University of Bologna

Dipartimento di Informatica – Scienza e Ingegneria (DISI)

Engineering Bologna Campus

Class of Infrastructures for Cloud Computing and Big Data M

Middleware & Cloud models

Antonio Corradi

Academic year 2018/2019

MIDDLEWARE

The term MIDDLEWARE has an obvious meaning

The set of tools that **sit in the middle between the application and the low-level support**: i.e.,

hardware, local **operating system** and **technology**, …

The term **middleware** goes back to **1968**, to a famous **NATO school on Software Engineering**

Anyway, middleware was not so **significant until the 90s**, when **distributed systems** became widespread and commonplace

Middleware is a solution to design and support **complex, distributed, deeply heterogeneous systems** also suitable for **very heterogeneous organizations to provide very differentiated services**

MIDDLEWARE

Another definition of MIDDLEWARE

The set of tools that allow integrating different application and services to be used in open **environment (heterogeneous) with an unlimited lifecycle, at least the whole organization life**

Middleware are offering and proposing the **support tools,** to **control** and **managing services during execution**, at all systems level, from physical one up the application level

RPC middleware (RMI)

They propose the usage of Remote Procedure Call as the unique communication tool among all available layers

The interaction is both between systems and final users (B2B and also **B2C)**

MIDDLEWARE: HETEROGENEITY

It is a middleware the infrastructure to overcome the problems inherent to ad-hoc solutions, or **ad-hoc approaches**

- custom conversion functions
- generic format conversions
- wrappers
- common protocols

Legend:

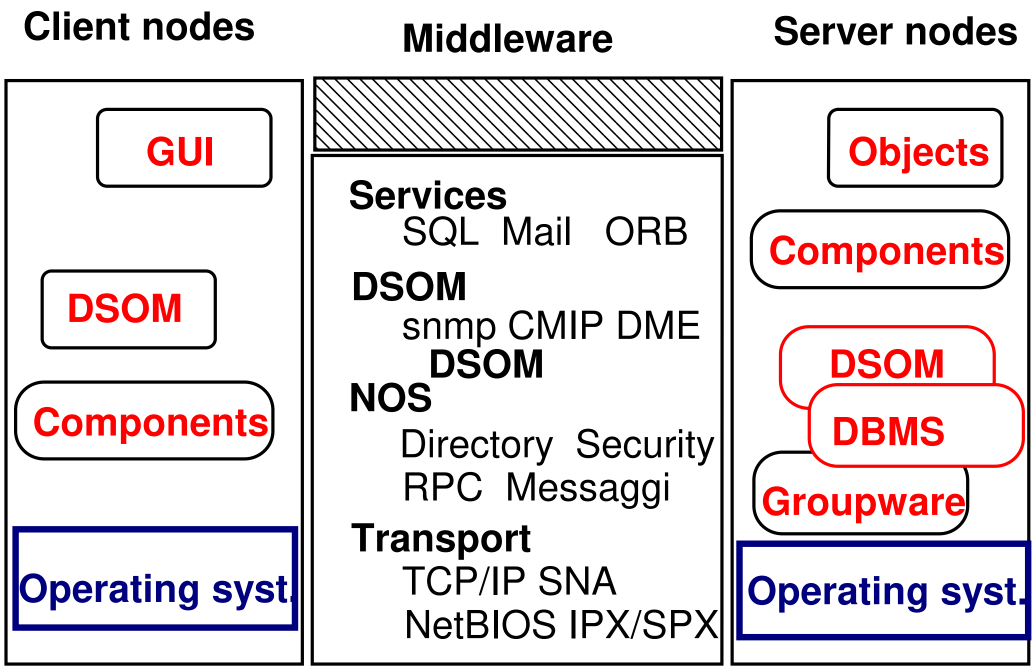
DSOM IBM management system

DME standard Open Software Foundation management system

SNA IBM network architecture

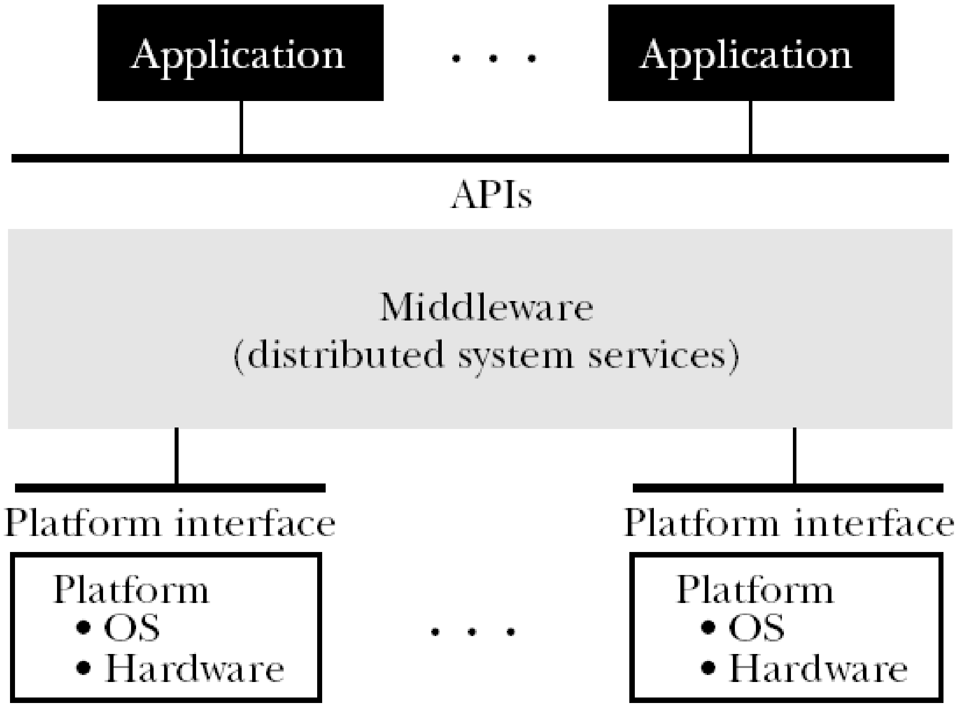
IPX/SPX Xerox network architecture

…



Another DEFINITION of MIDDLEWARE

The **software layer** resident between the **applications** and all local support levels: network **components, local operating system**, **heterogeneous hardware**, different application areas. That layer must grant **any application area operations**

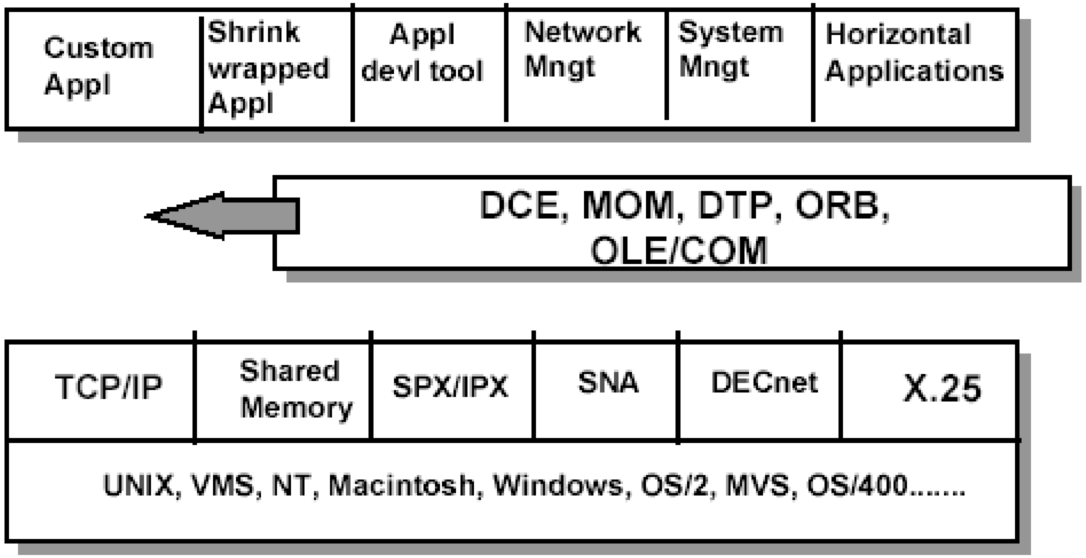


The decoupling layer among all system layers to permit a **continuous simplified design** of any **application part** (and also of the **support part itself**) and also allow any overcoming of the **intrinsic heterogeneity**

MIDDLEWARE

the Middleware is often invoked and activated in **a transparent and implicit way** to provide a **uniform access (API) to intrinsically heterogeneous local functions**

- Often used to **integrate legacy systems** since long available (obsolescent) but required by the **business logic**
- Often used as a **standard** (de facto or committee-based) **for a limited or even large community**



Custom Appl

Shrink wrapped Appl

Appl devl tool

Network Mngt

System Mngt

Horizontal Applications

DCE, MOM, DTP, ORB, OLE/COM

TCP/IP

Shared Memory

SPX/IPX

SNA

DECnet

X.25

UNIX, VMS, NT, Macintosh, Windows, OS/2, MVS, OS/400.......

MIDDLEWARE SUPPORT

MIDDLEWARE sits on the local operating systems and tend to give support to:

Hide component and resource physical distribution

To make transparent the split of the application in different parts executing on different machines

Hide heterogeneity

To make transparent the distribution of the application over different hardware, different operating systems, different protocols, …

Provide common interfaces

The entire application can be obtained by putting together legacy parts, already available, by sub setting and composing added parts toward the maximum of interoperability

Provide basic services

The application must have available a directory of available functions to avoid duplication and favor collaboration

Grant necessary availability and QoS

MIDDLEWARE typical SERVICES

MIDDLEWARE provide very different services in very differentiated areas to fulfill user needs

Presentation Management (print, graphics, GUI, user interaction)

Computation (common procedures, char services, parallelization, fast access, internationalization, sorting, …)

Information Management (file manager, record manager, database manager, log Manager, …)

Communication (messaging, RPC, message queue, mail, electronic data interchange…)

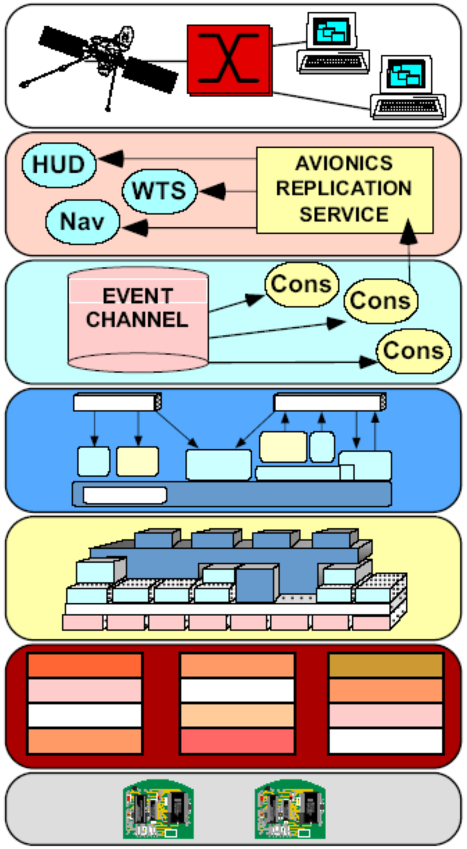
Control (thread manager, scheduler, transaction manager, …)

System Management (accounting, configuration, security, performance, fault management, … event handling)

MIDDLEWARE in a layered view

MIDDLEWARE between

Applications and Operating Systems



Application layer

Domain-specific Middleware Service

Common Middleware Services

Distribution Middleware

Host Infrastructure Middleware

Operating Systems

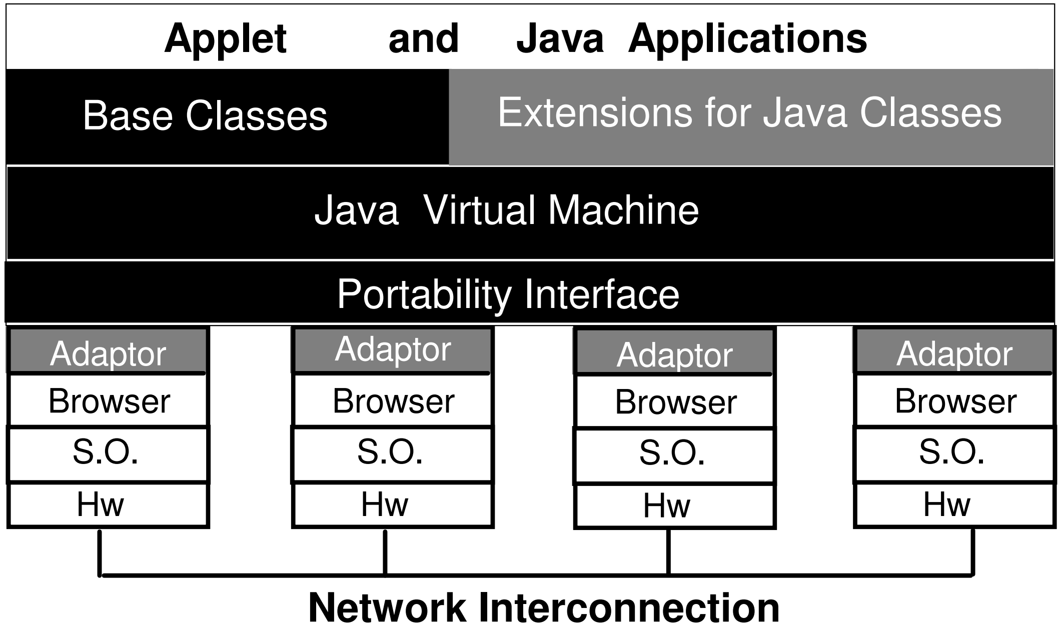
Hardware

MIDDLEWARE layers: Host Infrastructure

Host Infrastructure Middleware

This layer encapsulates and prepares for the common services to support distribution and to ease necessary communication

Examples: JVM, .NET, other local models



Applet and Java Applications

Base Classes

Extensions for Java Classes

Java Virtual Machine

Portability Interface

Adaptor | Browser | S.O. | Hw

Adaptor | Browser | S.O. | Hw

Adaptor | Browser | S.O. | Hw

Adaptor | Browser | S.O. | Hw

Network Interconnection

Some APIs are provided toward a unified support in different environments

MIDDLEWARE layers: Distribution

Distribution Middleware

This layer provides the programming models for distribution and to ease the applications in configuration and management of distributed resources

Examples: RMI, CORBA, DCOM, SOAP, …

Those systems allows an easier **communication and coordination** of all nodes taking part in the system, by introducing a **resource model** and

- some **communication APIs** , by proposing and enforcing a new conceptual model
- other basic functions for **communication, name support, discovery**, **fast storage and access**, **parallel processing**, …

MIDDLEWARE layers: Common Services

Common Middleware Services

Added-value services, typically higher-level to facilitate the duties of the designer and to enforce a component-oriented perspective fully supported and aided

Examples: CORBA Services, J2EE, .NET Web Services

Some additional functions are inspired to a common architecture idea and to a unified support model

Several additional services are available in terms of components that you can add at any time depending on your needs

events, logging, streaming, security, fault tolerance, …

MIDDLEWARE layers: Specific Services

Domain-specific Middleware Service

A set of application tools ad services grouped according to specific domains

Examples: Some task force are defining **ad-hoc functions for different areas** with very tailored goals, depending on the needs of specific groups, but always defining standards
Task Force within OMG (**Object Management Group**)
    Electronic Commerce TF,
    Finance (banking and insurance) TF,
    Life Science Research Domain TF,

Syngo Siemens Medical Engineering Group,

Boeing Bold Stroke within CORBA (flight and flight transport),

…

CLASSIFICATION of MIDDLEWARES

MIDDLEWARE

RPC / RMI middleware

Message Oriented Middleware (MOM)

Distributed Transaction Processing (TP) Monitor

Database Middleware

Distributed Object Computing (DOC) Middleware

Adaptive & Reflective Middleware

Other special-purpose middlewares:

Mobile & QoS Multimedia Middleware

Agent-based Middleware

***see:* computingnow.computer.org**

A PIONEER MIDDLEWARE !?

Wide Area Distributed Middleware (Web)

A global Middleware to allow easy reading actions to distributed data and also writing operation, in accessing to a global set of information

The web put together a global ‘transparent’ system with

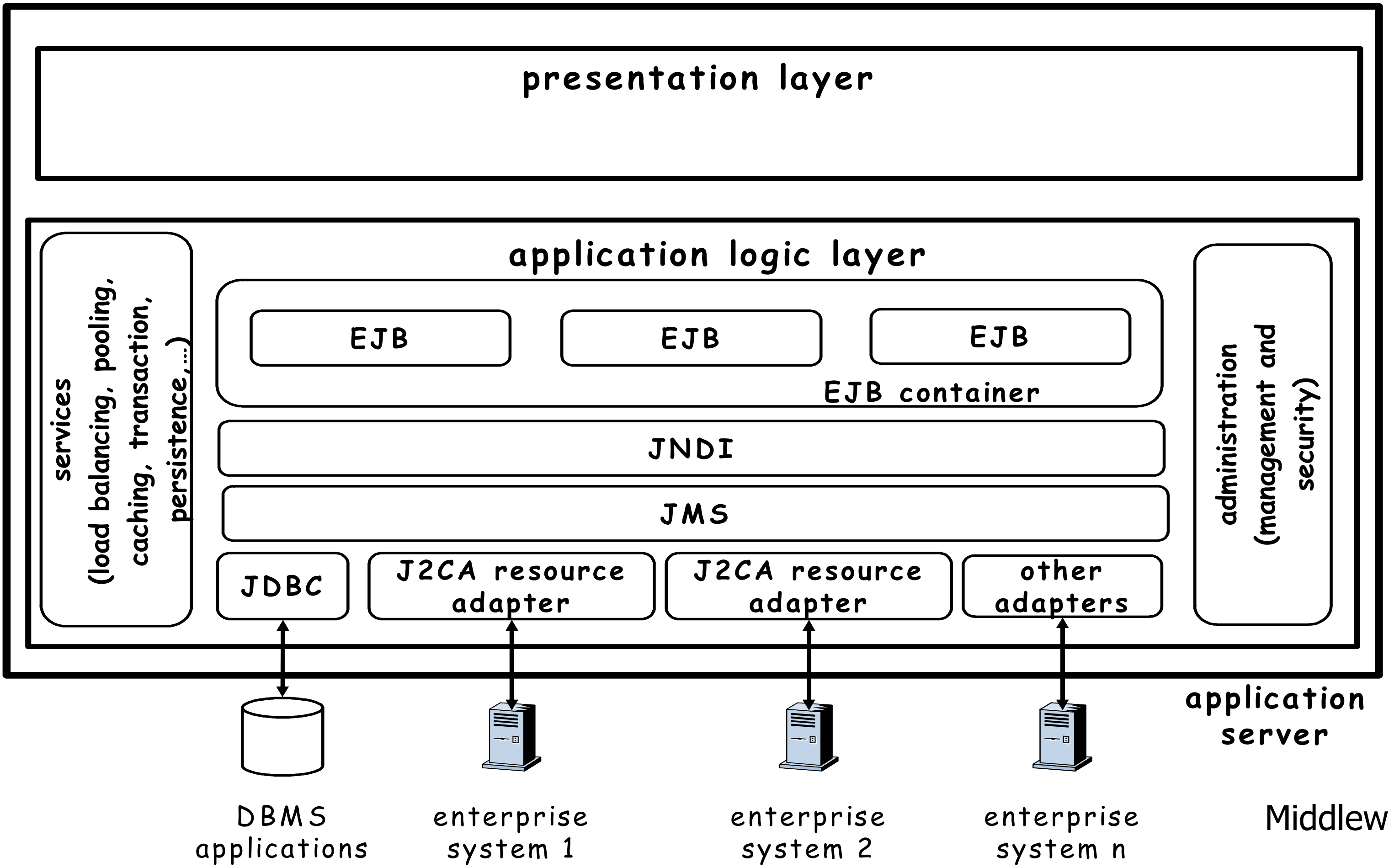
- an enormous number of administrative domains
- an enormous number of users
- an enormous number of host and machines
- an enormous heterogeneity of bandwidth, connections, **…**

The Web it is not a real middleware even if the most widespread and very legacy now

Web as a core example because of its extreme diffusion

WEB MIDDLEWARE !? and J2EE

Java-oriented vision **J2EE**: **J**ava **N**aming & **D**irectory **I**nterface, Java **M**essage **S**ervice, **J2**EE **C**onnector **A**rchitecture

