Cloud Platforms: Concepts, Definitions, Architectures and Open Issues

Samir Tata, Institut Mines-Télécom
Télécom SudParis

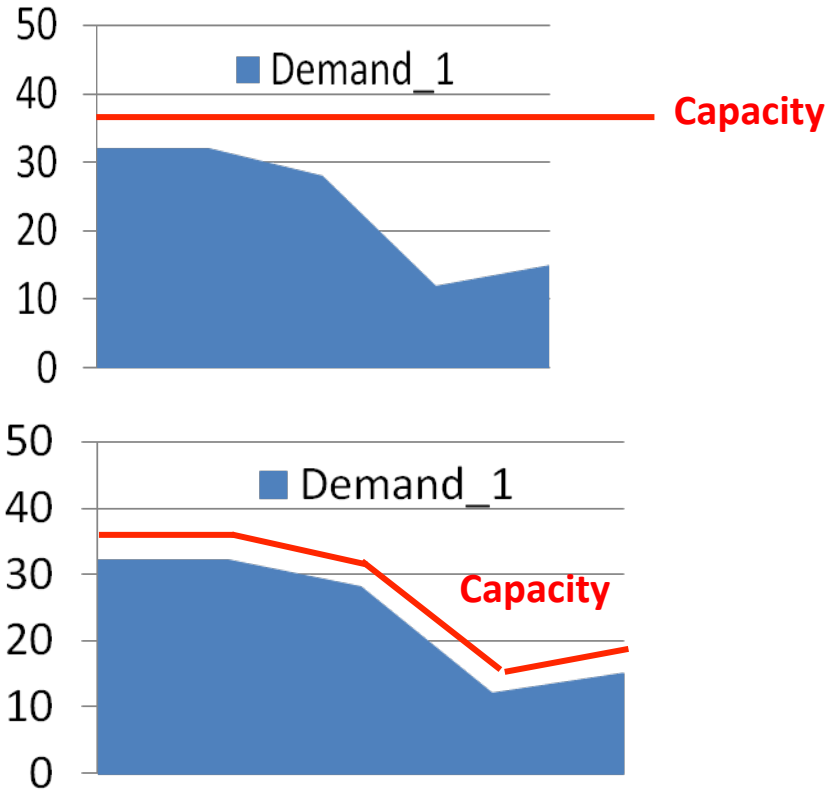
Outline

- Concepts & Definitions
- Architectures
- Standards
- Open issues
- Conclusion

Cloud Platforms: Concepts & Definitions

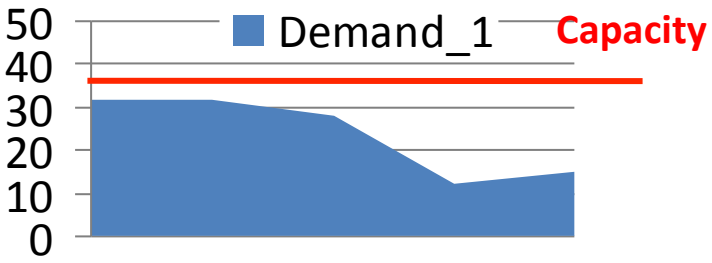
Cloud business model: Client benefits

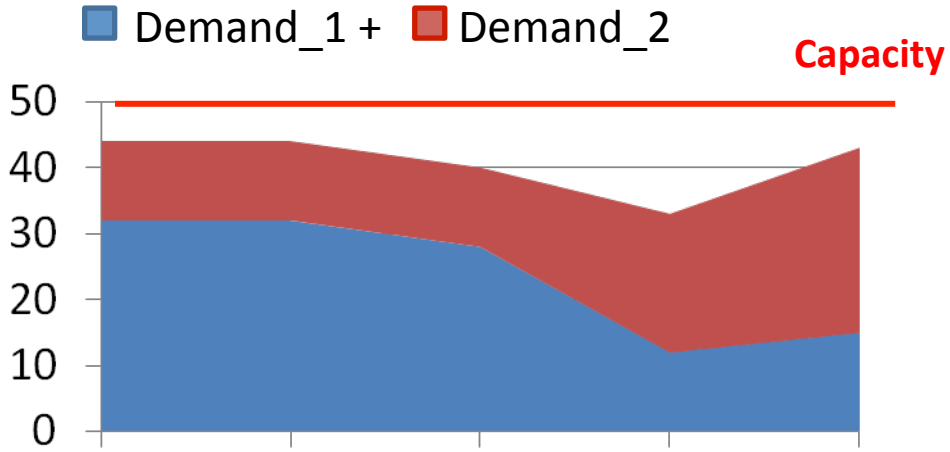
- **Pay by use instead of provisioning for peak**

