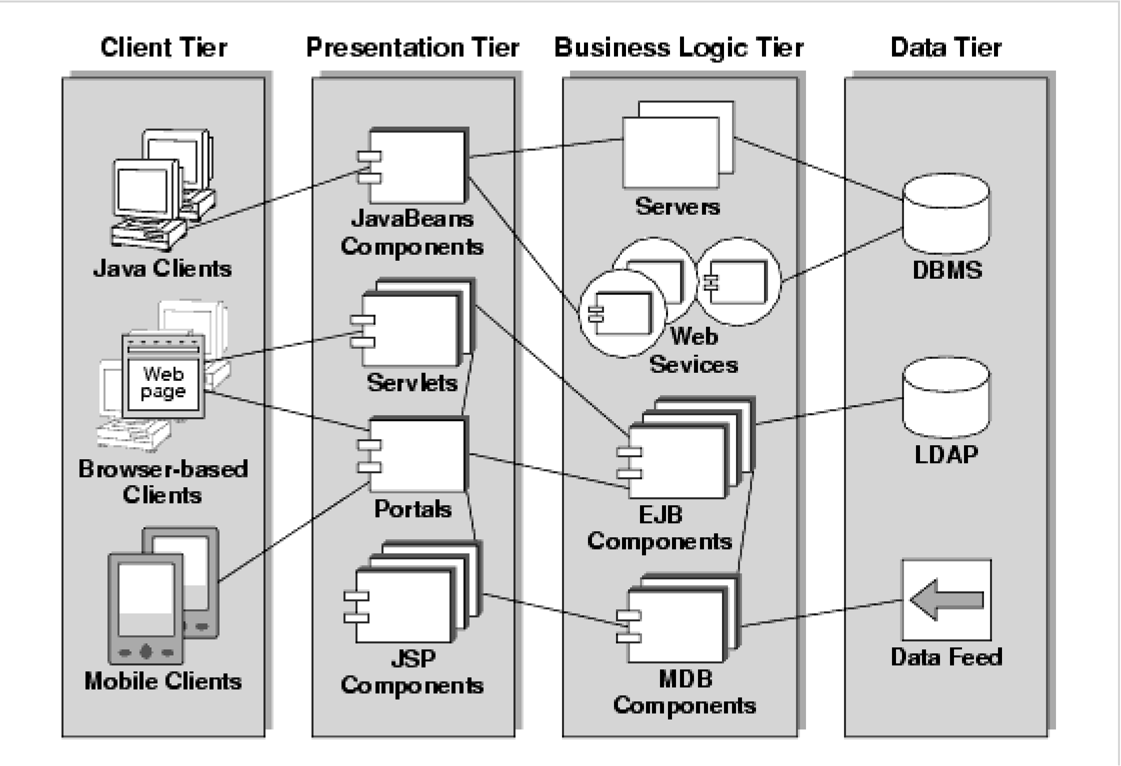


(a)

(b)

Multi-tier architecture

- Client tier: client machines
- Presentation tier (server-side)
  - displays information related services
  - communicates with other tiers
  - puts out the results to the browser/client tier and all other tiers in the network.
- Application tier (business logic, logic tier, or middle tier)
  - The logical tier is pulled out from the presentation tier and,
  - controls an application’s functionality by performing detailed processing.
- Data tier
  - includes the data persistence mechanisms (database servers, file shares, etc.)
  - Application Programming Interface (API) to the application tier that exposes methods of managing the stored data



Back to cloud

5 core characteristics of cloud

Core characteristics

- 5 specific characteristics that differ from what is offered by previous paradigms:
  - Higher degree of distribution
  - Multi-tenancy
  - Elasticity
  - Delivery as a service
  - Self-service

Core characteristics

- 5 specific characteristics that differ from what is offered by previous paradigms:
  - **Higher degree of distribution**
    - Execution environment is **distributed at a large scale**
      - Across different data centers (using commodity hardware)
      - At different geographical locations (geo-distribution)
    - to achieve **high availability and high performance** (i.e. high throughput and low latency) via replication
    - accessible from any device and location through standard mechanisms and APIs
  - Multi-tenancy
  - Elasticity
  - Delivery as a service
  - Self-service

Core characteristics

- 5 specific characteristics that differ from what is offered by previous paradigms:
  - Higher degree of distribution
  - **Multi-tenancy**
    - Dynamically assigning the available **shared resources** to multiple customer organizations (= tenants)
    - One of the key enablers to realize **economies of scale**:
      - operational costs are significantly reduced by multiplexing (shared) resources among multiple tenants
      - simplifies administration, provisioning and maintenance
  - Elasticity
  - Delivery as a service
  - Self-service

Core characteristics

- 5 specific characteristics that differ from what is offered by previous paradigms:
  - Higher degree of distribution
  - Multi-tenancy
  - **Elasticity**
    - <u>Dynamically provisioning </u>and releasing resources on demand
      - by **scaling horizontally**: adding more servers to the system and distributing the load (⇔ making servers more powerful)
    - **<u>Flexibility</u>** to scale out rapidly (and automatically) at a fine-grained level based on the current workload
      - without having to constantly provision sufficient resources for peak demand and thus avoiding over- or underprovisioning
  - Delivery as a service
  - Self-service

Core characteristics

- 5 specific characteristics that differ from what is offered by previous paradigms:
  - Higher degree of distribution
  - Multi-tenancy
  - Elasticity
  - **Delivery as a service**
    - Management of resources and applications **outsourced** to the cloud provider
      - No upfront commitments and investments in infrastructure
      - Only pay for resources and applications that are used (**pay per use**)
    - Cloud provider has to monitor usage per customer
  - Self-service

Core characteristics

- 5 specific characteristics that differ from what is offered by previous paradigms:
  - Higher degree of distribution
  - Multi-tenancy
  - Elasticity
  - Delivery as a service
  - **Self-service**
    - To preserve **scalability** with an increasing # customers
    - To **rapidly respond** to changing service demand and customer requirements
      - **Automation** (i.e. no human interaction with cloud provider)
      - **Empower customers** to manage cloud services (via APIs, configuration interfaces, tools), e.g. to scale out

Some of the key concerns – still (think *B2B*!)

- Lock-In
  - Hybrid Cloud and Multi-Cloud
- Regulatory Compliance (and Customer needs)
- Security
- SLA/SLO monitoring
  - Service Level Agreements – Service Level Objectives
- → Resilience Management

Going Forward…

- Case studies from large cloud providers
  - Google AppEngine
  - (Considering Azure)