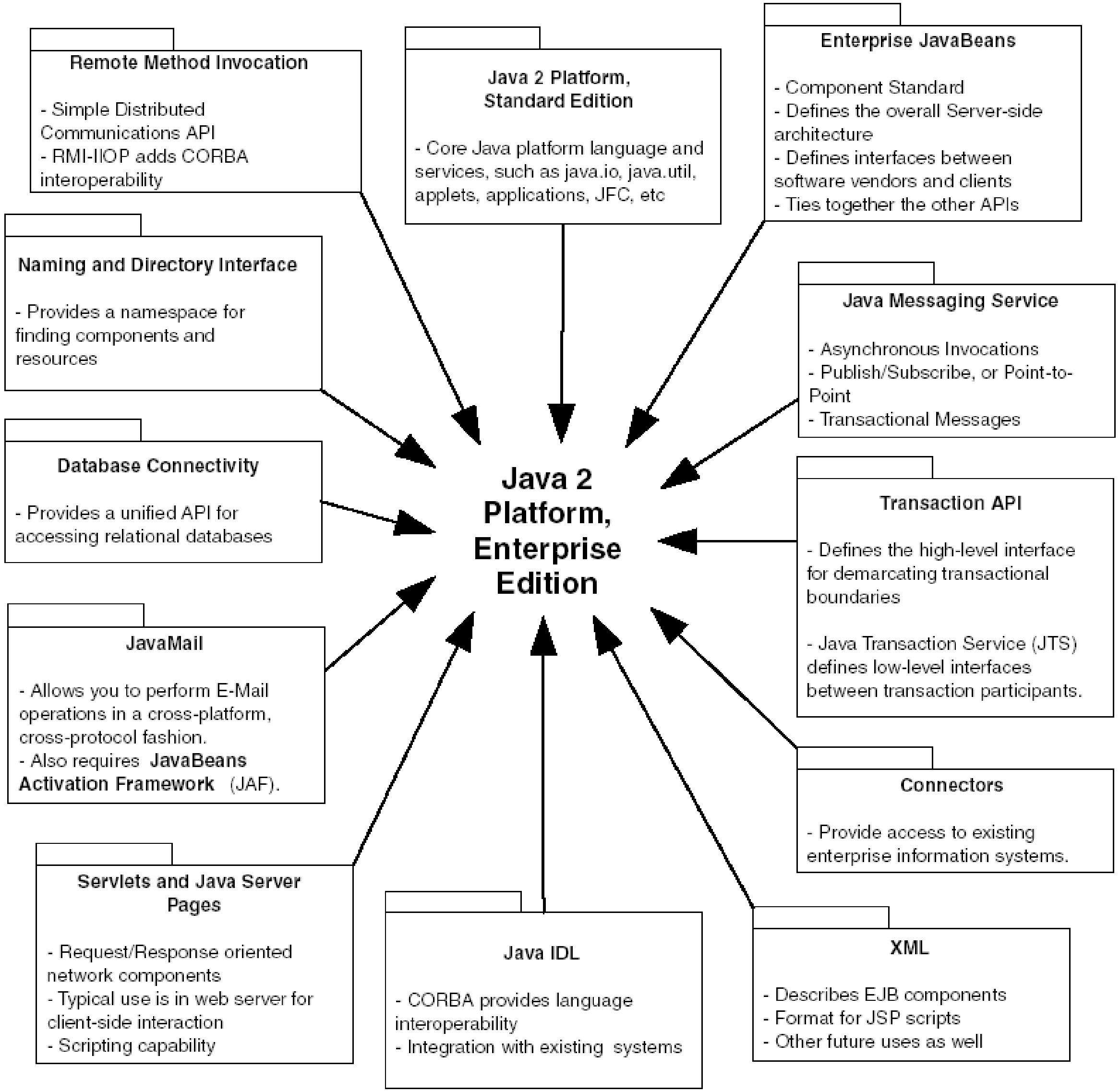


J2EE as an integration of modules

Java2 Enterprise

As a set of components

- databases and more
- naming systems
- components (Beans)
- integration of components via XML
- messaging systems
- communication
- JSP & servlet
- …



Remote Method Invocation

- Simple Distributed Communications API
- RMI-IIOP adds CORBA interoperability

Java 2 Platform, Standard Edition

- Core Java platform language and services, such as java.io, java.util, applets, applications, JFC, etc

Enterprise JavaBeans

- Component Standard
- Defines the overall Server-side architecture
- Defines interfaces between software vendors and clients
- Ties together the other APIs

Naming and Directory Interface

- Provides a namespace for finding components and resources

Java Messaging Service

- Asynchronous Invocations
- Publish/Subscribe, or Point-to-Point
- Transactional Messages

Database Connectivity

- Provides a unified API for accessing relational databases

Java 2 Platform, Enterprise Edition

Transaction API

- Defines the high-level interface for demarcating transactional boundaries

- Java Transaction Service (JTS) defines low-level interfaces between transaction participants.

JavaMail

- Allows you to perform E-Mail operations in a cross-platform, cross-protocol fashion.
- Also requires **JavaBeans Activation Framework** (JAF).

Connectors

- Provide access to existing enterprise information systems.

Servlets and Java Server Pages

- Request/Response oriented network components
- Typical use is in web server for client-side interaction
- Scripting capability

Java IDL

- CORBA provides language interoperability
- Integration with existing systems

XML

- Describes EJB components
- Format for JSP scripts
- Other future uses as well

RPC MIDDLEWARE (…RMI)

Remote Procedure Call as C/S tools

- **I**nterface **D**efinition **L**anguage (IDL) to define the contract
- **Synchronicity**: the client is blocked synchronously while waiting for the answer (result) from Server
- **Heterogeneous** data handling
- **Stub** as envelopes to achieve Transparency
- **Binding** often static (and not so dynamic)

The RPC Model is too **rigid, not scalable & replicable** with **QoS**

The server design must be explicit and any activity provisioning must explicitly defined

No optimization easy to grant shared and private resources

Not so flexible, with the growing extensions of services

RMI definitely in-the-small; RPC evolves into in-the-large

MOM MIDDLEWARE

Message Oriented Middleware (MOM)

Data and code distribution via **message exchange** between **logically separated entities**

Typed & **un-typed message exchange** with **ad-hoc tools** both **synchronous** and **asynchronous**

- **wide autonomy** between components
- **asynchronous** and **persistency** actions
- **handler (broker)** with different strategies and QoS
- easy in **multicast, broadcast, publish** / **subscribe**

Example: Middleware based on messages and queues
MQSeries IBM, **MSMQ Microsoft**, **JMS** SUN, **DDS**, **MQTT**, **RabbitMQ**, **Active MQ**, …

OO e DOC MIDDLEWARE

Distributed Object Computing (DOC) Middlewares

Data and code distribution via **operation requests and replies** between **clients and remote servers**

DOCs use objects within a **framework** and a **broker** as an intermediary for operation object handling

- **the object model** simplifies design
- **the broker** provides both base services and additional ones
- **some operations** can be completely **automated**
- system **integration** is **easier and effective**
- **open source technology** is usually adopted

Examples: **CORBA, COM e .NET, Java Enterprise**

DTP MIDDLEWARE

Distributed Transaction Processing (TP-) Monitor

Middleware to declare and support **distributed transactions**

TP monitors optimize database connections hiding applicative decisions to obtain coordinated and transactional access to data

- specialized interface **for queries** by **lightweight clients**
- **Standardized** actions and **ad-hoc languages**
- **multi-level applications** adopting flexible RPC (various configurations beyond only-synchronous semantics)
- ease in providing **A**tomicity, **C**onsistency, **I**solation, and **D**urability (**ACID**) guarantees
- efficiency in **addressed applicative area**

Examples: **CICS** (**IBM**), **Lotus Notes**, **Tuxedo** (**BEA**), **…**

DB MIDDLEWARE

Database Middleware

Middleware for **integration** and eased usage of **information stored in heterogeneous and different DBs**

to hide implementation-specific details and use standard interfaces

Open DataBase Connectivity ODBC standard

- **without** requiring to modify **existing DBs**
- **efficient support actions** (not much emphasis on optimizations and transactions) in terms of **data access**
- **only synchronous and standard operations**
- evolutions toward **data mining**

Examples: **Oracle Glue, OLE-DB Microsoft**

MIDDLEWARE – more and more…

Adaptive & Reflective Middleware

Middleware able to (self-)adapt to the **specific application** also in a **dynamic, reactive and radical way**

*In some cases the **visibility of underlying levels** can become crucial **to reach optimization***

- **Static variations**, typically **component-dependent**
- **Dynamic variations,** typically **system-dependent**

Via **reflection**, **action policies are expressed and visible** in the **middleware itself** and can change as **system components**

Obtains **adaptation and flexibility at execution time**

Examples: **not so widely diffused yet**

SPECIALIZED MIDDLEWARES

Middleware to support mobility

components designed to ease **transparent allocation and re-allocation** (working cross-layer, from network to application layers)

Middleware to support enterprise interactions

components designed to overcome issues typical **of enterprise services** and tackling **specific business models**

Middleware to support Real-Time (RT) applications

components designed to **guarantee response times** and **deadlines for the development of services in the RT area**

Middleware to support ad-hoc networking

lightweight components and algorithms designed for environments with **limited resources** and **consumption capacities**

AUTONOMIC COMPUTING

Complexity of computing systems makes really difficult to plan interventions and with which precision, whose involvement, …

Model such systems as human bodies where the system is capable of taking care of itself

Complex systems must organize themselves as entities capable of self-managing and self-administration

Also termed **self-*** properties (related to computer agents)

- self-configuration (**autonomy**)
- self-optimization (**social ability and cooperation**)
- self-healing (**reactivity**)
- self-protection (**proactiveness**)

ENTERPRISE MIDDLEWARE

MIDDLEWARE to provide business services

Enterprise Application Integration (EAI)

Need to ease the **integration** across **existing enterprise tools** and their expanded applicability and availability in the enterprise

EAI as environment for **fast and accurate integration** of applications and existing legacy subsystems

Also interfaces for **rapid prototyping** of new aggregates/mash-ups

Systems to grant **easier enterprise workflows (in heterogeneous envs)**

enterprise management (administration and management actions: SAP)

IT and resource management (development functions and application support: Websphere, Oracle)

Service Oriented Architecture (SOA)

MIDDLEWARE DIFFUSION

The middleware market is still expanding in quantitative terms, according to Gartner

World increase of **16.4 %** from 2005 to 2006; 2008 of **6.9 %**, in 2009 of **2,8** %

In 2010, an increase of **7.3 %**, up to **17.6** billions dollars

In 2011, an increase of **9.9 %,** up to **19.3** billions dollars

In 2012, an increase of **5.2 %,** up to **20.3** billions dollars

In 2013, an increase of **7.8 %,** up to **21.9** billions dollars

In 2014, an increase of **8.7 %,** up to **23.8** billions dollars

In 2015, an increase of **0.1 %,** up to **23.95** billions dollars

In 2016, 25,5; in 2017, 27,3 billions

The breakdown of sales percentages:

Company	2006%	2007%	2008%	2009%	2010%	2011%	2012%	2013%	2014%	2015%
IBM	31.8	28.8	30.8	31.5	32.6	32.1	30.9	29.8	29.1	25.2
BEA Systems	10.5	9.3	2	oracle	oracle	oracle	oracle	oracle	oracle	oracle
Oracle	8.6	8.5	13.6	16.7	17.2	16.8	16.1	14.8	13.8	13.3
Microsoft	4.2	3	3.6	3.9	4	5	5	5.1	4.9	4.7
Software AG	2.4	2.2	2.9	3	2.8	3.3	3.2	2.7	2.3	1.9
Tibco	3.5	3	2.9	2.6	2.5	2.9	2.6	2.5	2.4	2.5
SAP						1.2	2.7	2.4	2.2	2.6
Salesforce						0.7	1.6	2.2	3.1	4.2
Altri	39	45.2	44.2	42.4	40.9	38	37.9	40.5	42.2	45.6
totale	100	100	100	100	100	100	100	100	100	100

MIDDLEWARE FORECAST

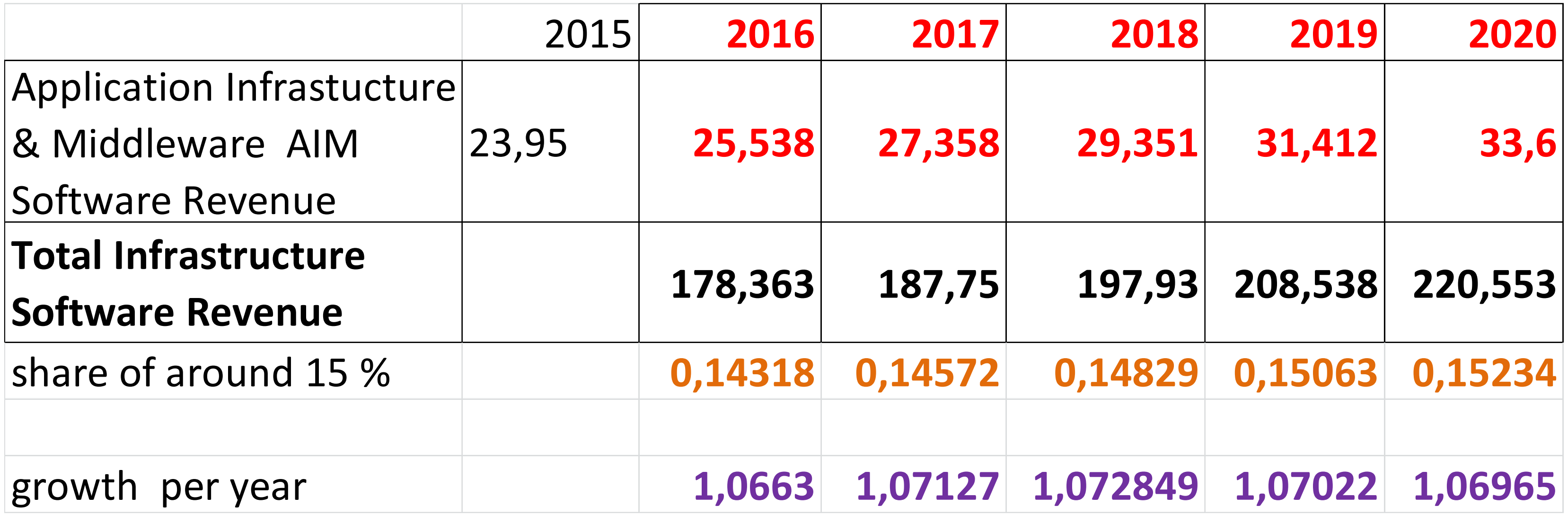
According to Gartner

The Application Infrastructure and Middleware (AIM) market is still growing by keeping a high share of the software market

In 2016, 25,5 billions compared with the software total revenue of 178,4

In 2017, 27,3 billions compared with the software total revenue of 188,3

The forecast after the crisis (in billion dollars)



	2015	2016	2017	2018	2019	2020
Application Infrastructure & Middleware AIM Software Revenue	23,95	25,538	27,358	29,351	31,412	33,6
Total Infrastructure Software Revenue		**178,363**	**187,75**	**197,93**	**208,538**	**220,553**
share of around 15 %		0,14318	0,14572	0,14829	0,15063	0,15234
growth per year		1,0663	1,07127	1,072849	1,07022	1,06965

MIDDLEWARE STILL GROWS

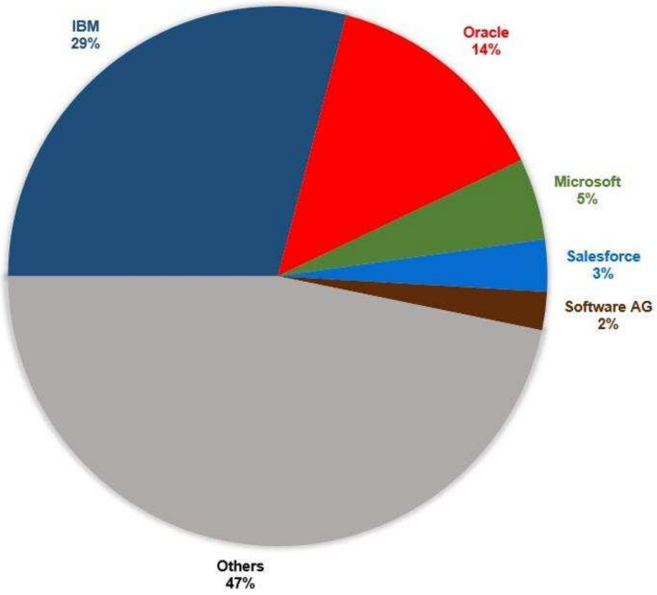
Still growing, according to **Gartner**

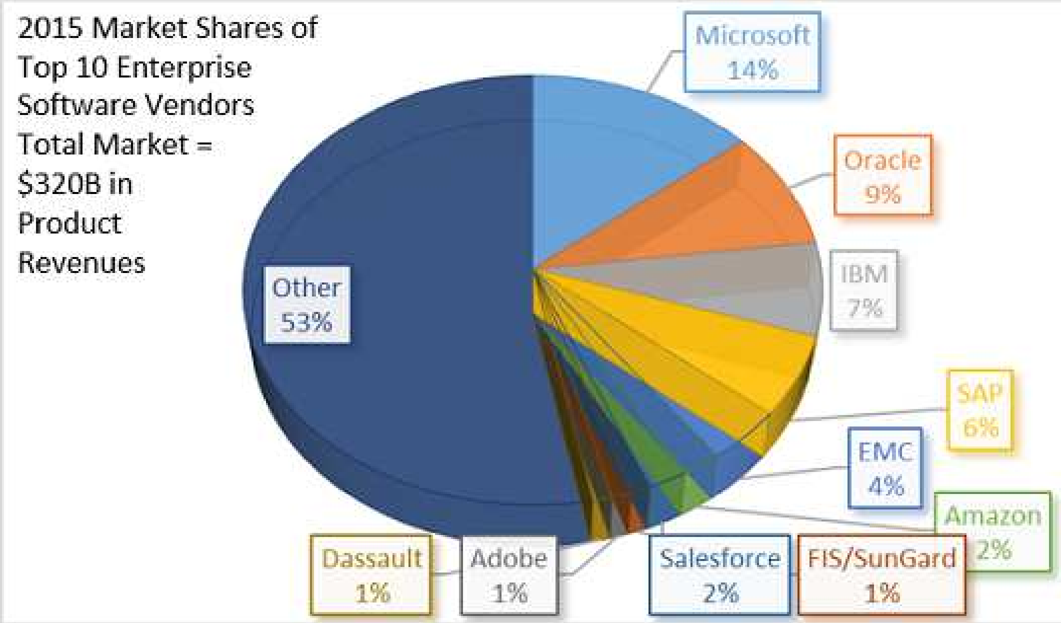
The segment that grows is the non traditional one, from PaaS:

- **PaaS services** and its integration capacity (Integration PaaS o iPaaS) 55 % growth
- **Application** (Application PaaS o aPaaS) 40 % growth

And those are real percentages

Worldwide Vendor Revenue for Total Application, Infrastructure and Middleware (AIM) Software, 2014 (Millions of Dollars) Source: Gartner





MIDDLEWARE DESIGN – ISSUES

The **design of middlewares** tend to consider more and more some **critical** factors that descend from the intrinsic complexity of possible solutions

The first issue is the **increasing set of functions** (objects, resources, etc.) that make **scalability** a very tough problem

- Middlewares tend to introduce **indirect and dynamic mechanisms (interception)** to enable management, introducing an **overhead that is unfortunately high and to be minimized**

- Middlewares tend to introduce **management costs** that require **increasingly sophisticated tools to be continuously adjusted and updated** (**monitoring, accounting, security, control**, etc.)