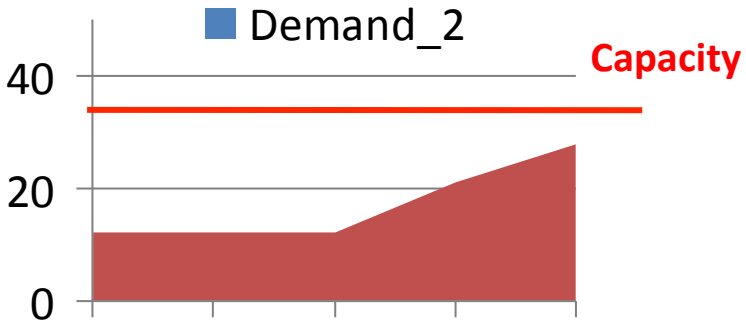


Cloud business model: Provider benefits

- **Share capabilities (resources, services, etc.)**



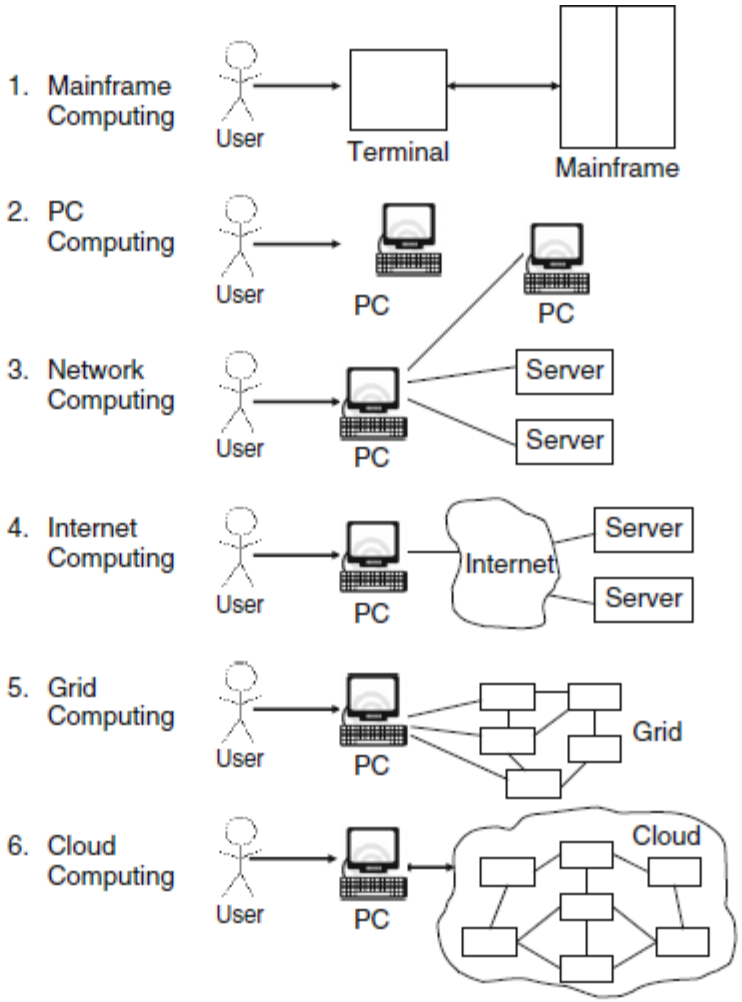




What is Cloud Computing

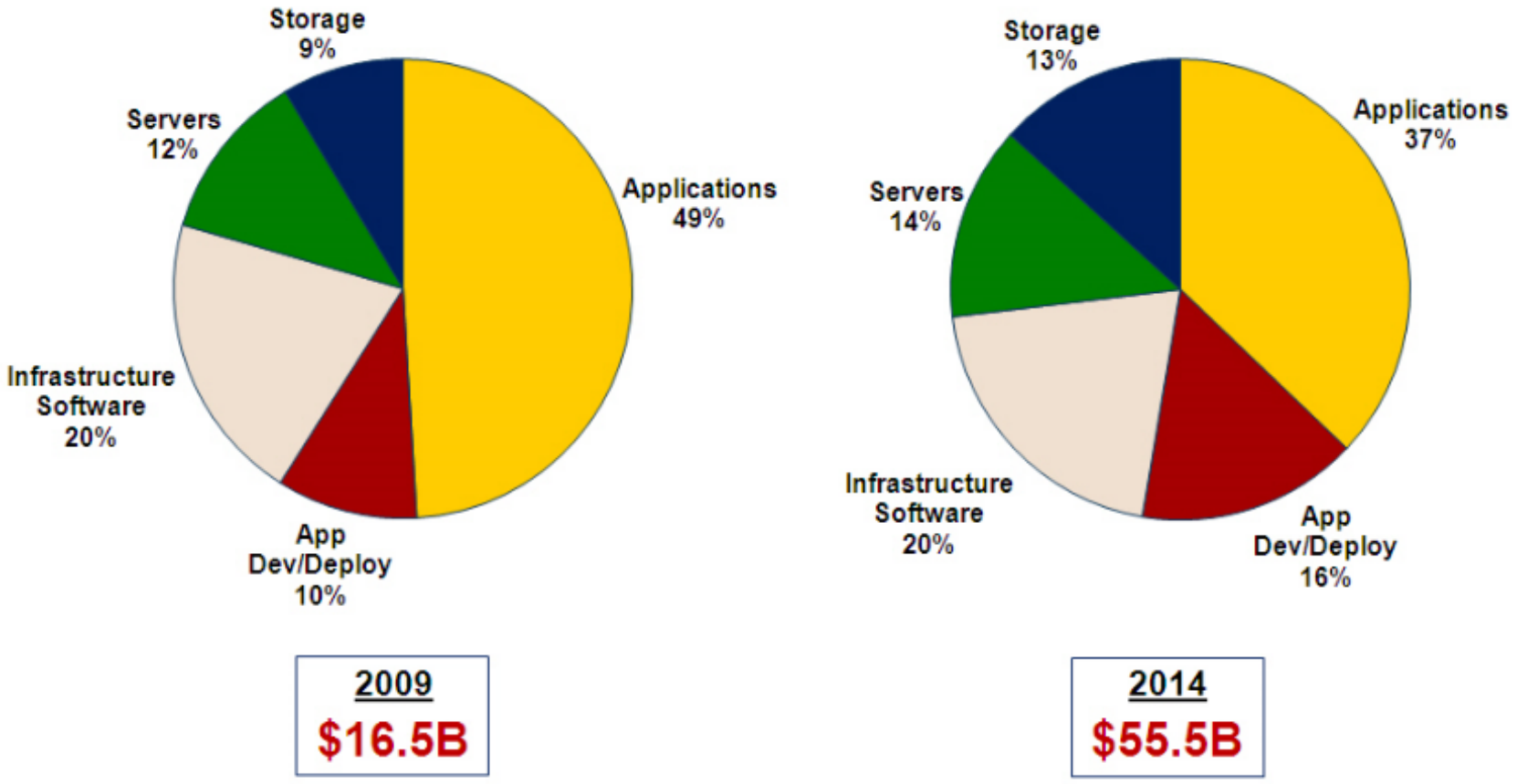
- **Not yet a common definition**
- **Some definitions**
  - A style of computing where massively scalable IT-enabled capabilities are provided "as a service" over the network (Petroleum Federation of India)
  - A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet (Foster et al 2008)
  - Clouds, or clusters of distributed computers, provide on-demand resources and services over a network, usually the Internet, with the scale and reliability of a data center (Grossman 2009).
  - 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 (NIST).

Cloud computing vs X computing



Voas, J., & Zhang, J. (March/April 2009). Cloud computing: New wine or just a new bottle? *IEEEITPro*, 15–17.

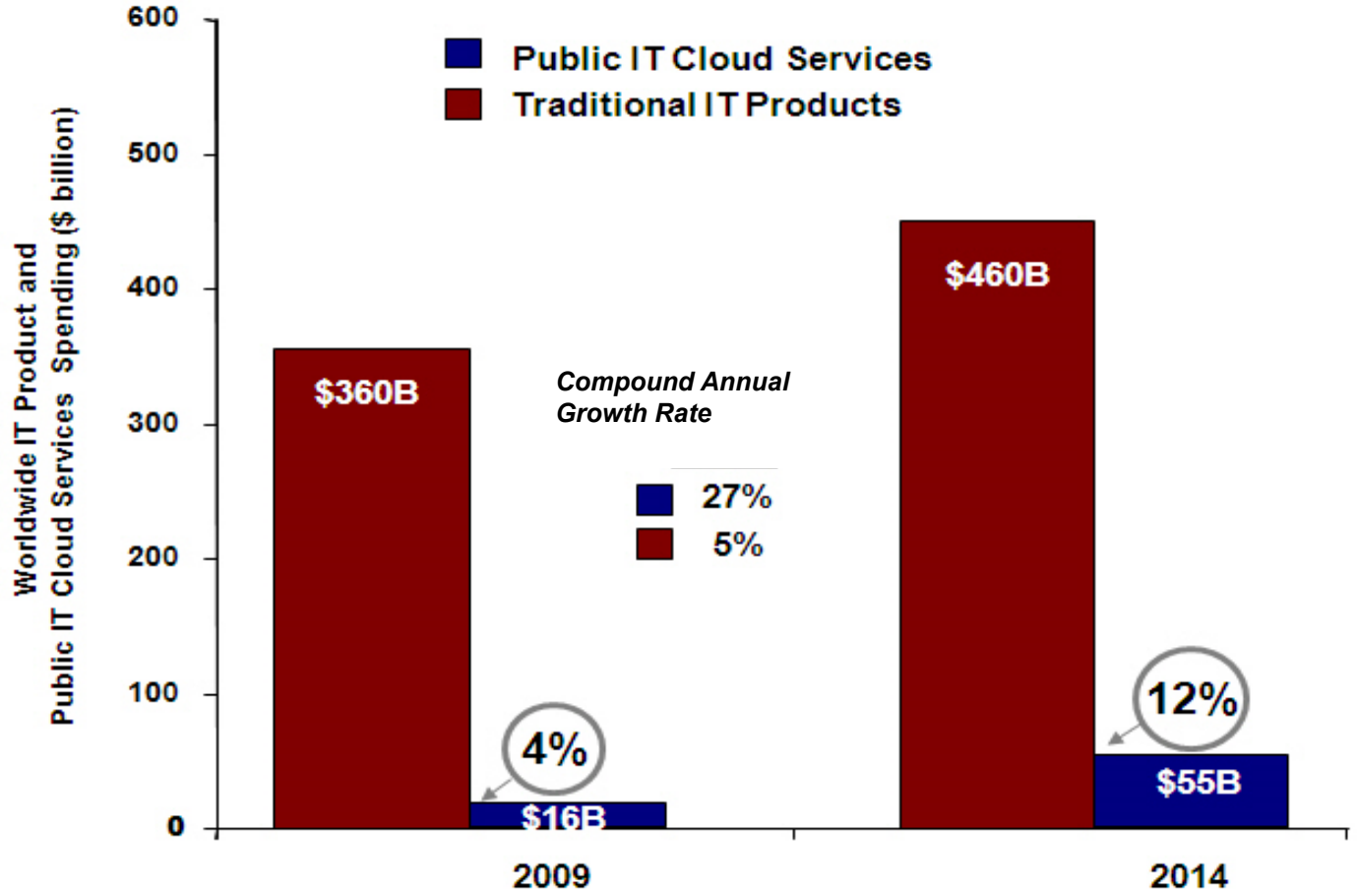
Worldwide Cloud Spending by category



Source: IDC, June 2010

* Includes spending on Applications, Application Development & Deployment Software, Systems Infrastructure Software, Server capacity and Storage capacity provided via the public Cloud Services delivery model.