- Middleware include **mobile and dynamic mobile devices,** with need for **continuous adaptation to the current context and situation**

Middleware usage SCENARIOS

Middleware propose an **architectural model** and **tools**, according to a **vision of precise use**: in this way, middlewares are suitable for very different situations of use and imply an **applicative exploitation** and **recommended use** by those who adopt them

Indeed, it is possible to think of general use cases, but there is always a but there is always a clear idea of who will be users and their requirements

- Middleware developed for **a specific application and that has to work in a precise, inflexible and isolated way: low cost and low intrusion requirements**
- Middleware developed for **applications that have to work in synergy with the middleware, in a flexible way: fast integration and eased communication requirements**
- Middleware that **represents the organization** and requires **continuous evaluation** of internal services with multiple **applications and adequate services**: **life time requirement**

MINIMUM COST MIDDLEWARE

A first scenario is that of **minimum cost middleware**, that drives the configuration of **an application**, according to an **internal interaction model**, without dynamic scenarios

- Users require the configuration of the architecture and to obtain the functionality of **an application in a closed way and with no changes** with MW services at **very low intrusion and very low cost**

Disappearing middleware

MOM Middleware are within this category

- It defines an application that involves a number of nodes and that only provides some participants **statically determined** (only hw resources and provided for specific architectural components), with **default interaction**, **rigid** and **non-adjustable** but **optimized** with **very low costs**
  - No need for **services to support dynamicity** as service names or other similar functions
  - No support for possible inputs and/or **dynamic reconfiguration, turn-on and turn-off of resources**

MIDDLEWARE for FAST APPLICATIONS

A second scenario is that of **middleware** for **very streamlined and optimized applications** that require services and get them quickly and efficiently

- Applications can **provide each other services**, middleware uses its functions and those currently available dynamically
- Support for **dynamic management** of resources and **applications** that self-adapt to fit to the current usage situation

Middleware to facilitate integration of applications
Microsoft Middleware are within this category

- It aims at making easy and straightforward the cooperation among currently active applications
  - Middleware **installed on demand** for applications that can **interact in various ways** (DOC) with **other active applications** running at execution time
  - Support the **dynamism and even possible optimized choices implemented automatically or based on user indication**
- **Middleware life time tied to application life cycle**

MIDDLEWARE for CONTINUITY

A third scenario is that of a **middleware** that needs to **extend the lifetime** of service, seeing services as the set of all the features that an organization can make available for **coarse-grained and facilitated applications**

- Applications can also add **services** that become part of the **middleware** and can be used by everyone so also **dynamically**, eventually
- Support for **dynamic management** of resources and applications

Middleware for infinite life cycle

Middleware of CORBA in this category

The middleware **enriches itself by operating** so to enhance its capacities, via enhancing and adapting to current needs

- The middleware is **initially installed** and is also **populated by different applications** (DOC), **enriching through the introduction of new services**, incrementally and seamlessly
- The support allows use of services with varying degrees of **dynamicity** and **possible choices adapted automatically**

Middleware life time maximized (no downtime)

Cloud Computing Problem Space

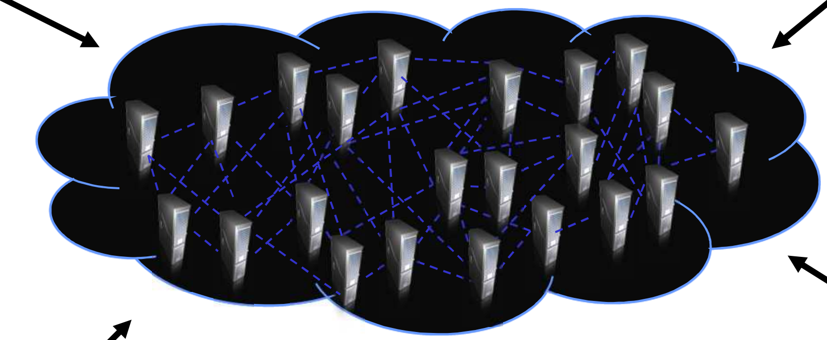
“It starts with the premise that the **data services and architecture** should be on **servers**. We call it **cloud computing** – they should be in a ‘cloud’ somewhere. And that if you have the right kind of **browser** or the right kind of access, it doesn’t matter whether you have a PC or a Mac or a mobile phone or a BlackBerry or what have you – or new devices still to be developed – you can get access to the cloud…”

Dr. Eric Schmidt, Google CEO, August 2006

Explosion of data intensive applications on the Internet

Skyrocketing costs of power, space, maintenance, etc.

Advances in multi-core computer architecture



The Cloud data center

Fast growth of connected mobile devices

SaaS Models

Some increasing resources models for providing some resources as a service, X**aaS**

SaaS Software **as a S**ervice

- Resources are simple **applications available** via remote Web access

PaaS Platform **as a S**ervice

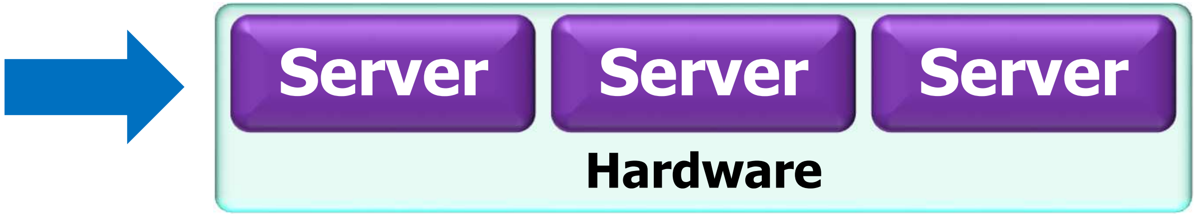
- Resources are **whole software platforms available** for remote execution, i.e., several programs capable of interacting with each other

Iaas Infrastructure **as a S**ervice

- Resources are intended **in a wider and complete way,** from **hardware platforms**, to **operating systems,** to **support** to final **applications**: usually via virtualization up to **Cloud Computing**

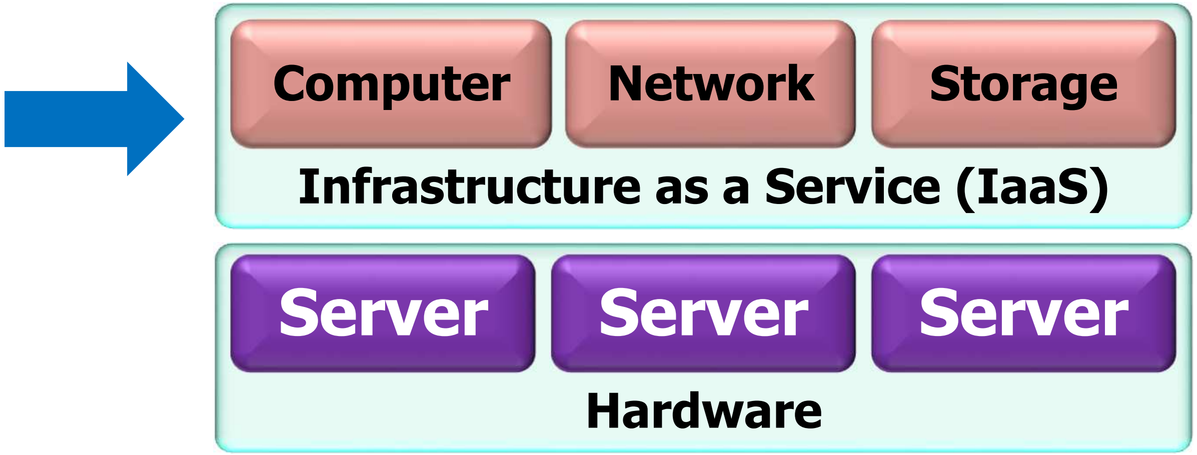
Layered Architecture: IaaS, PaaS & SaaS

- Below the real architecture: **hardware** components & software products



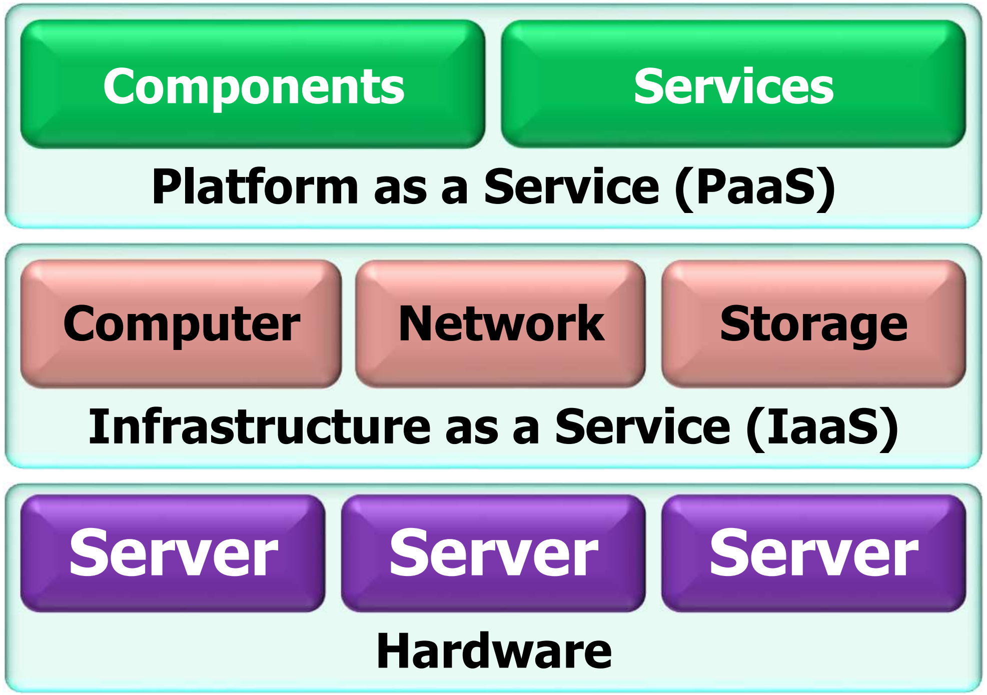
Layered Architecture: IaaS, PaaS & SaaS

- **Infrastructure**: layer to enable the distribution of Cloud services, typically realized by a virtualization platform



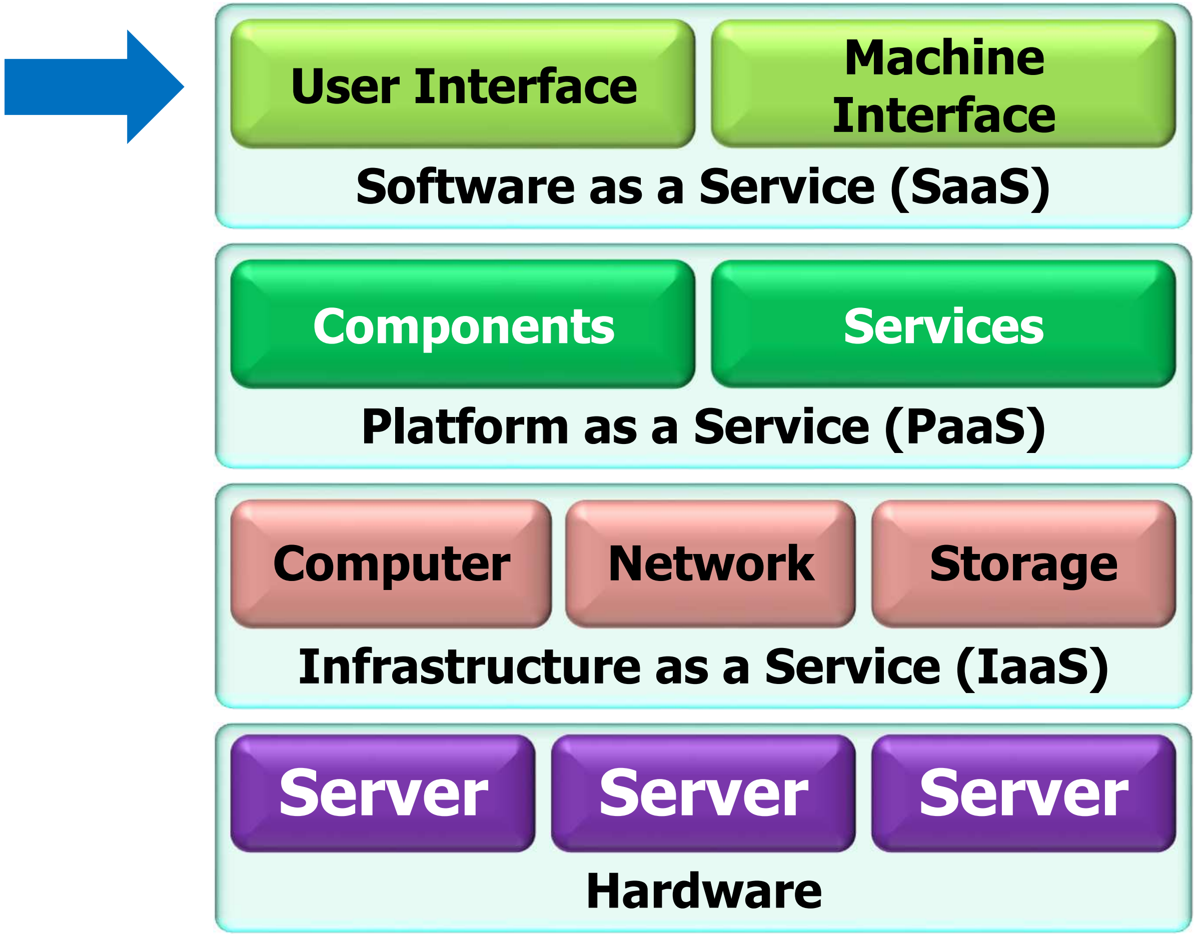
Layered Architecture: IaaS, PaaS & SaaS

- **Platform:** layer to provide to upper layers a set of services and components remotely available



Layered Architecture: IaaS, PaaS & SaaS

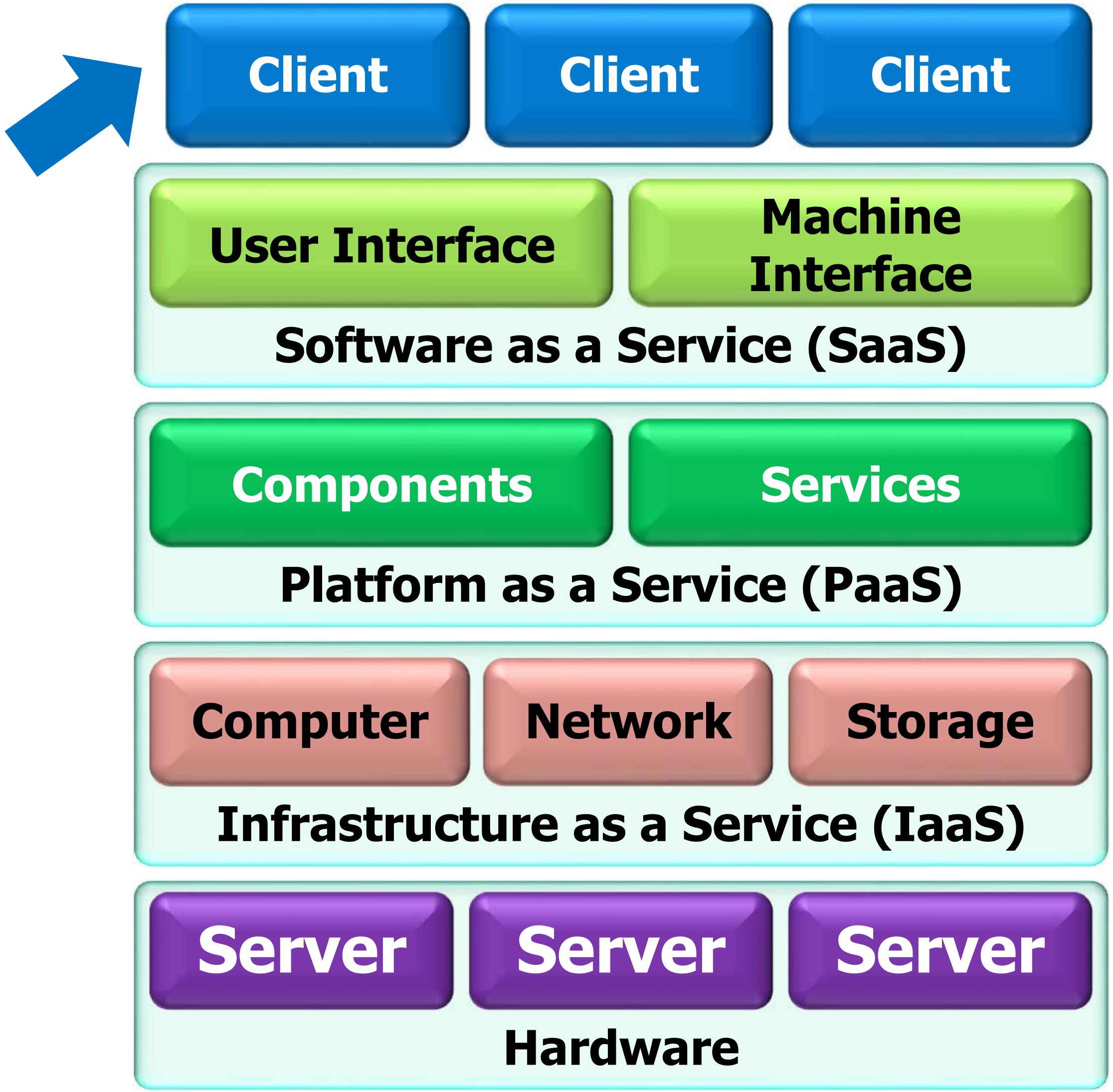
- **Application**: layer to install applications, to be available via Web and Internet via Cloud



Layered Architecture: IaaS, PaaS & SaaS

- **Client** software to get access to the system. Those applications execute on the **client physical platforms** (remote computers) owned by the final remote user

  they can communicate with the Cloud via the **available interfaces**



Client Client Client

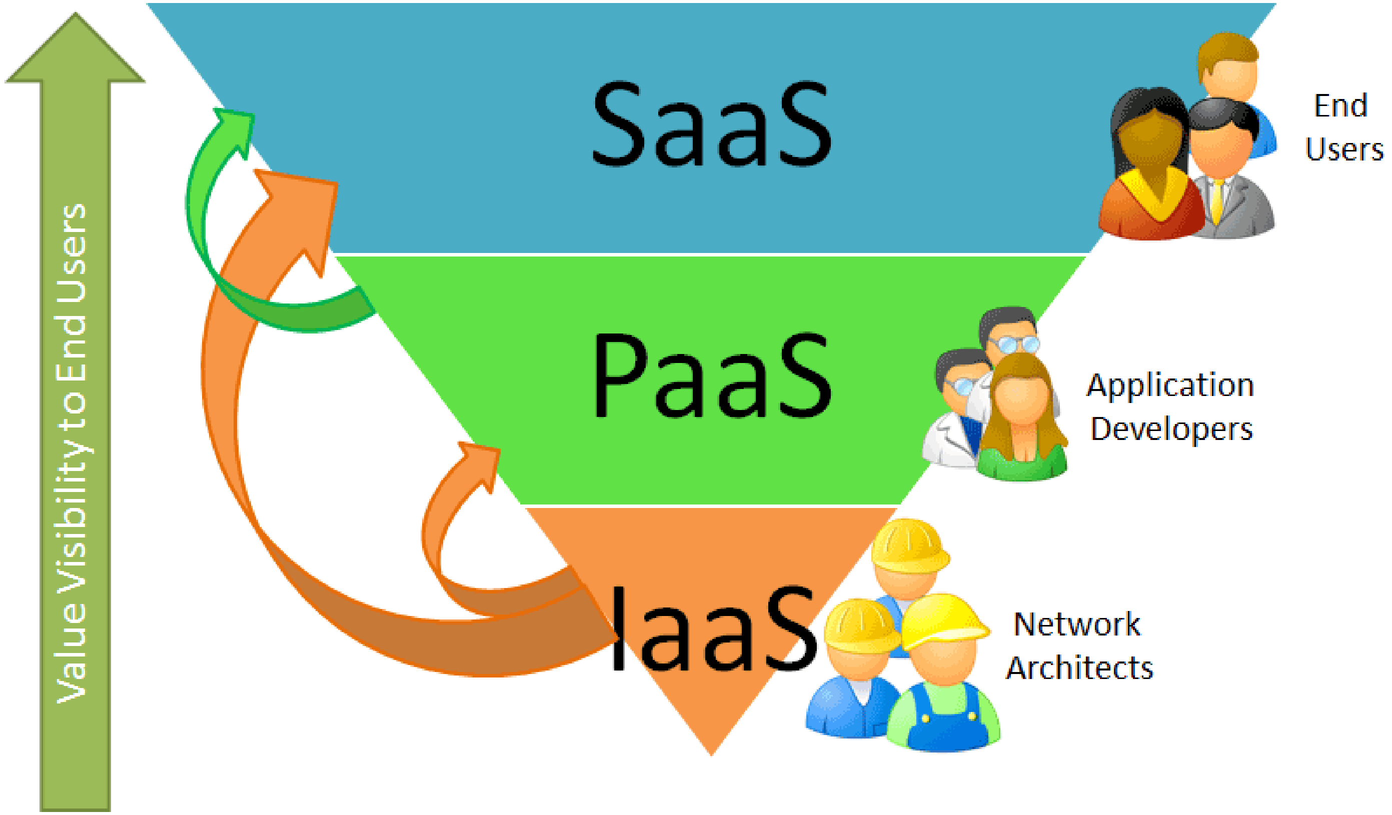
User Interface
Machine Interface
Software as a Service (SaaS)

Components
Services
Platform as a Service (PaaS)

Computer Network Storage
Infrastructure as a Service (IaaS)

Server Server Server
Hardware

Layered Architecture: Actors



Some SaaS and *-aaS examples

SaaS

From desktop applications: **Google Apps** (Gmail, Google calendar & docs), **Microsoft Window live** (Hotmail, Messenger, …) to search engines, Google, Yahoo,
Several **social networks (**Facebook, LinkedIn, Twitter, …)

PaaS typically accessed via Web service

Services available internally to and interacting with other applications, as **Google Maps** in the **G**oogle **A**pplication **E**ngine (**GAE**)

Iaas some experimental infrastructures

Several examples, with virtualization services, **Amazon Web Services** (AWS & S3), **Elastic Computing Cloud** (EC2), to several management and **monitoring desktops** to control execution (Sun global desktop, Zimdesk, …)

MIDDLEWARE for CLOUD (?)

A **CLOUD** solution, from the **user** perspective, is the provisioning of a scenario of **virtualized resources** to obtain in an elastic and fast way resources needed to serve each phase of users request (user 1 - 1 provider)

From provider perspective, need to provide **services (-aaS)**, according to agreed SLA and following two principles:

- **Efficiency** to respond to all users
- **Effectiveness** in carefully using available resources

In general, every provider uses **its resources** and finds the best mapping of configurations and QoS for better services

Scenarios to trend: **many to many**

– Federation between **Cloud providers** to exchange services and resources

– Customers interested not only in having resources of a provider but of **more providers,** but also in **balancing** in accordance with their internal **policies**

– **Cloud as integrator of software resources (full stack or IPaas, Integration Platform as a Service)**

Middleware resource management

Middleware as a **container** and **manager of resources, more and more autonomous,** and **self-handling**

Main managing operations

- **Automatic configuration**
- **System monitoring**
- **Context management**
- **Resource discovery**
- **Resource composition and integration**
- **Resource reconfiguration**

An index of middleware success is its **invisibility**:

the more it disappears, the more the main support goal is met

QoS-related properties

These **non functional properties** are **crucial to solution acceptance** (even necessary on the **long term**)

- **Correctness**
  - ⇨ consistency, stability, timeliness, …
- **Efficiency**
  - ⇨ optimal usage of resources, prompt answer, …
- **Scalability**
  - ⇨ dynamic usage of resources, limited operating costs, …
- **Robustness**
  - ⇨ fault tolerance, replication, availability, reliability, …
- **Security**
  - ⇨ access control, privacy, integrity, …

Cloud Concepts

- IT on demand pricing
- Best benefits in a reliable context
- Pool of virtualized computer resources
- Rapid live providing while demanding
- Systems on scaling architecture

Cloud keywords

on demand,
reliability,
virtualization,
provisioning,
scalability

What is a Cloud

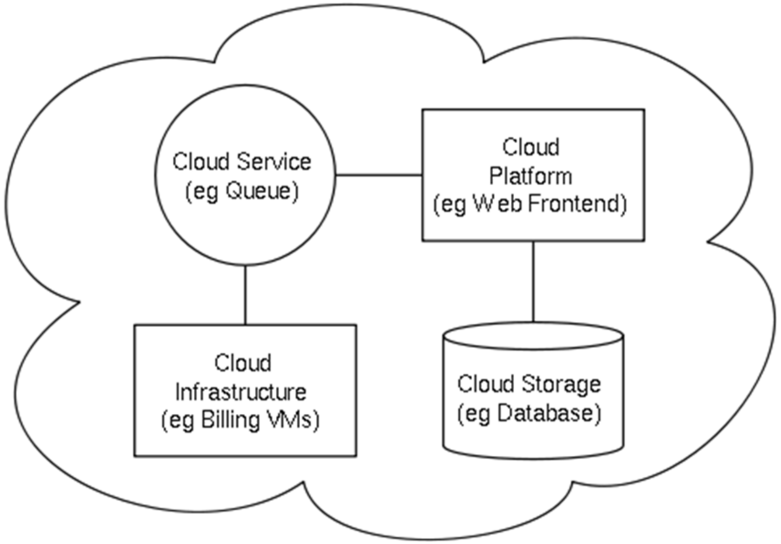
One Cloud is capable of providing IT resources ‘as a service’

One Cloud is an **IT service** delivered to users that have:

- a **user interface** that makes the infrastructure underlying the service transparent to the user
- reduced **incremental management costs** when additional IT resources are added
- **Service-oriented management** architecture
- **massive scalability**

Cloud components

Cloud Computing software systems have a typical **structure based on components** that can communicate with each other via well defined **interfaces** (often Web Services)

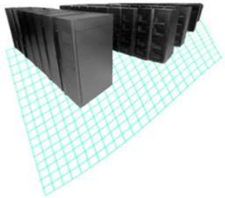

Cloud Service (eg Queue)
Cloud Platform (eg Web Frontend)
Cloud Infrastructure (eg Billing VMs)
Cloud Storage (eg Database)

- **Four main components**:
  - one **Cloud platform**, with an externally available **interface accessed via web** to cooperate with the real or virtual internal infrastructure
  - one **virtualization** **infrastructure** and the management system for the control, monitoring, and billing for **client requests**
  - one internal **memory system** **typically via** a **database**
  - one internal **manager** to handle **external requests** (management, queuing, and controlling)

A bit of history

Grid Computing

- Solving large problems with parallel computing



Utility Computing

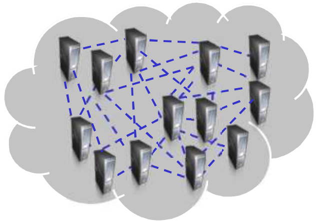
- Offering computing resources as a service

Software as a Service

- Network-based subscriptions to applications

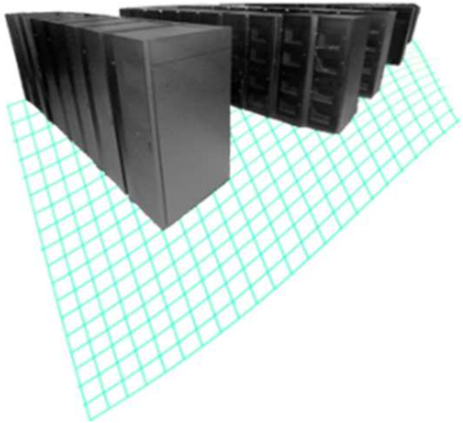
Cloud Computing

- Anytime, anywhere access to IT resources delivered dynamically as a service.



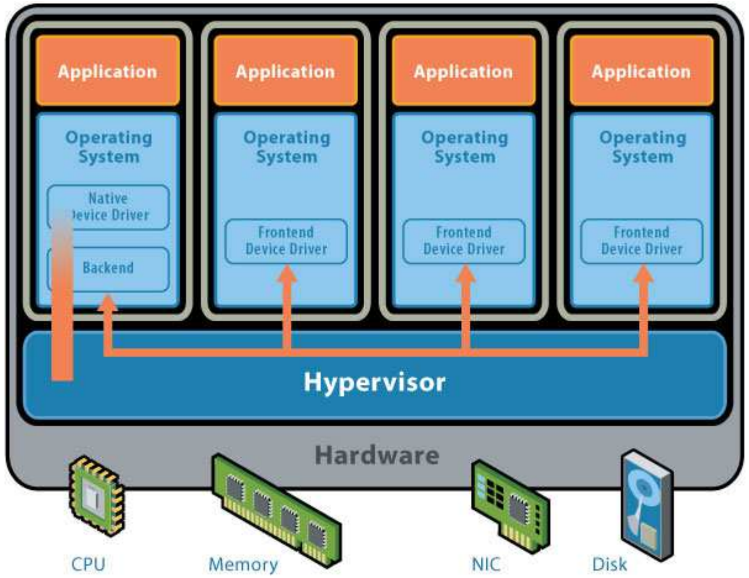
Before Cloud computing: GRID

- **Grid computing**
  - Sharing of **heterogeneous resources** (computer, software, data, memory, computational power,, …) in **highly distributed environments** with the goal of **creating a virtual organization scalable** (***by need!***)
  - Interfaces (for management), often **too fine grained, with low level of abstraction,** and **non self-contained** ☹
  - Application areas very **limited and specific** (parallel computation for scientific, engineering scenarios, …)



Before the Cloud: Virtualization

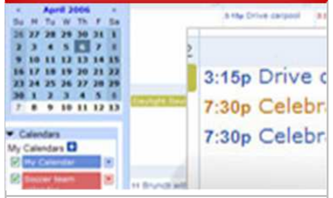
- **Virtualization**
  - Technologies for **virtualization** (either system-based or hosted), as in a server farm**:** Vmware, Xen, …
  - **Isolation** & **personalized infrastructure and/or SW platform** (O.S. and some additional applications)
  - Tool for the **efficient management** of computing infrastructures (IBM Tivoli suite, Xen monitoring tools, …)



Before Cloud computing: Web 2.0

- **Web 2.0**
  - **Usage of asynchronous protocols not visible** to users to ask only **really required info** and not the **whole web pages: Asynchronous Javascript And XML (AJAX)**
  - New ways of **using Web services coupled with new applications easier to use, collaboration based** and **openly available, without requiring any installation** by interested users: new business model, **very, very cooperative** (**S**oftware **as a S**ervice ☺)

Office



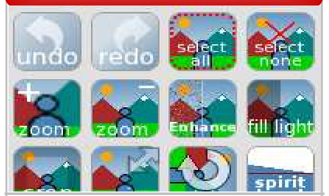
https://www.google.com/hosted
http://smallbusiness.yahoo.com/email/
www.zimbra.com
many others…

Word

Top Email Trends (2005)
- Relevance drives success
- Frequency driven by the individual, r
- UI is critical for usability

www.writely.com
www.writeboard.com
www.inetword.com
many others…

Graphics

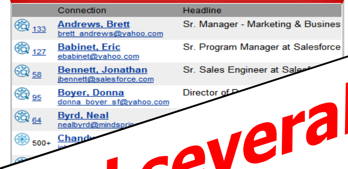


www.pxn8.com
www.pixoh.com
many others…

Database

www.dabbledb.com
www.Lazybase.com
www.quickbase.com
many others…

Contacts



thers…

And several others…

Before Cloud computing: Utility computing

- Huge computational and storage capabilities available from **utilities**, the same as for energy and electricity, and on pay-per-use base.
- "**Computing may someday be organized as a public utility**" - John McCarthy, MIT Centennial in 1961
- **Metered billing** (pay for what you use)
- **Simple to use interface** to access the capability (e.g., plugging into an outlet)

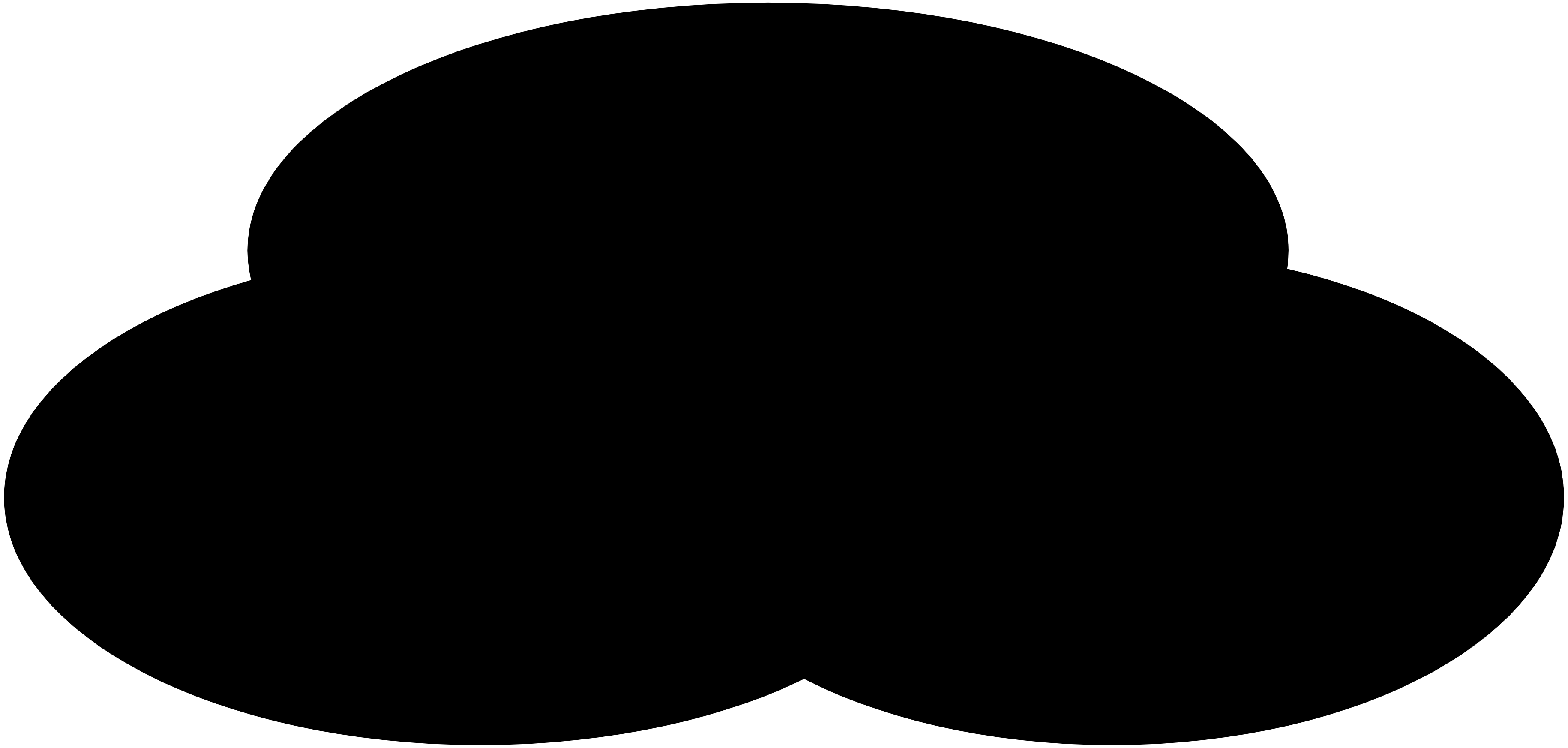
Cloud different from …

- **Grid Computing**
  - A cloud is more than a **collection of computer resources** because a cloud provides a mechanism to manage those resources

  Provisioning, change requests, workload balancing, monitoring

  - Cloud computing is an infrastructure that sits on top of a data centre for efficiency
- **Utility Computing**
  - Service that allows users to **deploy, manage, and scale** online services using the provider's resources and pay for resources they consume
  - Users want to be in control of what runs on each server
  - **Cloud users** want to avoid infrastructure. The provider is in complete control.
- **SaaS**
  - Software that is **owned, delivered, and managed remotely** by one or more providers
  - Software that allows a sharing of application processing and storage resources in a one-to-many environment on a pay-for-use basis, or as a subscription

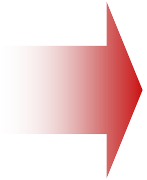
Evolution of Cloud Computing



Software as a Service (SaaS)

Traditional Software

Build Your Own



On-Demand Utility

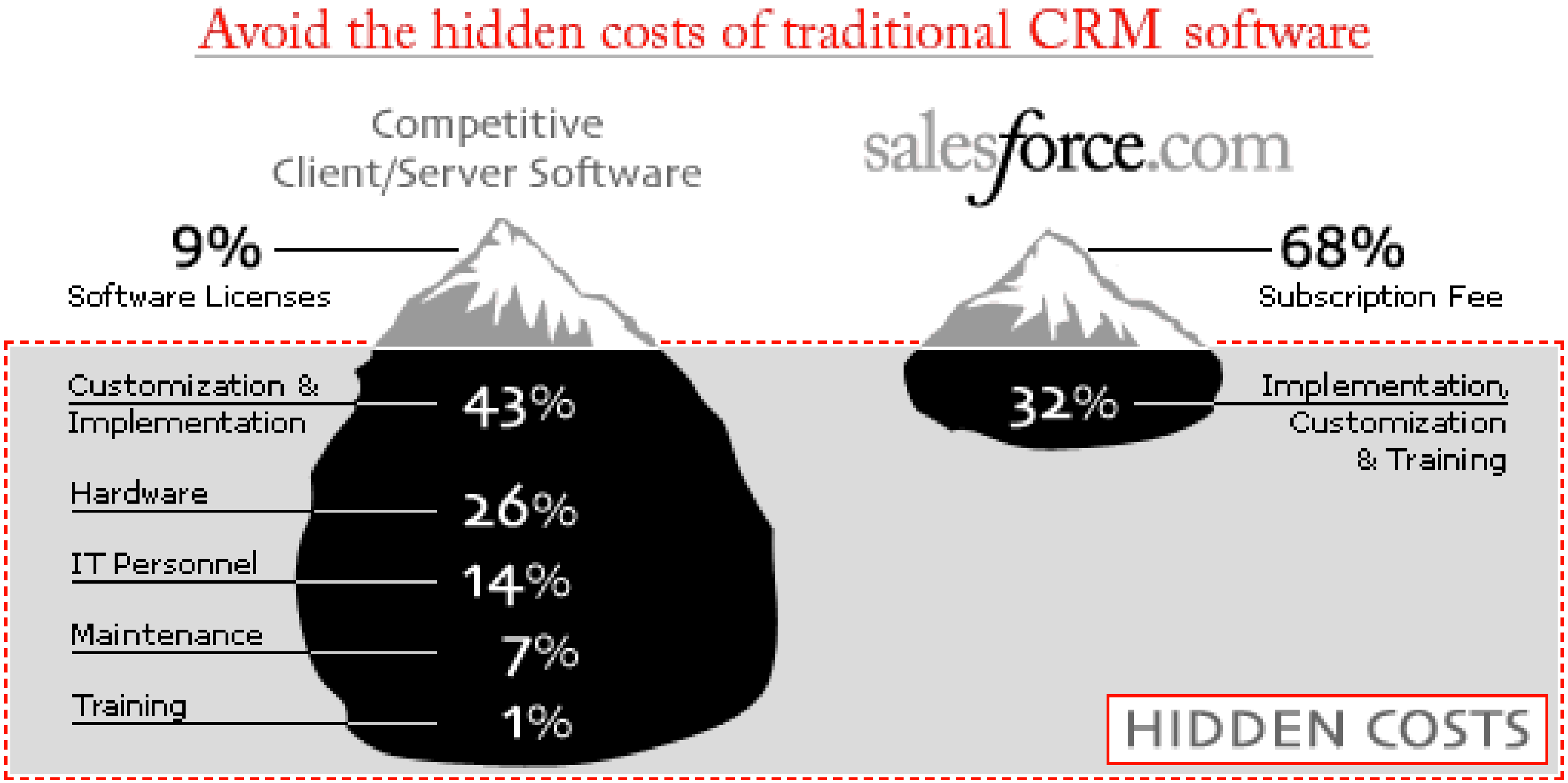
Plug In, Subscribe

Pay-per-Use

SaaS - Software as a Service

- **Software ownership costs pushed to vendor** - hardware, software, system security, disaster recovery, maintenance, monitoring
- **Return to core competency** - organizations shift resources to core competencies, vendors focus on managing their SaaS
- **More efficient deployment** - instant evaluation, more collaboration between vendor and IT organization, much faster deployments
- **Eliminate shelfware & maintenance** - pay for what you use
- **Always on current version** - version-free software means the latest for the customer
- **Modern, Web 2.0 interface** - drive technician usage and better customer interaction with IT
- **SaaS homogeneity costs less** - one version for the vendor to support means lower costs for everybody

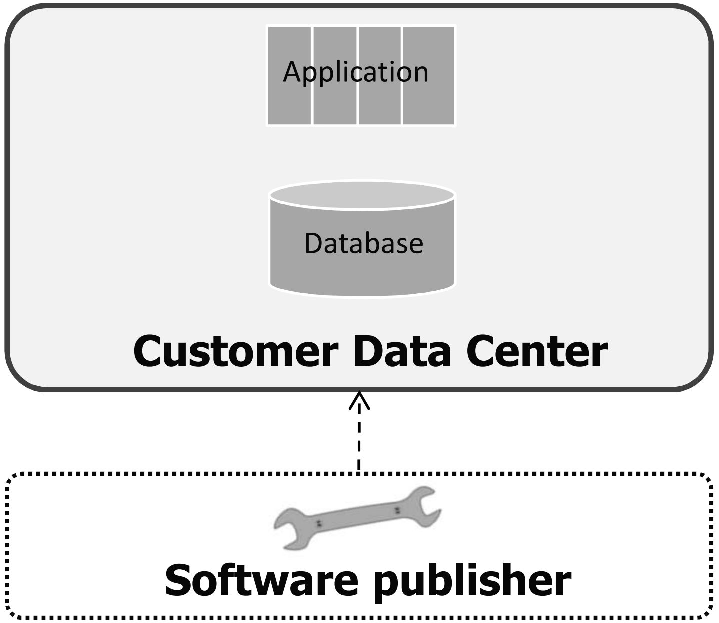
Hidden Cost of IT



Application areas suitable for SaaS

- **ERP vertical business** applications, both specialized and very specific
- **General-purpose** applications without any adaptation (potentially sharable)
  - self-service provisioning and ad-hoc personalization
  - applications available to several different users
- **Business B2B** applications **domain specific**
  - no need of third party hosting and involvement
- **Customer/Supplier** applications
  - applications where most of users and access is externally to the organization and where ubiquitous access via Web is critical and intrinsic
- **Business** applications **even critical**,
  but not the **core business ones**