Worldwide Cloud Spending by consumption



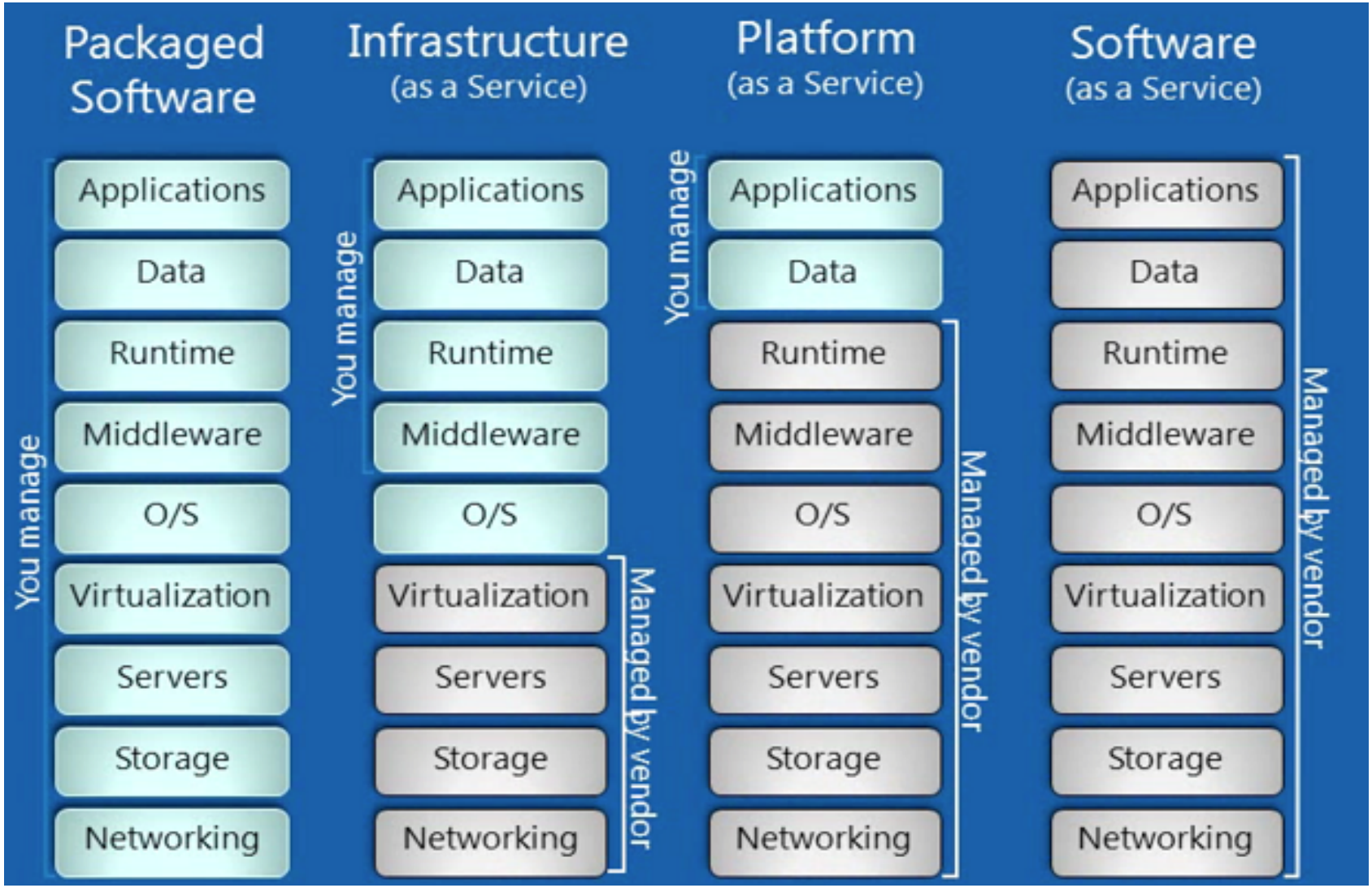
Source: IDC, June 2010

* Includes spending on Applications, Application Development & Deployment Software, Systems Infrastructure Software, Server capacity and Storage capacity via both traditional product model and the public Cloud Services model.

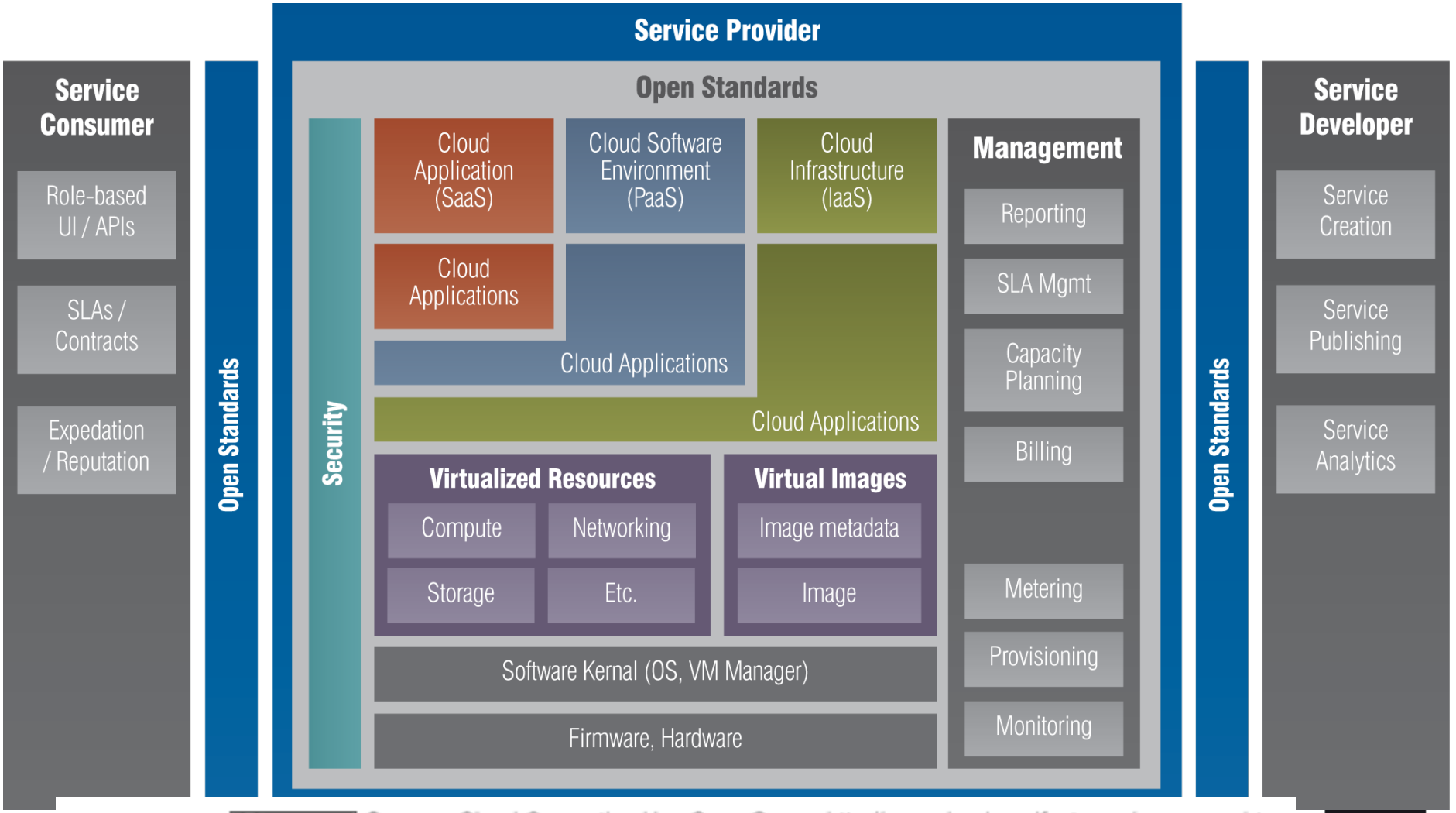
Cloud models and services

- **Models**
  - Private : enterprise owned
  - Public: sold to public
  - Hybrid: composition of different cloud models
  - Virtual: intermediate cloud clients and cloud providers (different models)
  - Community: shared for a specific community
- **Services**
  - IaaS: ready to use storage, computing and/or network resources
  - PaaS: ready to use platforms to host client created applications
  - SaaS: ready to use software
  - XaaS: DaaS, NaaS, etc.

Cloud Provision



Cloud Computing Taxonomy



CC BY-SA **Source:** Cloud Computing Use Case Group. http://opencloudmanifesto.org/resources.htm

Traditional platform … ?

- **A platform is a software that**
  - is a software used to develop, deploy, host, run and/or manage software application
  - includes support programs, compilers, code libraries, an application programming interface (API) and tool sets
  - Developers focus on business code, the platforms provides the rest (ensure non functional properties)

- **Example**
  - Apache
  - Tomcat
  - Jboss
  - JVM

PaaS

- **A PaaS is a platform with**
  - more cost-effective model for application development and delivery
  - functionalities provided as (virtualized) services
- **Definition**
  - Delivery of a computing platform and solution stack as a service. It facilitates deployment of applications without the cost and complexity of buying and managing the underlying hardware and software layers.
- **Virtualized services**
  - Whatever its location, implementation, etc.
    - Example: bank services
  - Composable → Platform a la carte

PaaS Examples

- **Comercial**
  - Amazon Web Services (AWS)
  - Google AppEngine
  - Salesforce.com
  - Microsoft Azure
  - Etc

- **Open Source**
  - OpenShift
  - CloudFoundry
  - Etc

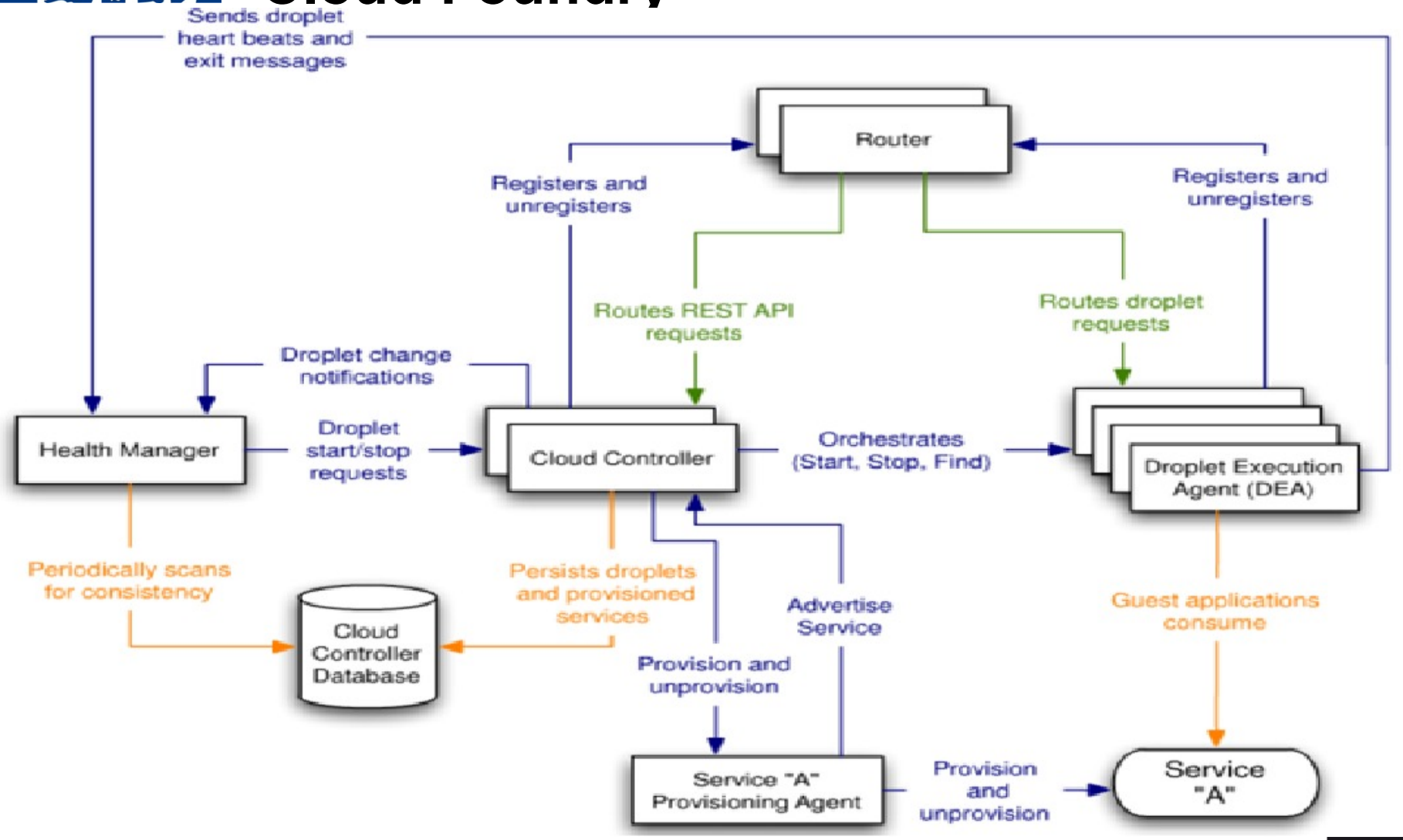
Cloud Platforms: Architectures

Architecture ?

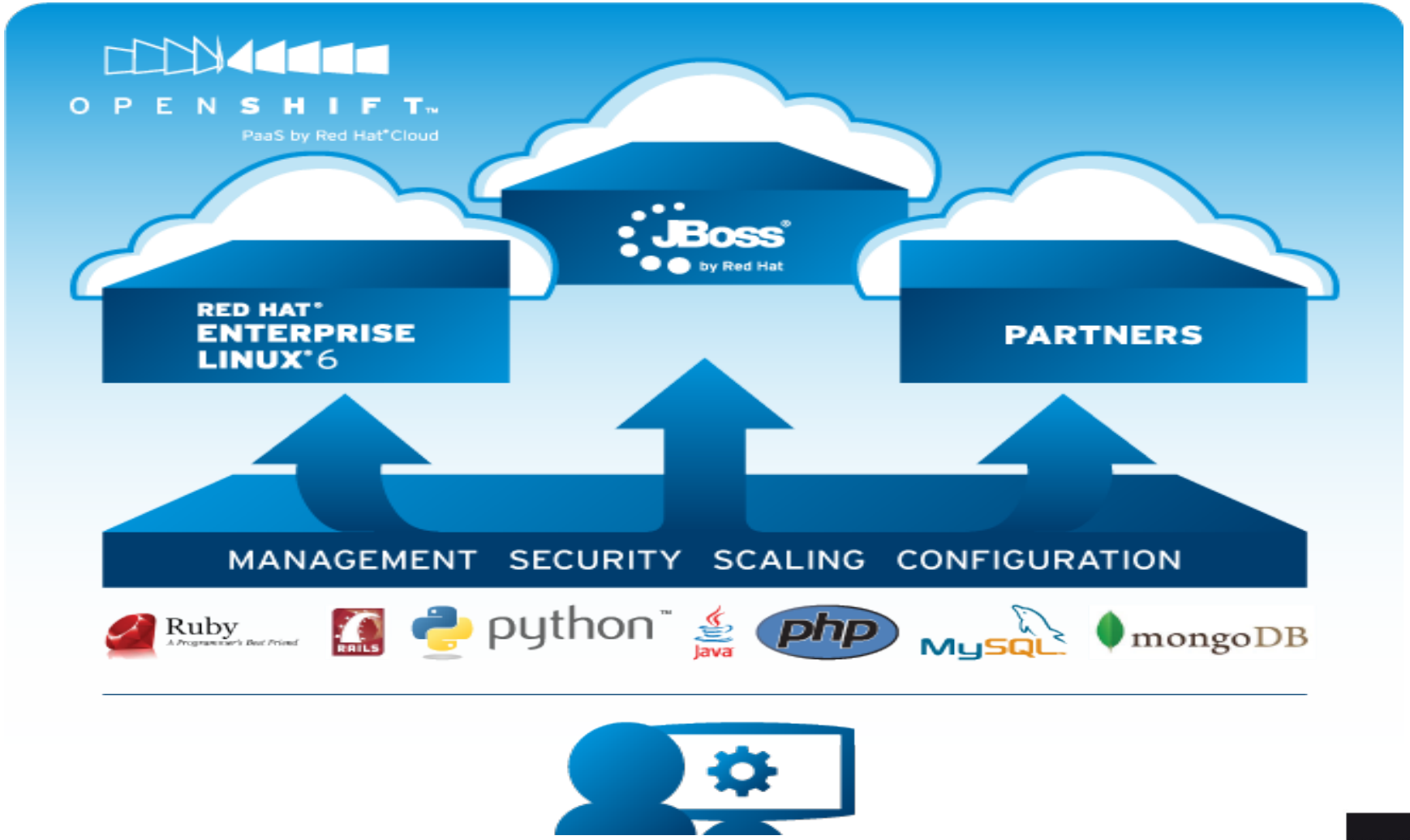
- **Traditional platform : set of components for**
  - Automatic deployment (somehow)
  - Monitoring
  - Manual sizing → do not meet Cloud requirements