Traditional on-premise Deployment at the client site



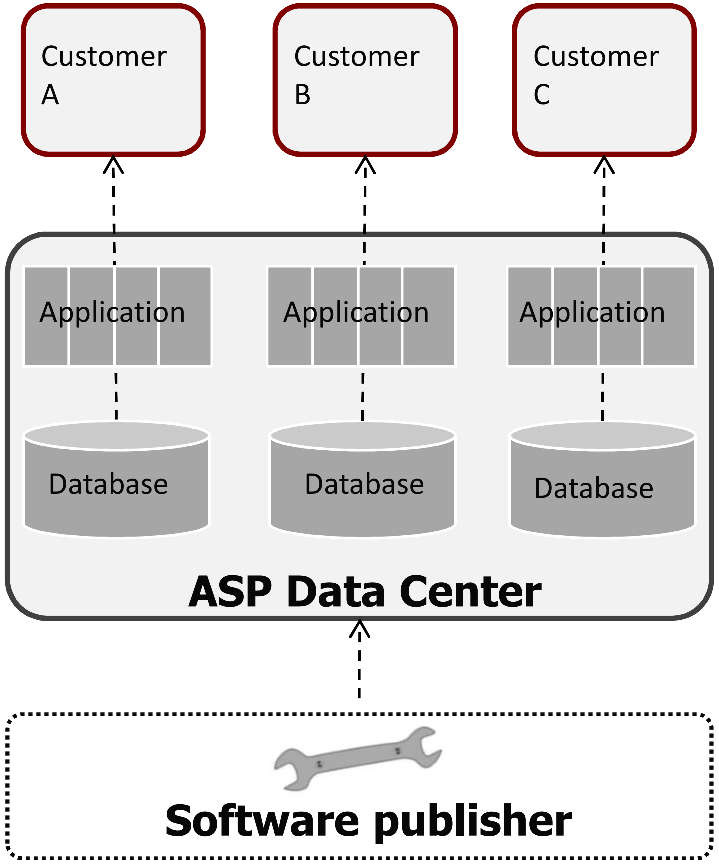
Details

- Full ownership
- Significant implementation
- Customizable
- Difficult to upgrade / maintain

Examples

- HP Service Manager
- BMC Remedy
- CA Service Desk
- EMC Infra

Application Service Provider (ASP)



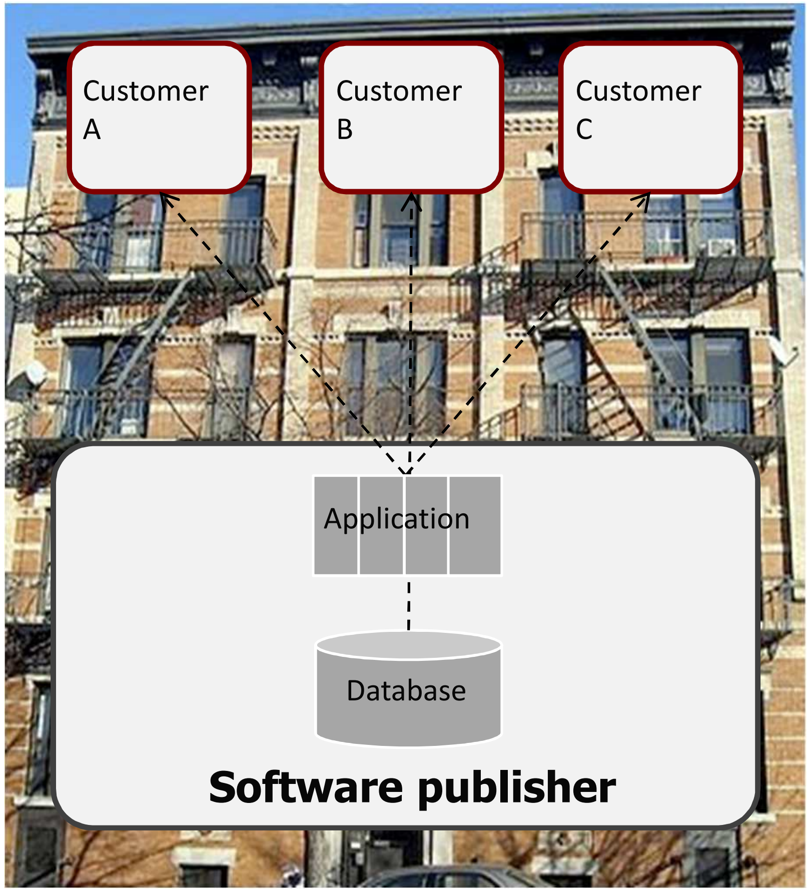
Details

- Procures app and resells service
- Broker between customer and publisher
- Focus on ‘out-of-box’

Examples

- IBM GS
- HP Services
- BMC AAS
- CSC

SaaS multi-tenant



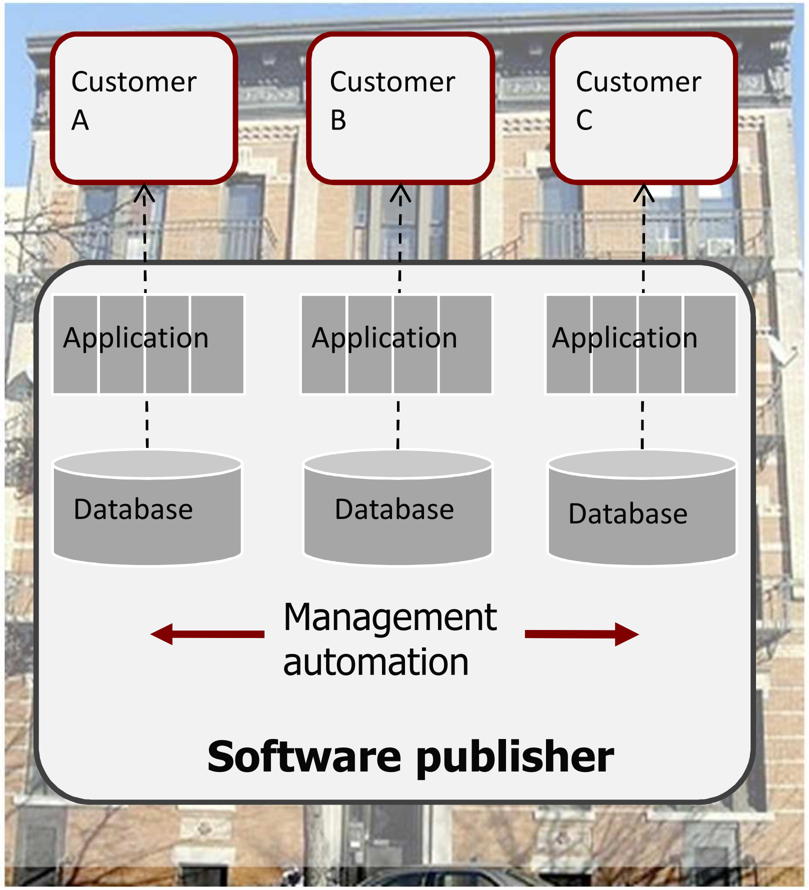
Details

- Hosted by software publisher
- Many customers to one application set
- Thought to be inflexible

Examples

- Salesforce.com
- Workday
- Innotas

SaaS single-tenant



Details

- Hosted by software publisher
- Customers receive their own app and database
- Auto-upgrades
- Extensive customization

Examples

- Service-now.com
- InteQ
- Eloqua

Technology wrap up

SaaS

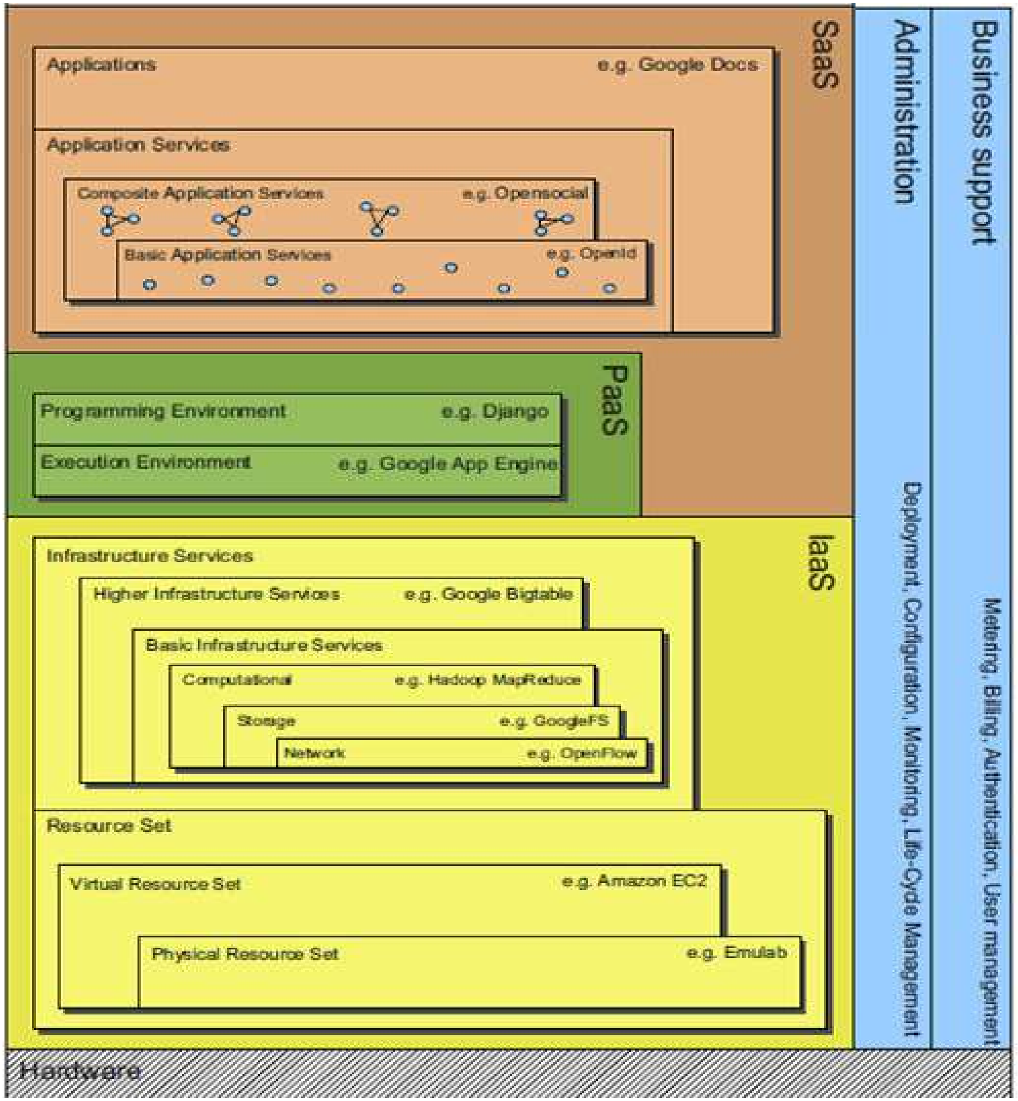
- Google docs

PaaS

- Google App Engine

IaaS

- Google Bigtable
- Hadoop MapReduce (Yahoo)
- GoogleFS
- Amazon EC2



Support functions for all levels

Cloud computing: reality check

- **Amazon Elastic Computing – EC2**: virtualized images (DB+Software and middleware+OS), Xen, simple SLA console
- **Google App Engine** (Software as a Service, web applications, Google App Engine, sandbox for management and security)
- **IBM Blue Cloud**: virtualized images (DB+Software and middleware+OS), Xen, Tivoli (monitoring and management), simple SLA console
- **HP/Yahoo/Intel Test Bed**: virtualized images, Xen, simple SLA console
- **Microsoft Azure**: recently launched by Microsoft
- **Openstack**: standard effort with large and spreading diffusion
- **Research initiatives** (***RESERVOIR EU FP7 project***, previous projects on grid computing such as EEGE, …)

Others ongoing projects: **Eucalyptus**, **3Tera**, …

Cloud Key Goals

Infrastructure Perspective

- How can we provide flexible compute resources quickly to promote **rapid prototyping**?
- How do we deploy applications that **scale up** to meet increasing demands over time?

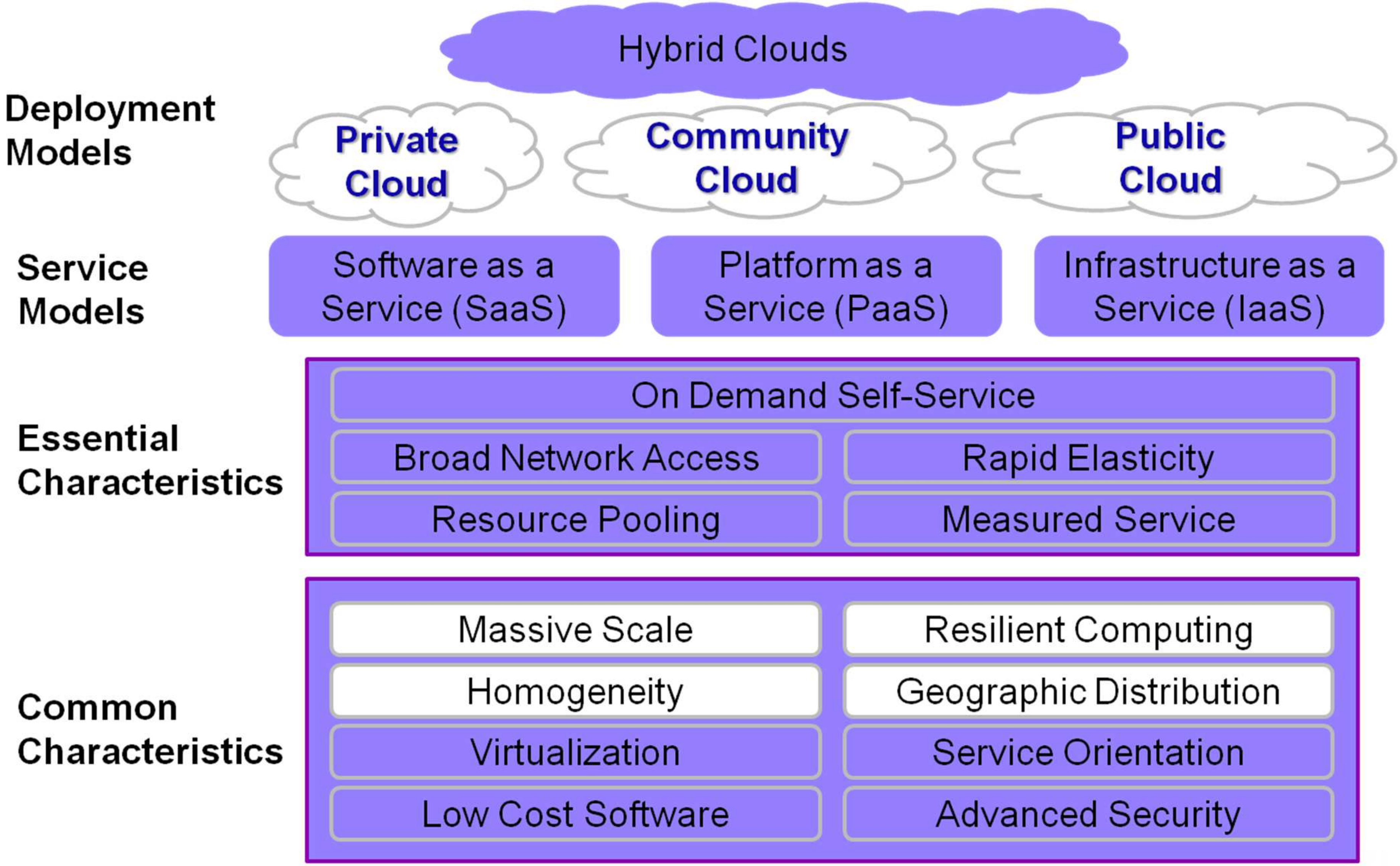
- How do we manage 100,000's of machines with **minimal human intervention**?
- How can we make the most **efficient** use of all the compute resources in a data center?

Cloud Deployment Models

Typically three models

- **Private cloud**
  - enterprise owned or leased
- **Community cloud**
  - shared infrastructure for specific community
- **Public cloud**
  - sold to the public, mega-scale infrastructure
- Hybrid cloud
  - composition of two or more clouds

The NIST Cloud Definition Framework



Hybrid Clouds

Deployment Models

Private Cloud

Community Cloud

Public Cloud

Service Models

Software as a Service (SaaS)

Platform as a Service (PaaS)

Infrastructure as a Service (IaaS)

Essential Characteristics

On Demand Self-Service

Broad Network Access

Rapid Elasticity

Resource Pooling

Measured Service

Common Characteristics

Massive Scale

Resilient Computing

Homogeneity

Geographic Distribution

Virtualization

Service Orientation

Low Cost Software

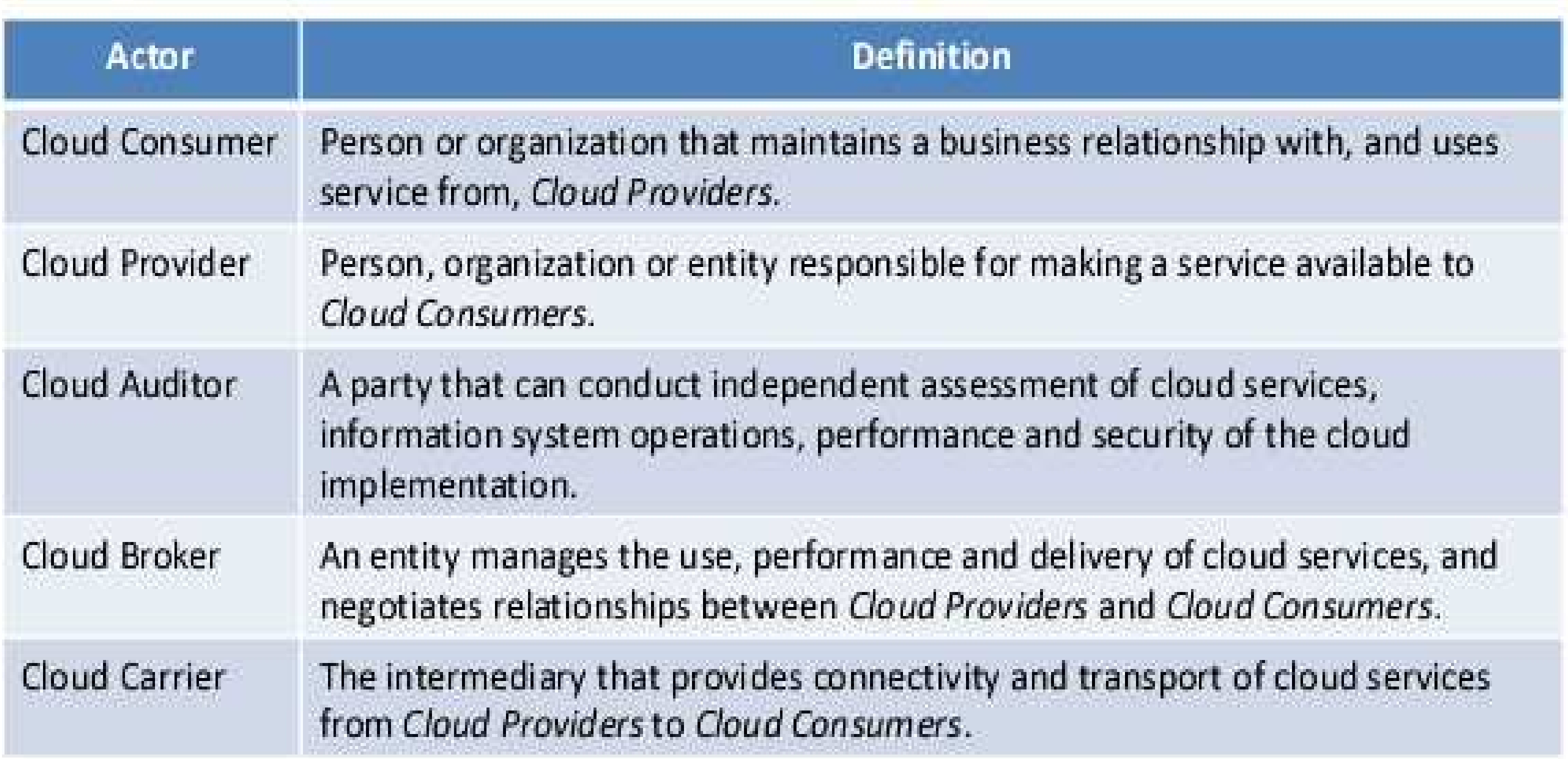
Advanced Security

National Institute of Standard and Technology

New Business models (NIST - March 2011)