- **PaaS architecture**
  - Automatic management: architecture where components should be automatically managed
  - Elasticity: resource provisioning evolves with resource demand

Cloud Foundry

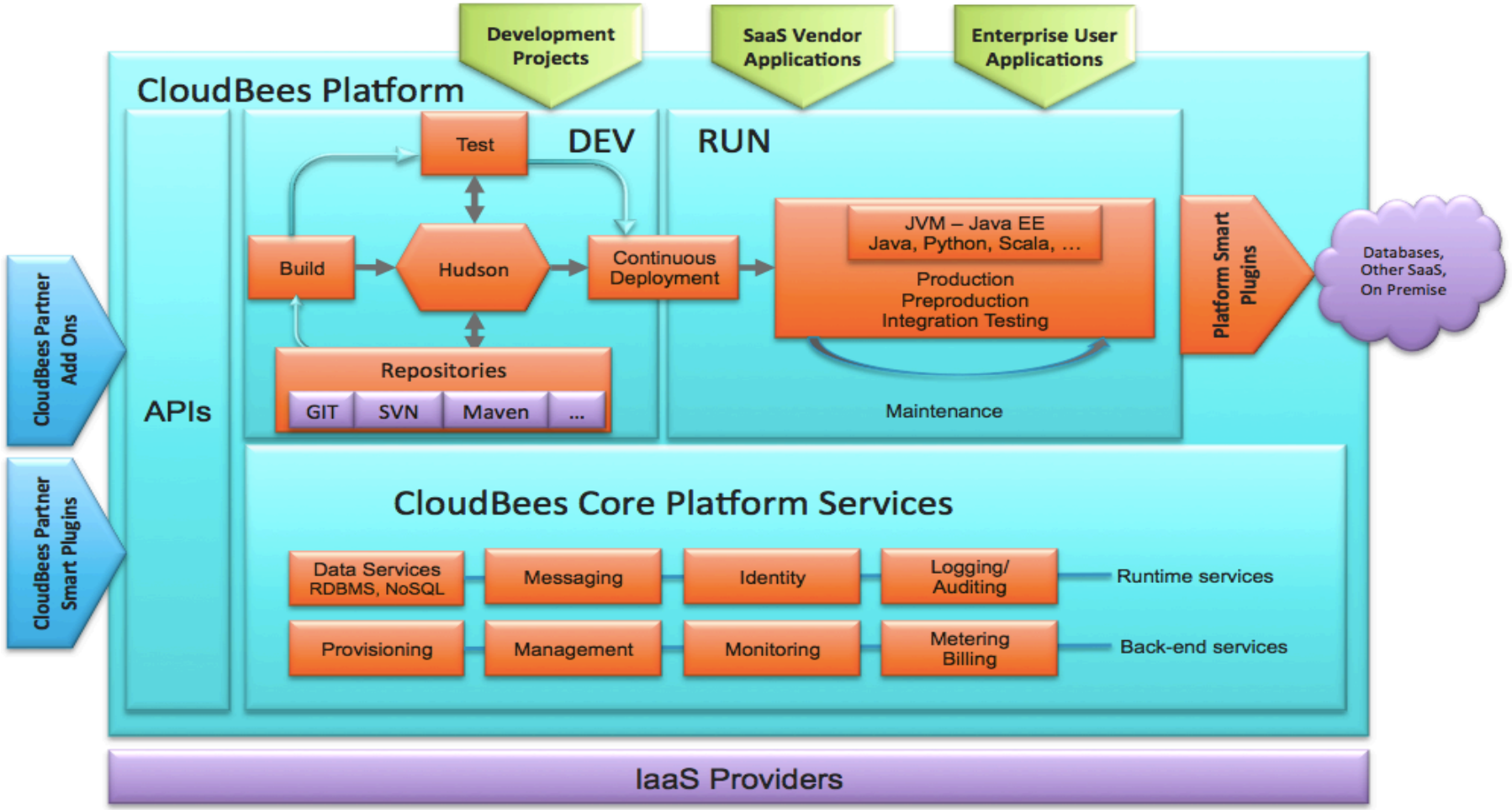


OpenShift

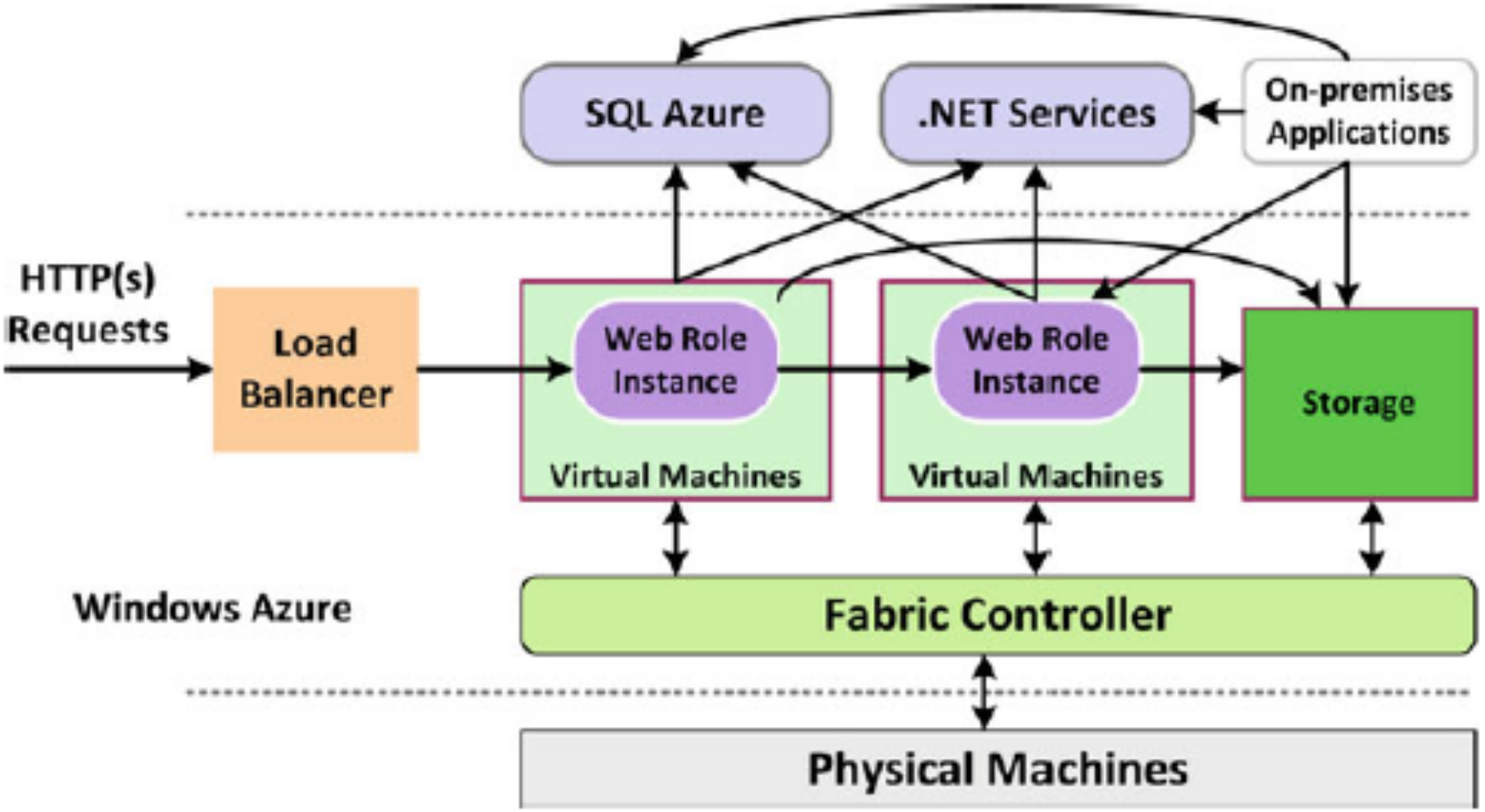

OPENSHIFT™
PaaS by Red Hat®Cloud
JBoss
by Red Hat
RED HAT®
ENTERPRISE
LINUX®6
PARTNERS
MANAGEMENT SECURITY SCALING CONFIGURATION
Ruby
RAILS
python™
Java
php
MySQL
mongoDB

CloudBees Architecture

CloudBees Platform as a Service



Microsoft Azure

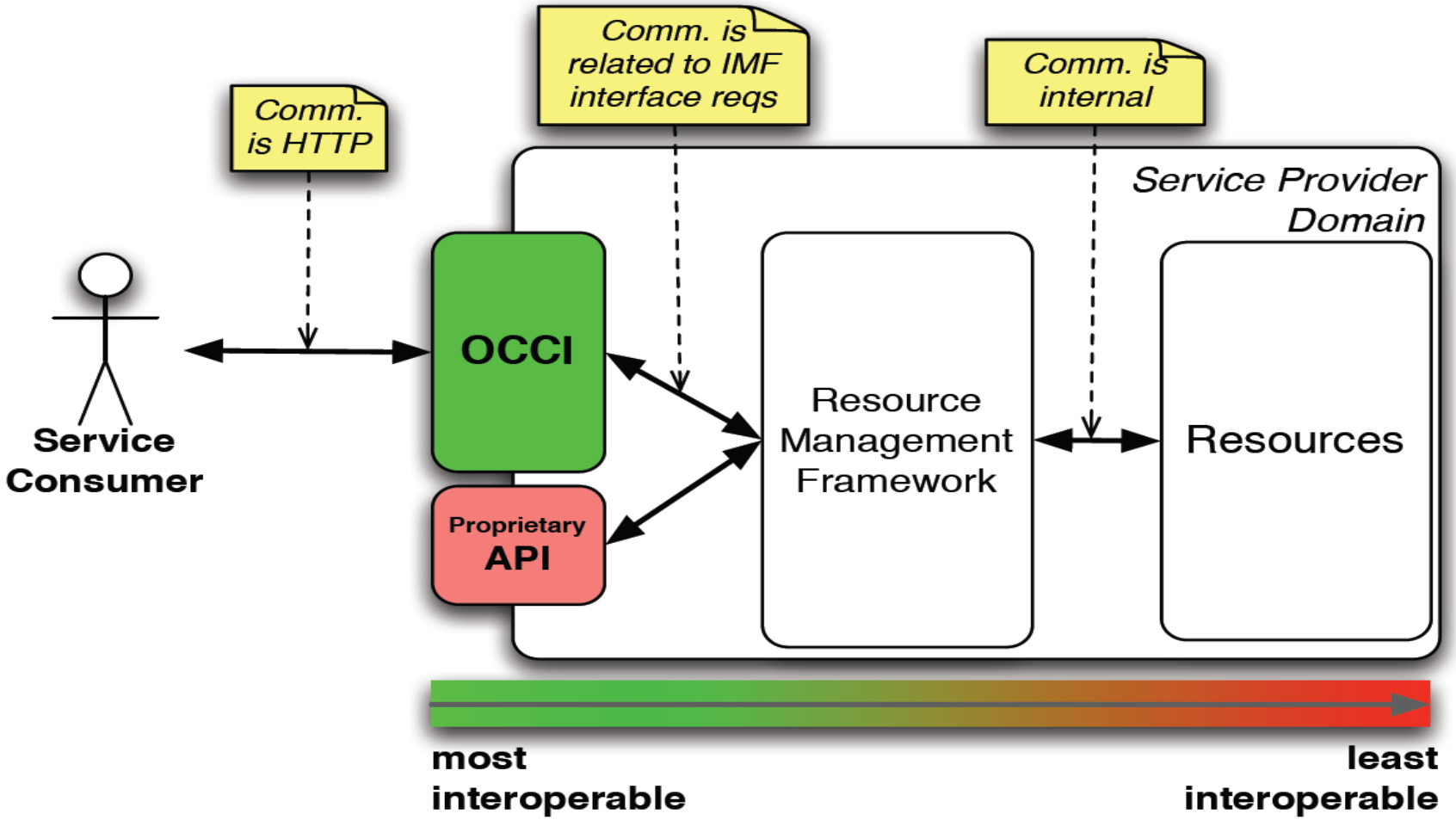


Cloud Platforms: Standards

Cloud Standards

- **Resource abstraction**
  - DMTF Open Virtualization Format (OVF)
- **DaaS API**
  - SNIA Cloud Data Management Interface (CDMI)
- **IaaS management API**
  - OGF Open Cloud Computing Interface (OCCI)
- **PaaS**
  - OASIS TOSCA
  - OASIS CAMP
- **SaaS**