New business roles stemming fromCloud

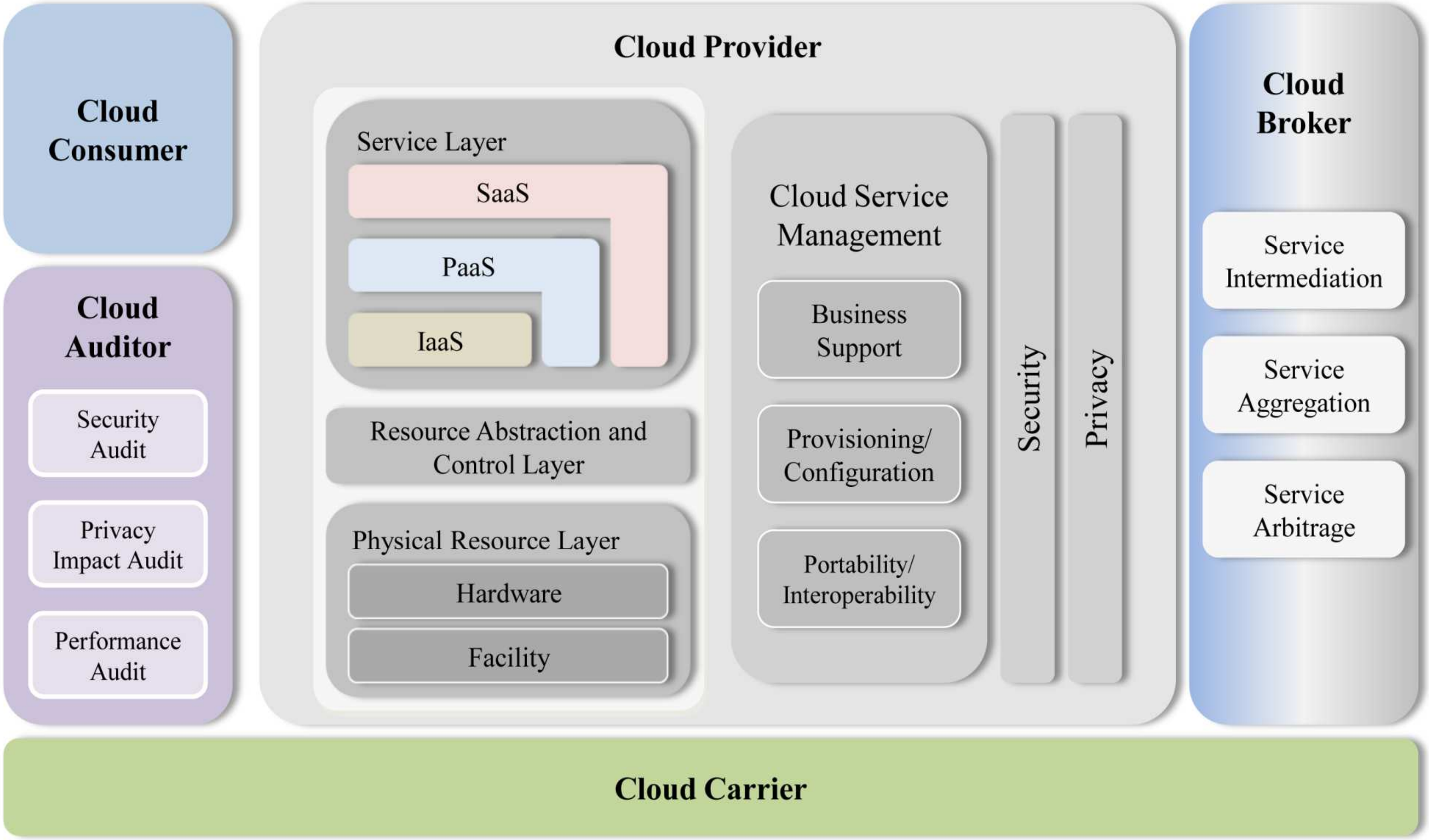
- Consumer, Provider,
- Carrier, Broker, Auditor



Actor	Definition
Cloud Consumer	Person or organization that maintains a business relationship with, and uses service from, *Cloud Providers.*
Cloud Provider	Person, organization or entity responsible for making a service available to *Cloud Consumers.*
Cloud Auditor	A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation.
Cloud Broker	An entity manages the use, performance and delivery of cloud services, and negotiates relationships between *Cloud Providers* and *Cloud Consumers.*
Cloud Carrier	The intermediary that provides connectivity and transport of cloud services from *Cloud Providers* to *Cloud Consumers.*

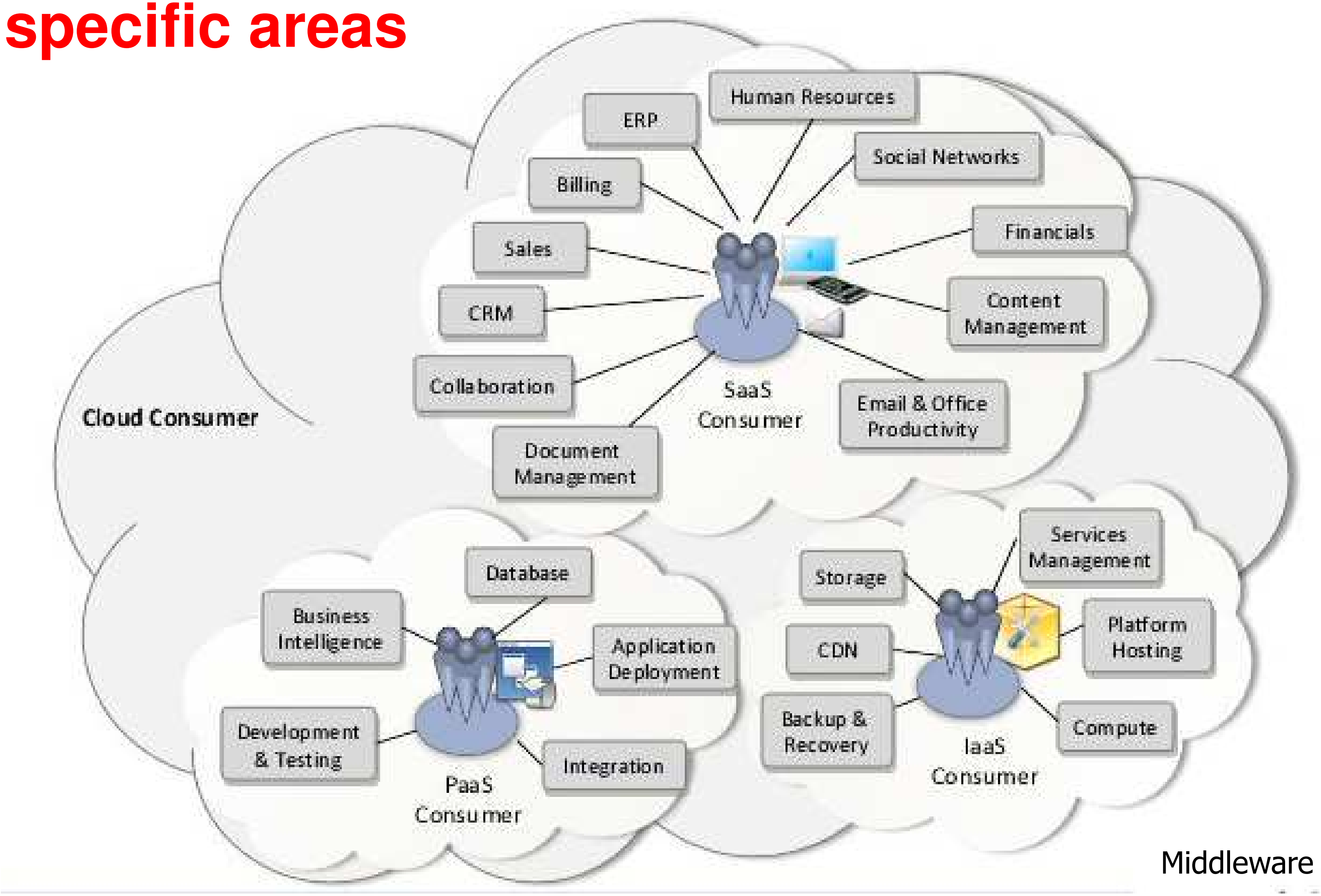
Complete model of service

Some roles and offerings are still not so widely available



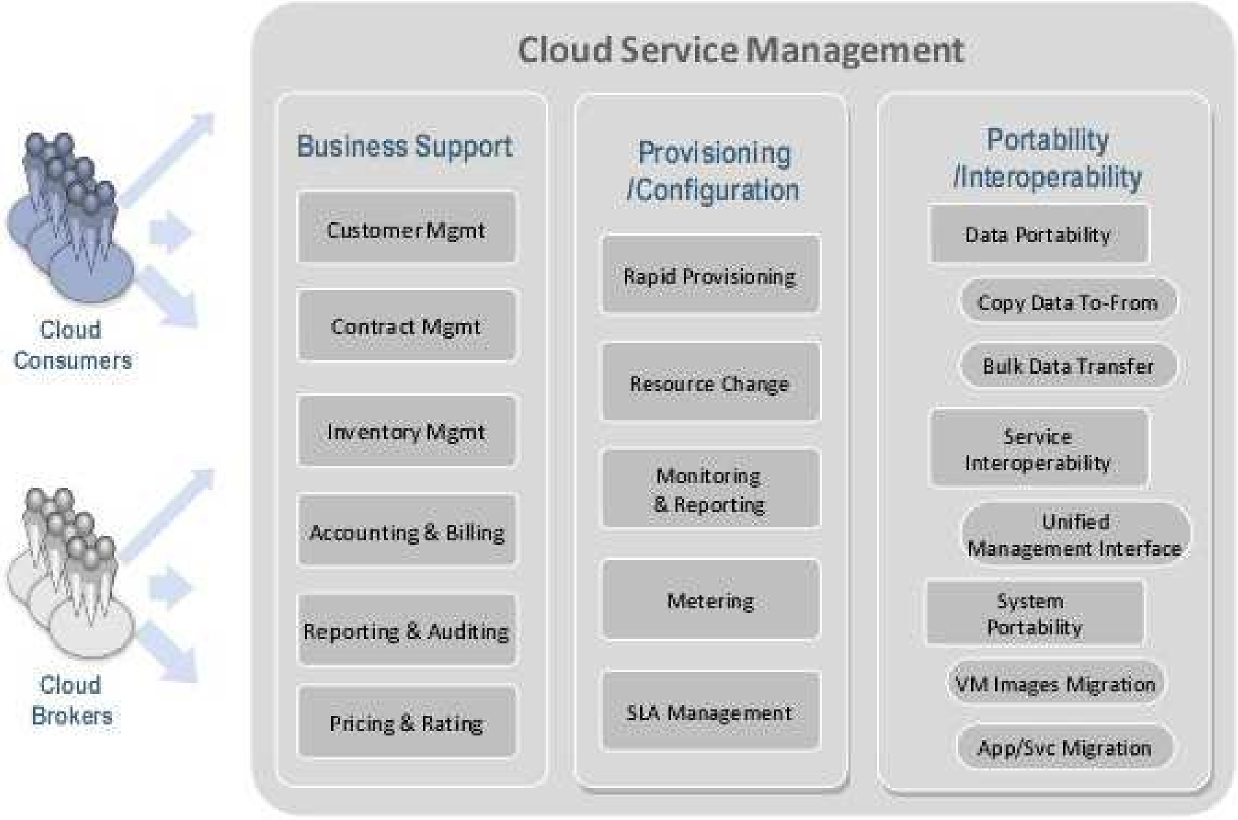
Typical areas of service offering

Clients tend to receive services in some specific areas



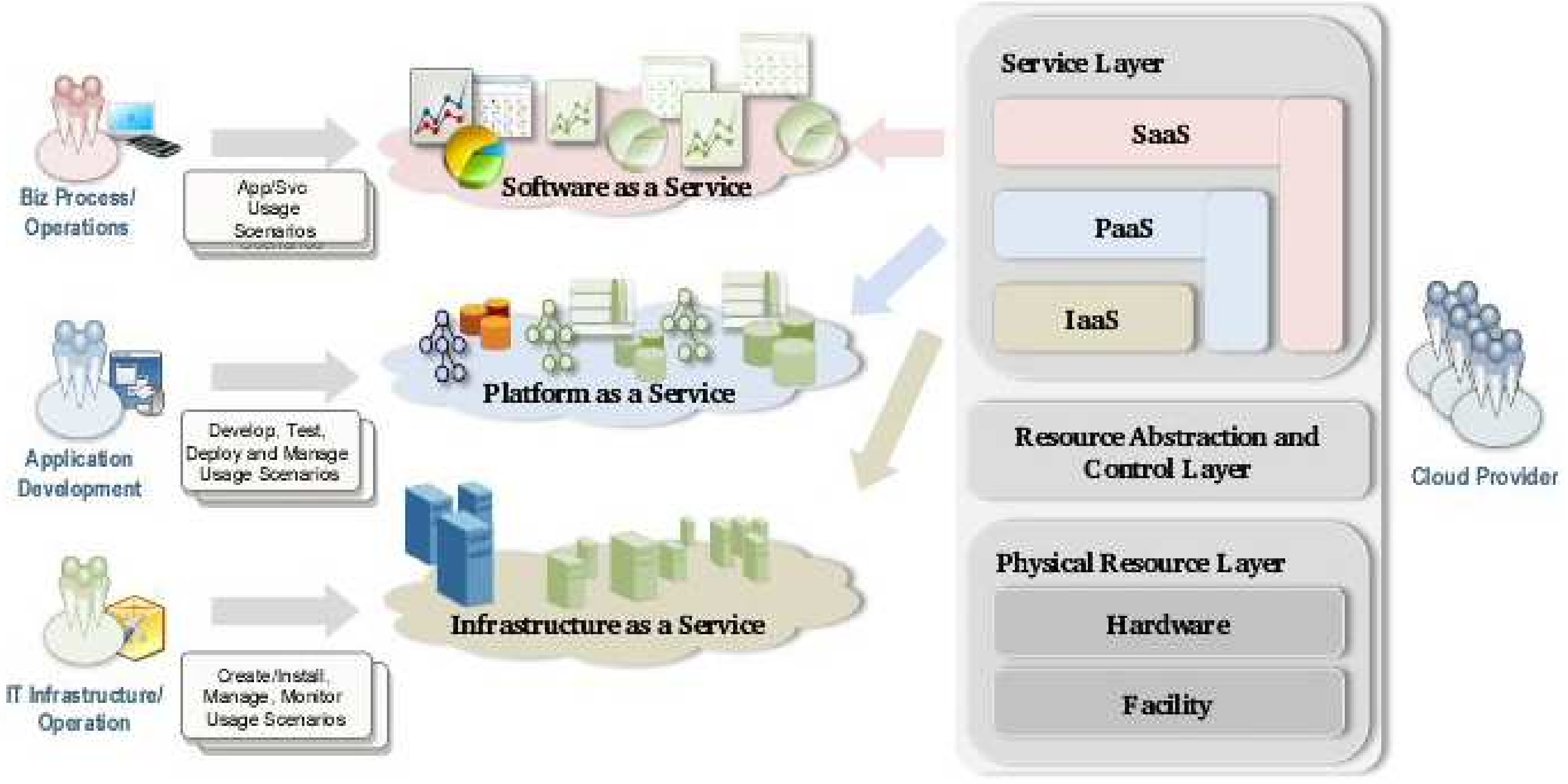
Cloud Provider – Service Handling

Providers must grant QoS of services, by assuring portability, interoperability and security, apart from performance

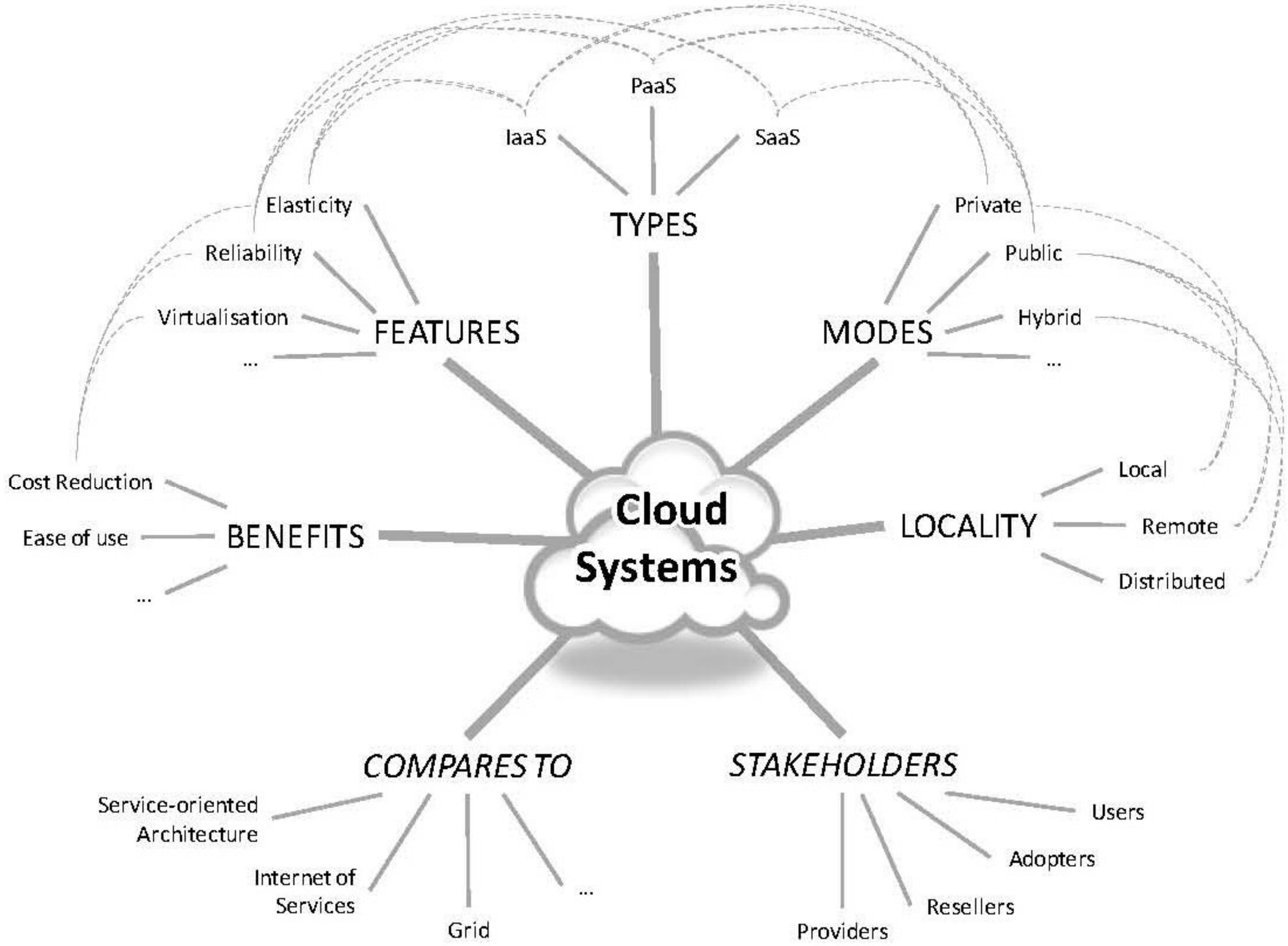


Cloud Providers - Orchestration

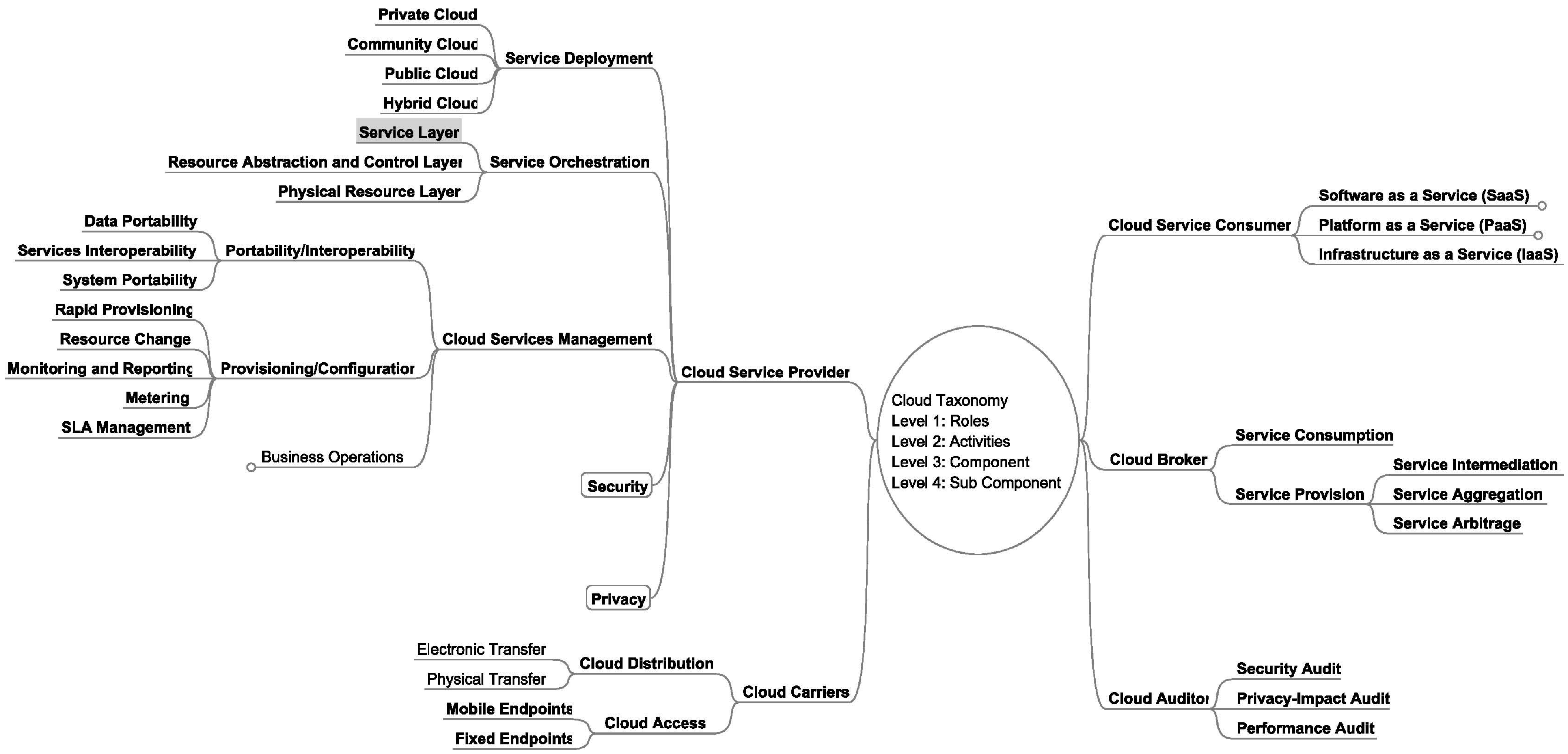
Providers should (could) coordinate offered services, implementing aggregation, intermediation, control and monitoring



Some significant aspects



A CLOUD TAXONOMY



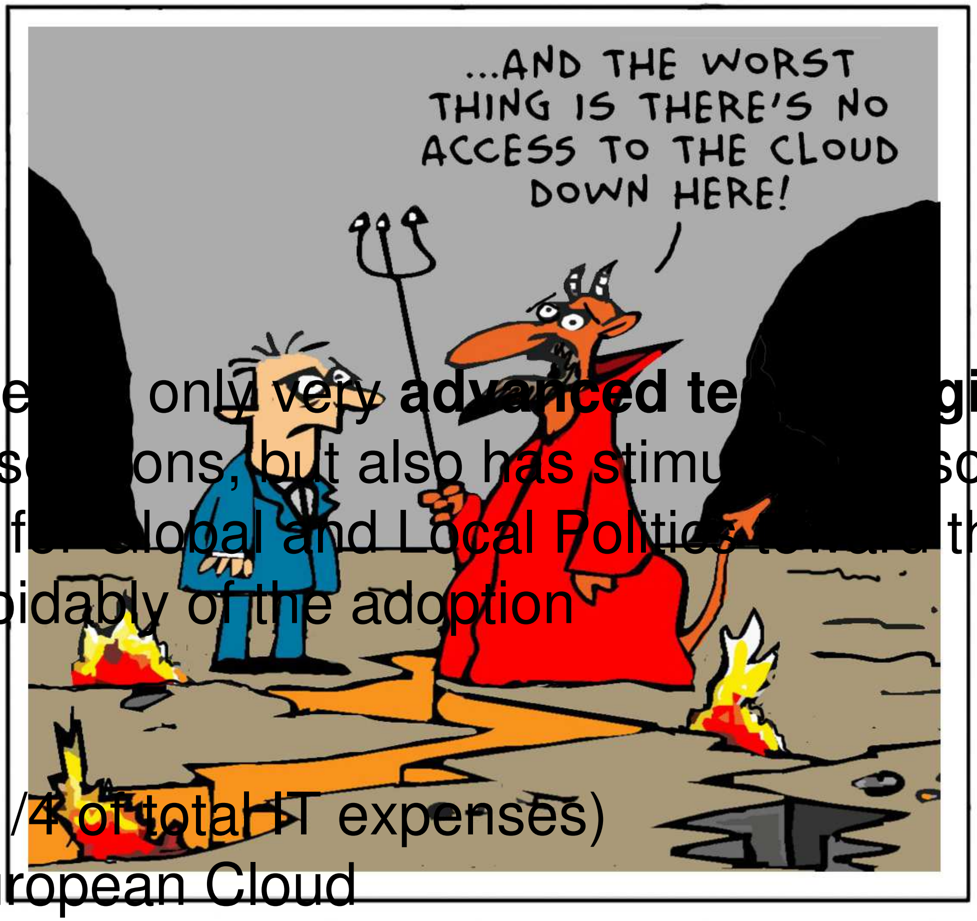
Cloud computing today

The Cloud term and its related technologies have become very common also for non technical users

- Advertising

- Humor

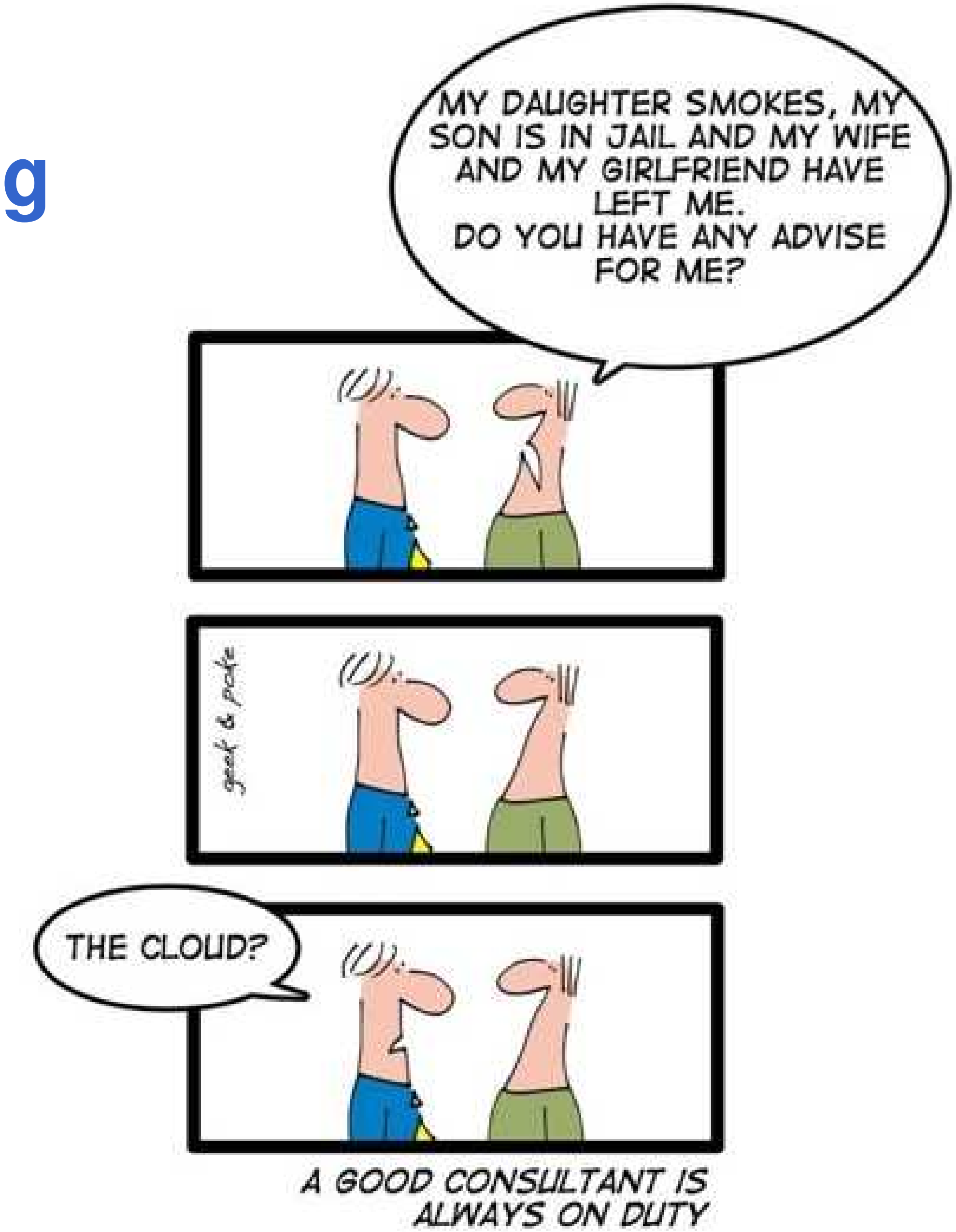
- Buzzword


...AND THE WORST THING IS THERE'S NO ACCESS TO THE CLOUD DOWN HERE!

Cloud has provided have [illegible] only very **advanced technologies also very widespread** solutions, but also has stimulated some directions as guidelines for Global and Local Politics toward the necessity and the unavoidably of the adoption

- G Cloud in UK
- USA: Federal Cloud (1/4 of total IT expenses)
- EU pushing toward European Cloud

Cloud for everything

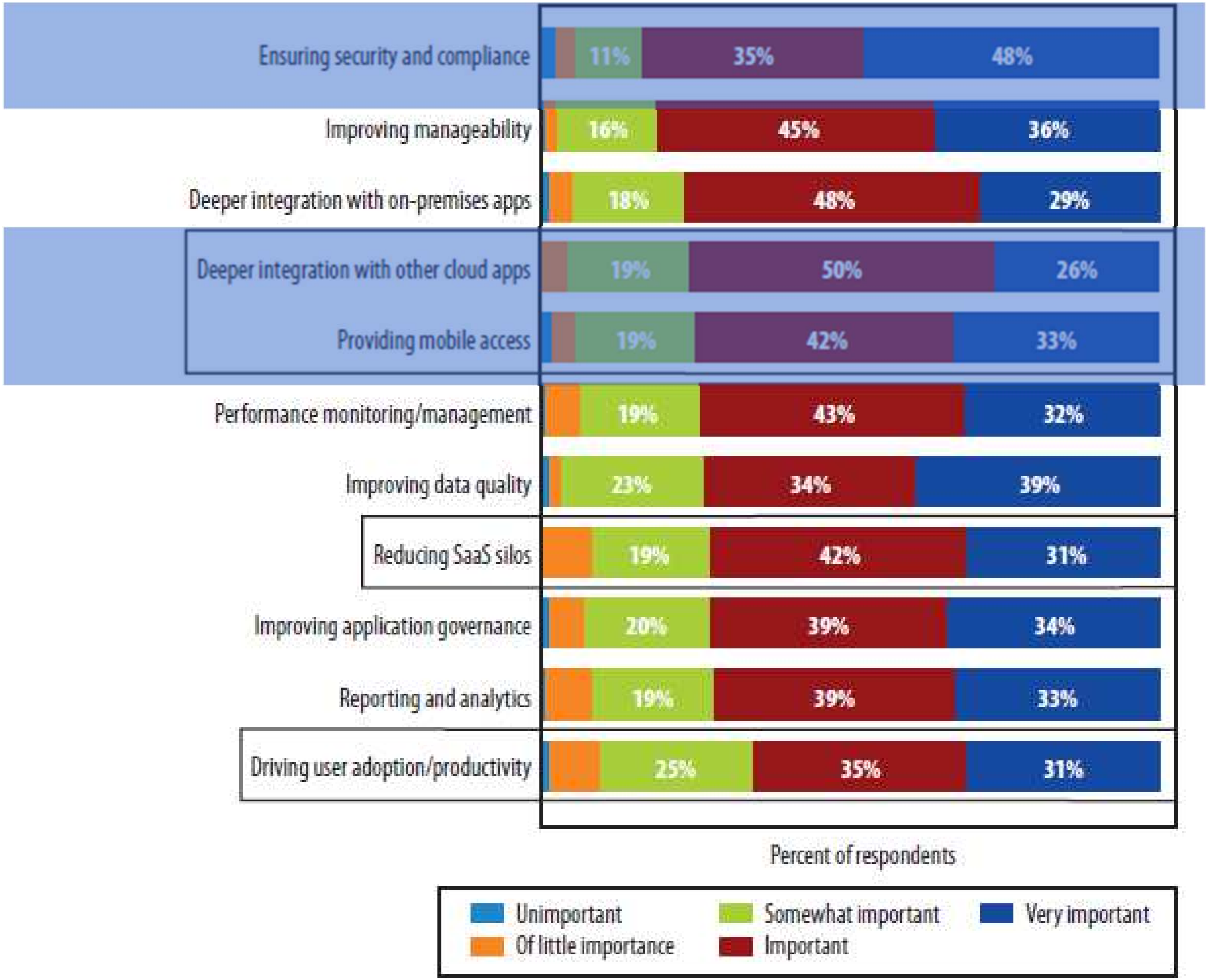

MY DAUGHTER SMOKES, MY SON IS IN JAIL AND MY WIFE AND MY GIRLFRIEND HAVE LEFT ME. DO YOU HAVE ANY ADVISE FOR ME?
geek & poke
THE CLOUD?
A GOOD CONSULTANT IS ALWAYS ON DUTY

Not exactly for everything ☺

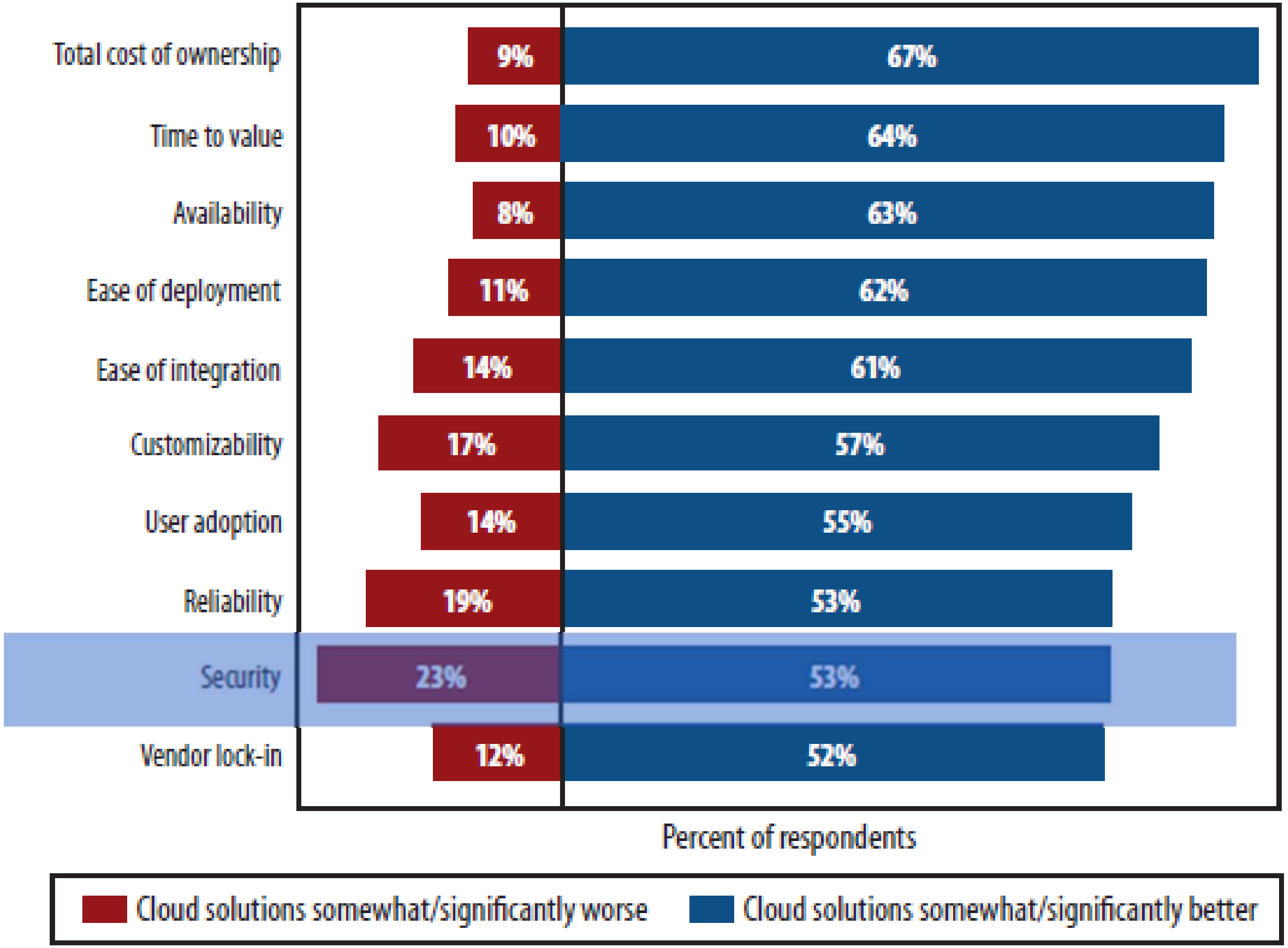
Cloud : perception and challenges

Security

Integration



Cloud computing 2.0



CLOUD ARCHITECTURE

In case of Cloud, resources must be considered in a less traditional way

Not only you have the application mapping but you should consider **very different execution environments** and **very different choices**

You can define and command:

- **logical resources** (already considered)
- **physical resources** (already considered)
- **virtual resources (**not only machines, but also **any kind)**

The degree of freedom you have are **many** and also **from different architectures** and **choices** can stem **very different final behavior**

So, you typically **decide**

how to put **your logical components over virtual resources**

and then also to **map the virtual over the physical one**

CLOUD CASE

We design an application thinking to **a client** that obtain **on-demand services** requested and obtained via **Web** and **the user must not worry (too) much about their management**

Their management is **Cloud-internally decided** and **provided**

Virtual and physical resources for Cloud are in **one data center** or in **different data centers (transparently)**

The user should definitely **use Resources-aaS** (**R**esources **as a S**ervice) and should expect a very **dynamic behavior** from the requested services

⇨ On need, the data center must prepare **new resources**, both physical and virtual ones, in a more or less automatic fashion

⇨ That makes the architecture perceived by user very **elastic, adaptable** and **flexible**

⇨ The problems are left to the **management of the data** center

CLOUD ARCHITECTURE

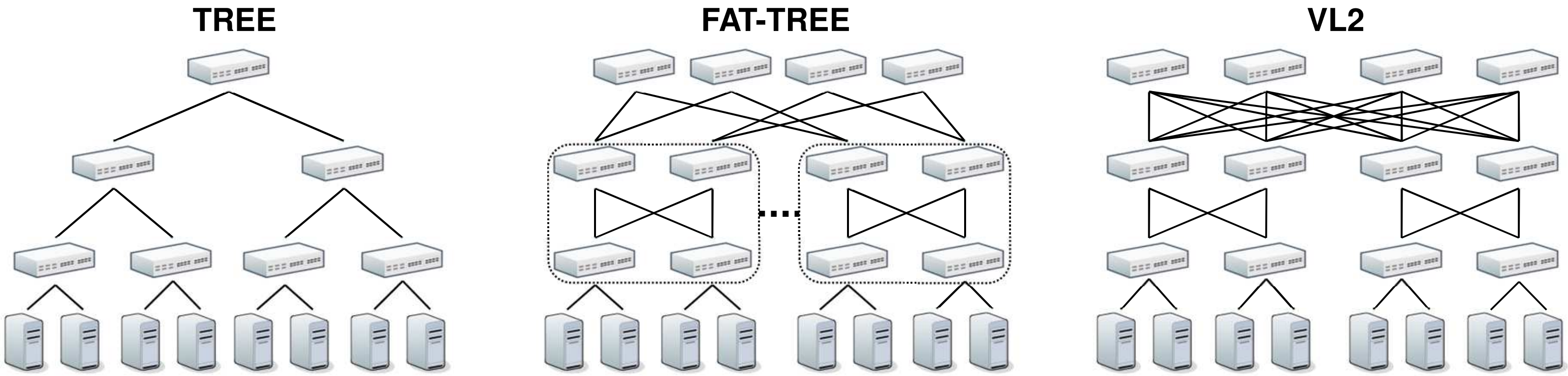
The Cloud makes an important step toward **transparency for users (PaaS, SaaS)**