OGF OCCI: Use case

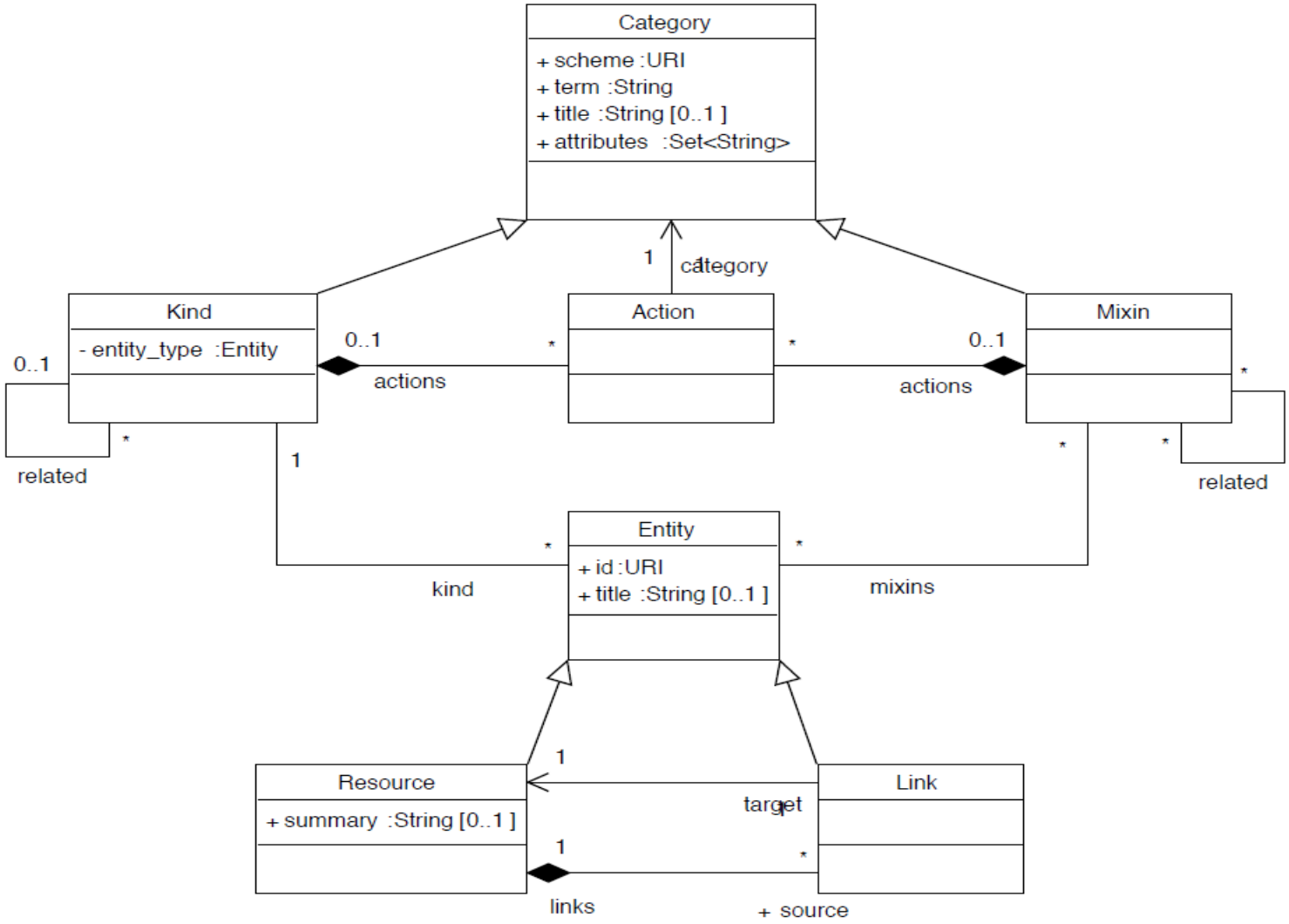


What is OCCI ?

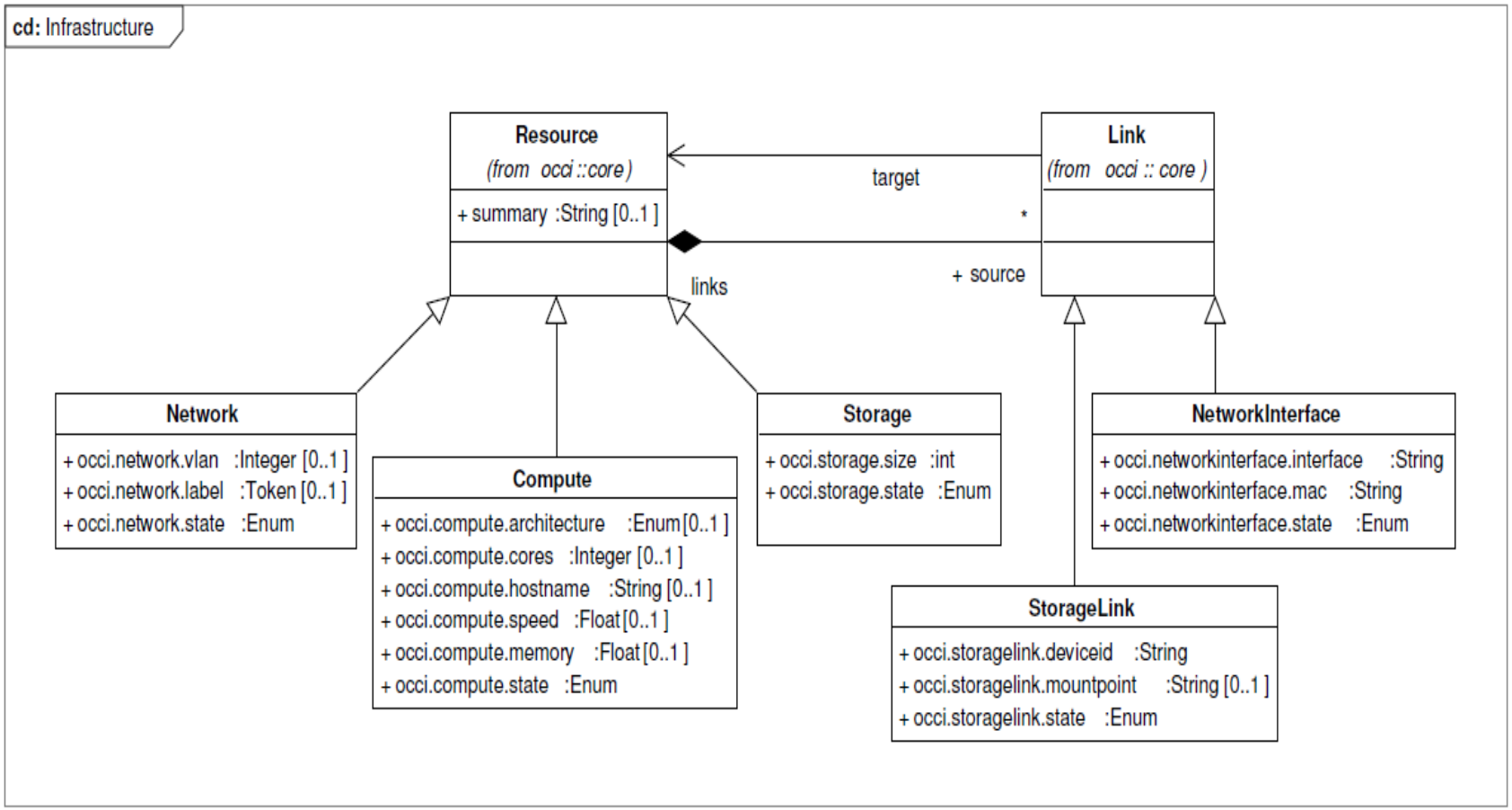
- **Open Cloud Computing Interface (OCCI)**
  - RESTful **Protocol** for management tasks
  - Flexible **API** with a strong focus on interoperability while still offering a high degree of extensibility.

- **OCCI specification:**
  - OCCI Core
  - OCCI Renderings: RESTful HTTP rendering
  - OCCI Extensions: Infrastructure