But also makes available more **low-level details** (**IaaS**)

In particular the data center complexity is visible inside

- The data center has no **flat net** but typically **hierarchical ones that interconnect machines** and that can be **optimized by exploiting specific dynamic connections**

To reduce application time, the management can **allocate depending on internal data center interconnection**



CLOUD DEPLOYMENT (USER)

Choosing a CLOUD deployment instead of another **can have a big impact** during the **execution** and must be carefully evaluated and decided

Let you assume you need some resources and you do not have considered any policy,

- Typically you have several **setting** to decide among (some free, some are most expensive, …)
- You have to decide a suitable offering by considering the **average behavior and also its quality**: is it constant?, are there peaks?, are they regular?
- Your application has **specific requirements:** geographic allocation, reliability (multiple copies), QoS in terms of response time, specific persistency constraints, …
- Any specific **internal allocation constraints:** some parts must be close and heavily communicating
- **Last but most important: is your application compatible with the chosen Cloud?**

INTEREST for DEPLOYMENT

Choosing a deployment instead of another

can have a big impact during the **specific execution** and must be carefully evaluated and decided

Let you assume you need communication resources,

- we must consider **internode communication tools available** whether resources will be allocated to different nodes
- we must choose the **most appropriate communication tools for allocation that we are determining** (in case of different and heterogeneous architectures support)
- we also need to **optimize communication tools** when resources are present on the same machine, **inter-and intra-node communications differentiating node** (as they often do the existing middleware)
- we need to verify that the **deployment is suitable with expressed communication tools** and does **not cause problems** (by identifying and eliminating bottlenecks and critical cases)

CLOUD PROVISIONING & QoS

In a Cloud environment, we have a ***similar setting***

On the **external site**, **several users are possible** and they may interact *among themselves* but also

- must **discover services** and interact with them
- can **pack some resources** inside the Cloud

On the **internal site**, there are several other aspects to be considered

- Many services may **be made available,** at different levels
- Services can be **temporary or persistent**
- User must be able to **control resource consumption**
- User can command **not only available services**, but **ship new ones** and control them and manage their lifecycle
- Any resource must be **available for access, inspection, maintenance**, and **changes** (even in case of sharing)
- Other constraints may be part of the **SLA** and internal management

REMOTE MANAGEMENT FOR QoS

In **remote environments**, such as in **outsourcing** and in **Cloud ones**, it is compulsory something to ascertain the current state of the remote installation, not only for accounting purposes

we have to offer a very rich **management interface**, to allow to:

- **Access to any user related resource** (processing, memory, persistent data, network, … any *-aaS)
- **Control of the consumption** of any user related resource (current state, history for some periods, peaks, trends, … user-defined indicators)
- Discovery of **new services and new available resources** (new service can offer off-the-shelf ready-to-use solutions)
- Installation of special **user settings and environment** (new service to be developed from composing available ones or in a more specifically client-tailored way)
- Enlargement to **federated environments** for resource integration

Cloud computing …

- **Goal and requirements**
  - **Cost reduction** (to minimize deployment cost, energy, storage, computing power, …)
  - **Scalability on demand** (resources handled in an “elastic way”, all **system resources** are **virtualized** as for **virtual machine, agreed** and **granted** in **SLA** (Service Level Agreement
  - **Automated provisioning and ease of use** (utility computing + infrastructure, platform, and SaaS)
- **Technical areas of intervention**
  - **Management** (system resources, power-saving, …)
  - **Interoperability & portability** (data, applications, and virtual machine images)
  - **Measurement and monitoring** (dynamic on line monitoring, accounting control, …)
  - **Security** (privacy/data control, reputation, …)

Some open technological problems

Many aspects have been solved, not all of them are, some still hard to tackle

- Virtualization
  - New forms of resource virtualization
- Differentiated and global resource localization
  - Federation and coordination of global resources
- Security, Privacy and SLA adherence
  - Verifiable and trusted assurance policies
- Easy Control, handling and management by user
  - Easy-to-use and user friendly actions and tools
- Data and QoS management
  - self-* and automated system capable of adaptation
- API and platform enhancements
  - New platforms e new interaction modes

Standard: a necessity

Cloud as a new sector, unavoidable in expansion and spreading, but
acceleration favored by standard acceptance

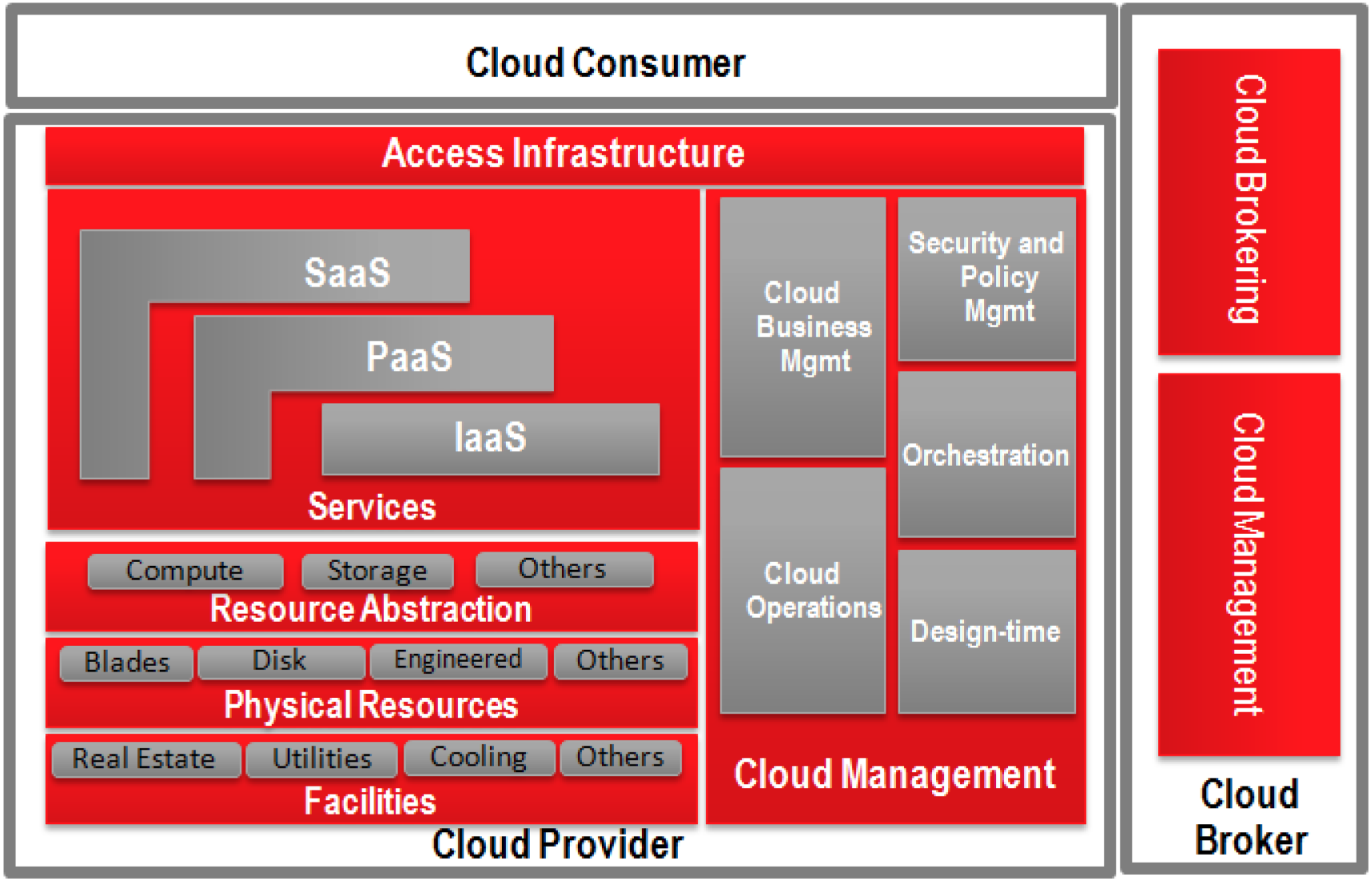
- Clarity about new roles and responsibility
- Open source standard and implementations
- Integration with existing protocols (mobile …)
- Supports for sustainability
- Global and local legal clarity

- Ties with other areas:

  Big Data, Open Data, and **Smart City**

Cloud Solution roles

A possible **Cloud set of scopes**

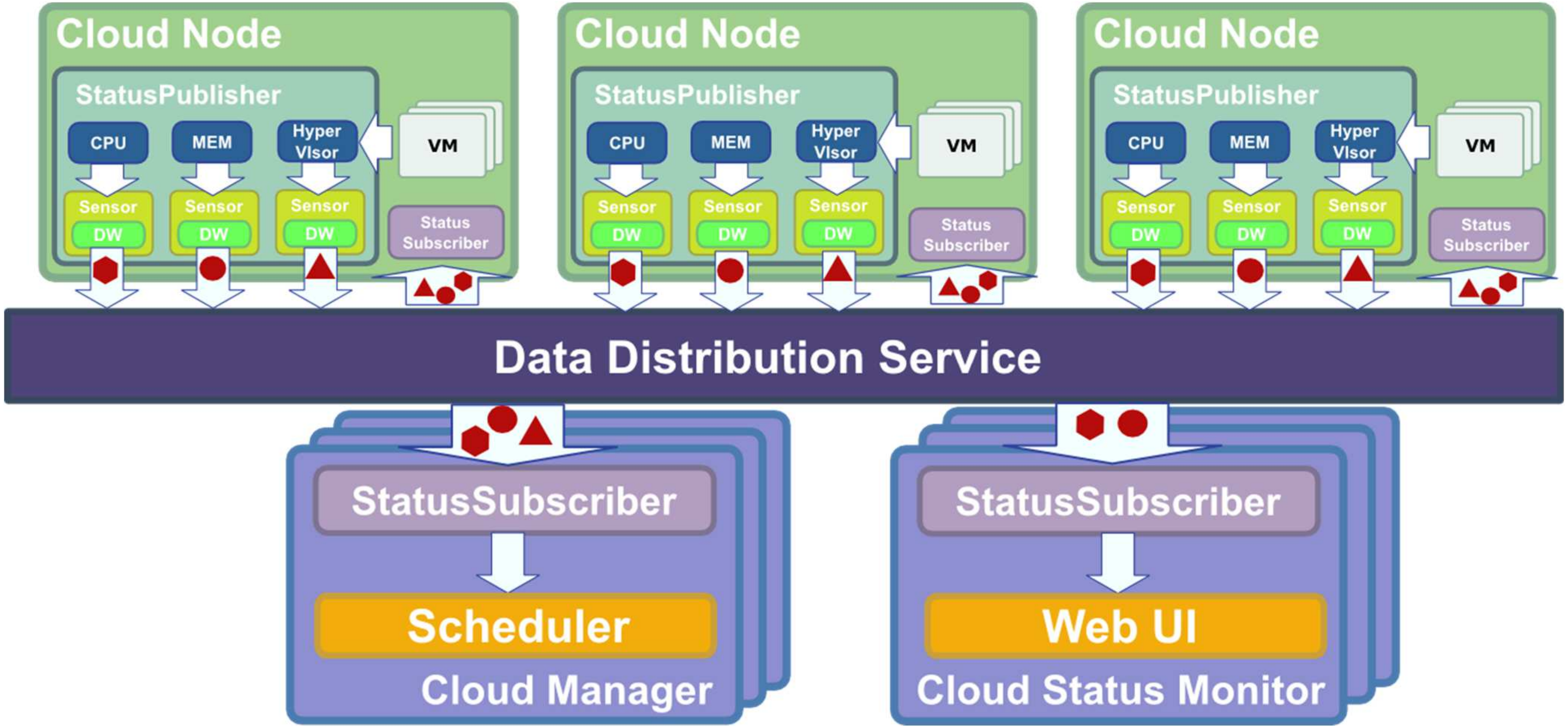


Cloud Monitoring

Monitor and **manager components** (several possible deployments)

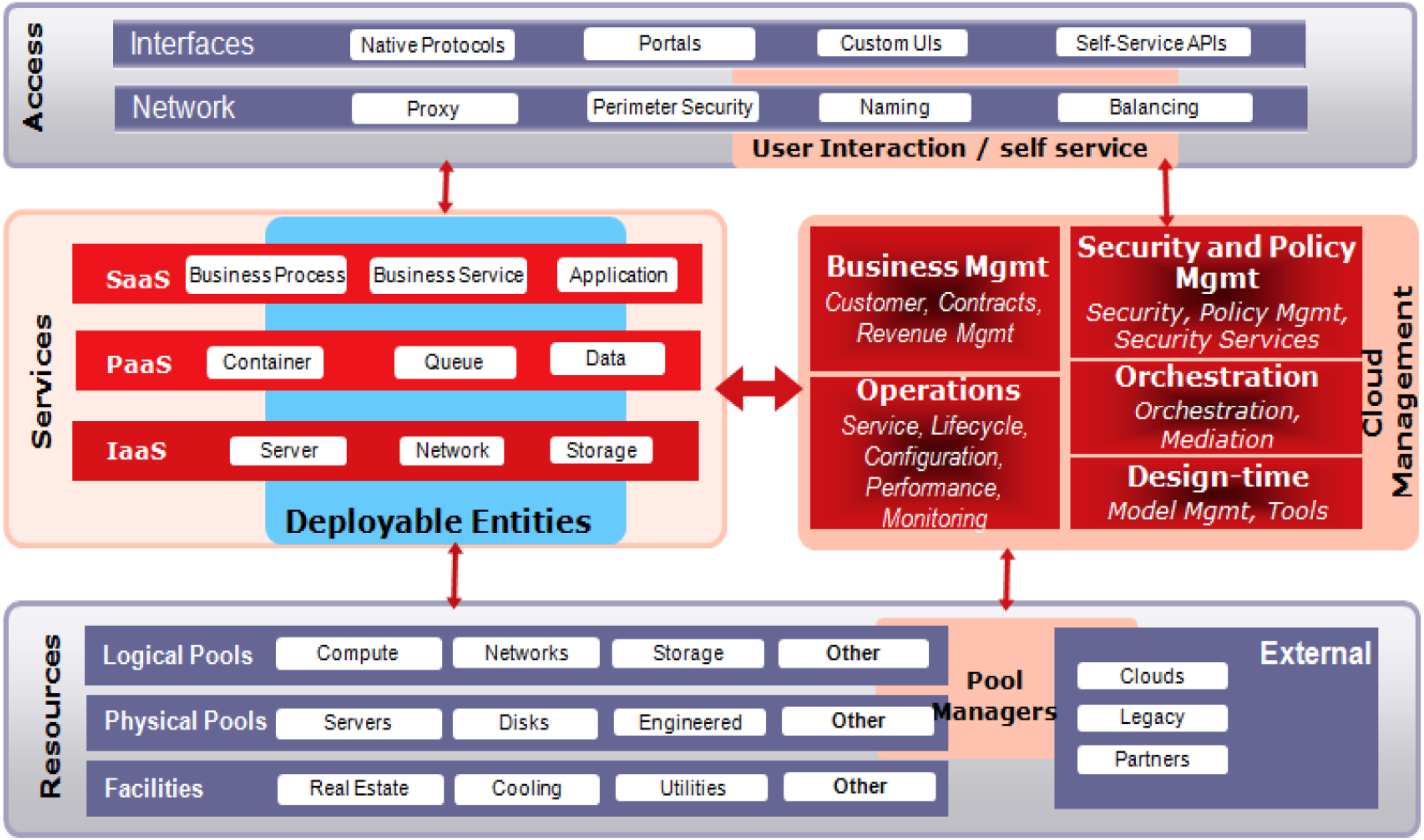
Physical and **virtual resource** monitoring

Many-to-many communication for **fine-grained local monitoring**



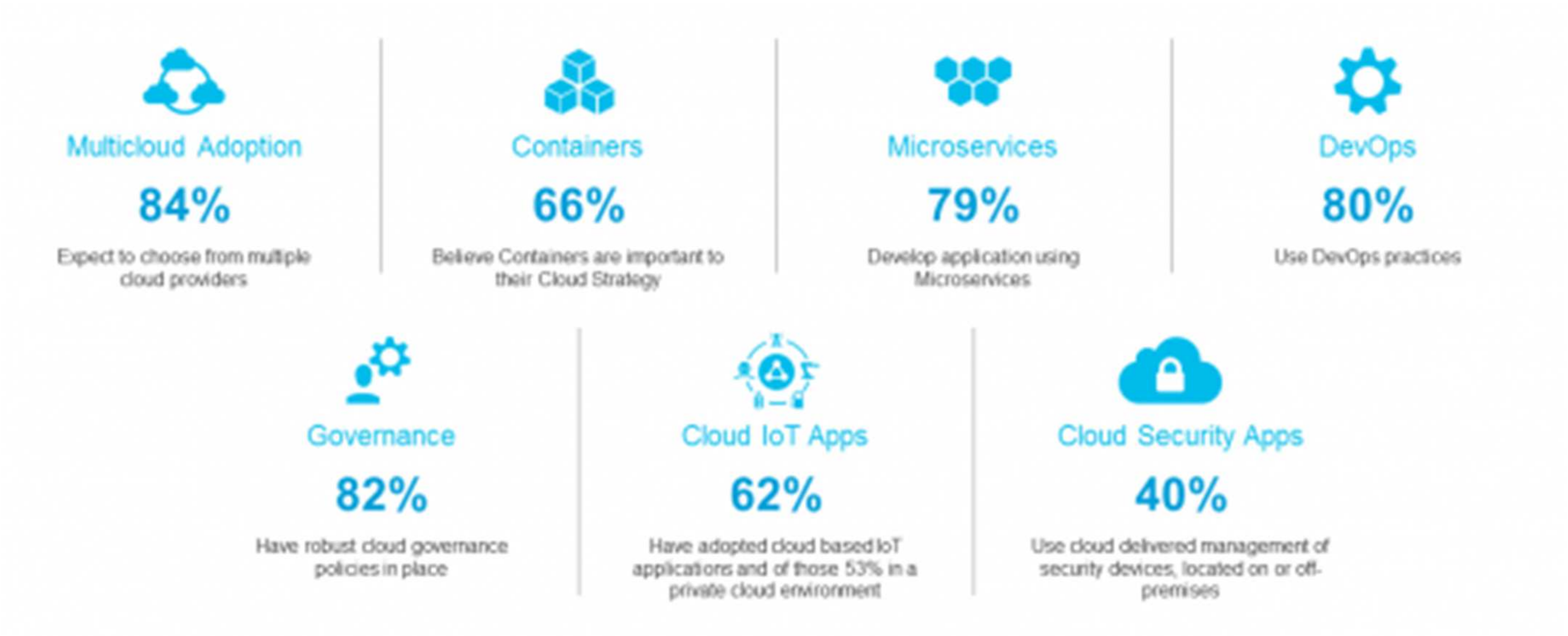
Cloud Components

A **Cloud-layered infrastructure in Cloud components**



Cloud connections

Cloud is connected with other **state of the art technologies**



Multicloud Adoption

84%

Expect to choose from multiple cloud providers

Containers

66%

Believe Containers are important to their Cloud Strategy

Microservices

79%

Develop application using Microservices

DevOps

80%

Use DevOps practices

Governance

82%

Have robust cloud governance policies in place

Cloud IoT Apps

62%

Have adopted cloud based IoT applications and of those 53% in a private cloud environment

Cloud Security Apps

40%

Use cloud delivered management of security devices, located on or off-premises

Readings (and more to come…)

M. Creeger, "***Cloud Computing: An Overview***", ACM Queue, vol. 7, no. 5, pp. 3-4, 2009.

A. Lenk, M. Klems, J. Nimis, et al., "***What's inside the Cloud? An architectural map of the Cloud landscape***", ICSE Workshop on Software Engineering Challenges of Cloud Computing, 2009.

W. Vogels, "***Eventually consistent***", Communications of the ACM, vol. 52, no. 1, pp. 40-44, 2009.

B. Narasimhan, R. Nichols, "***State of Cloud Applications and Platforms: The Cloud Adopters' View*** ", IEEE Computer, vol. 44, no. 3, pp. 24-28 (March 2011)