Occi core Model Overview



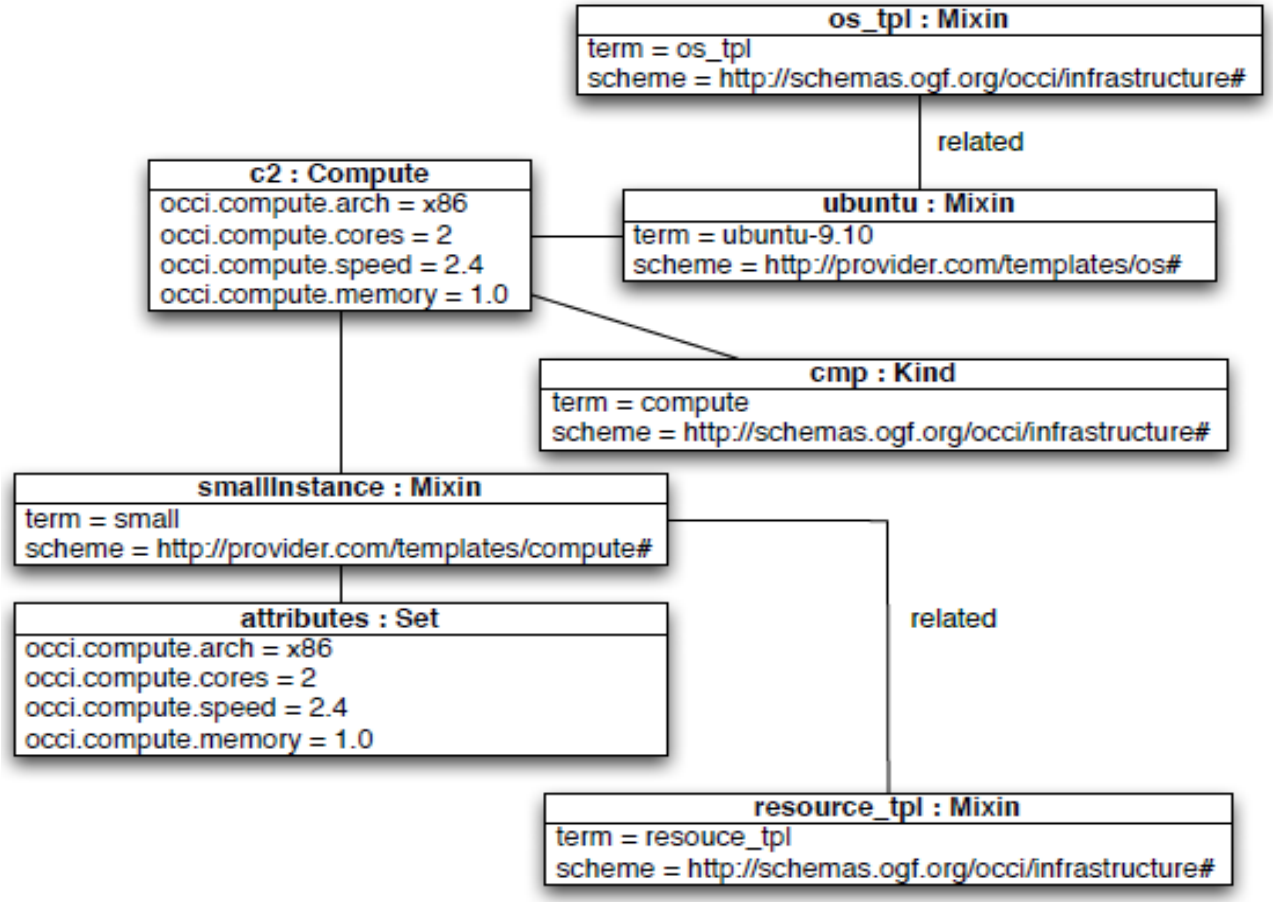
OCCI Infrastructure



Actions and states

- **Compute**
  - Actions: start, stop, restart, suspend
- **Storage**
  - Actions: online, offline, backup, snapshot, resize
- **Network**
  - Actions: up, down

Mixin


os_tpl : Mixin
term = os_tpl
scheme = http://schemas.ogf.org/occi/infrastructure#
related
c2 : Compute
occi.compute.arch = x86
occi.compute.cores = 2
occi.compute.speed = 2.4
occi.compute.memory = 1.0
ubuntu : Mixin
term = ubuntu-9.10
scheme = http://provider.com/templates/os#
cmp : Kind
term = compute
scheme = http://schemas.ogf.org/occi/infrastructure#
smallInstance : Mixin
term = small
scheme = http://provider.com/templates/compute#
attributes : Set
occi.compute.arch = x86
occi.compute.cores = 2
occi.compute.speed = 2.4
occi.compute.memory = 1.0
related
resource_tpl : Mixin
term = resouce_tpl
scheme = http://schemas.ogf.org/occi/infrastructure#

OCCI Rendering

- **Creating a resource instance**
  - > POST /compute/ HTTP/1.1
  - > [...]
  - >
  - > Category: compute; scheme="http://schemas.ogf.org/occi/infrastructure#"; class="kind";
  - > X-OCCI-Attribute: occi.compute.cores=2
  - > X-OCCI-Attribute: occi.compute.hostname="foobar"
  - > [...]
  - < HTTP/1.1 201 OK
  - < [...]
  - < Location: http://example.com/vms/foo/vm1

OCCI Rendering

- **Retrieving All resource instances Belonging to Mixin or Kind**
  - > GET /compute/ HTTP/1.1
  - > [...]
  - < HTTP/1.1 200 OK
  - < [...]
  - <
  - < X-OCCI-Location: http://example.com/vms/foo/vm1
  - < X-OCCI-Location: http://example.com/vms/foo/vm2
  - < X-OCCI-Location: http://example.com/vms/bar/vm1

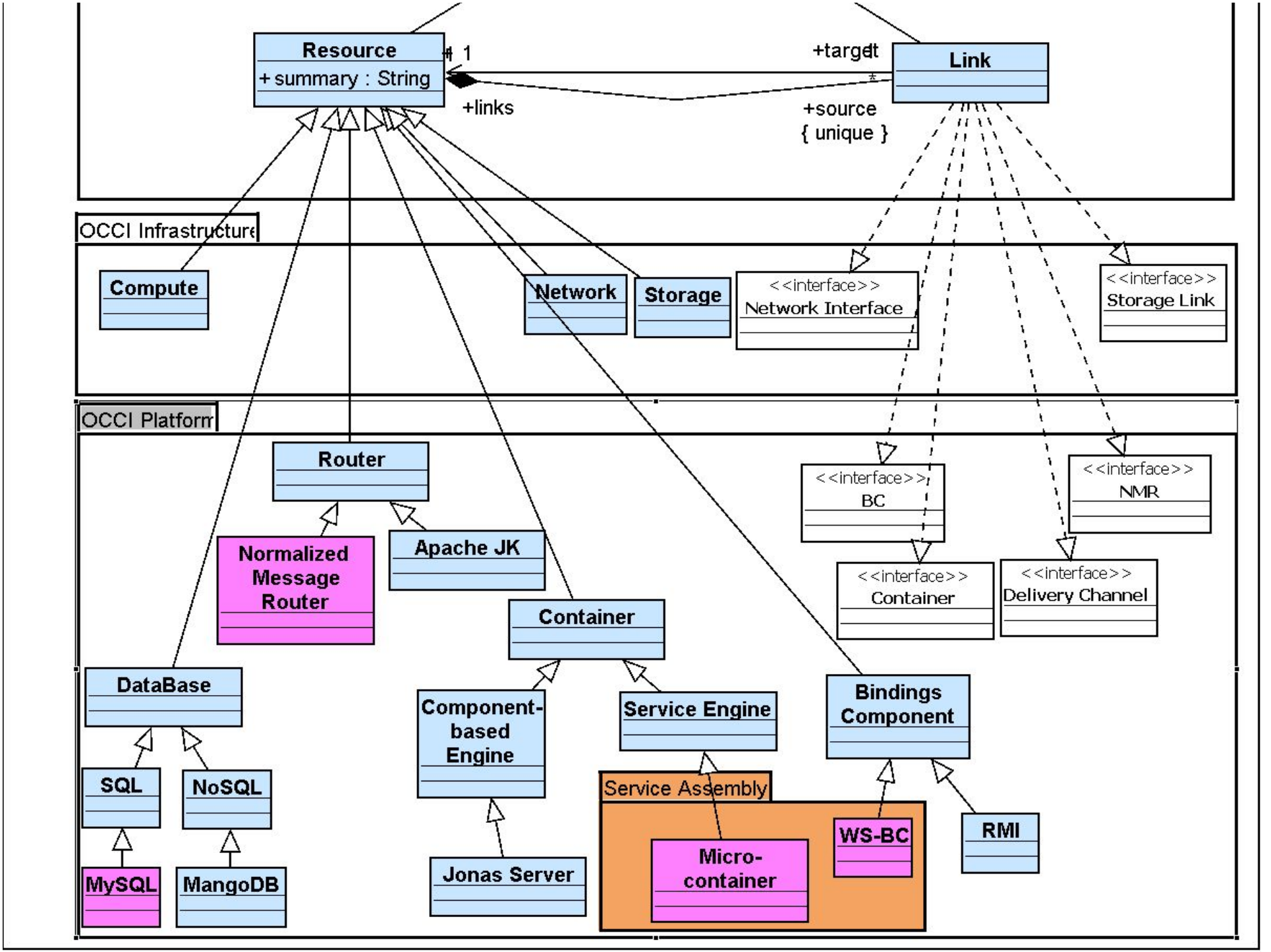
- **Triggering Actions on All Instances of a Mixin or Kind**
  - > POST /compute/?action=stop HTTP/1.1
  - > [...]
  - > Category: stop; scheme="[...]"; class="action";
  - > X-OCCI-Attribute: method="poweroff“

OCCI Rendering

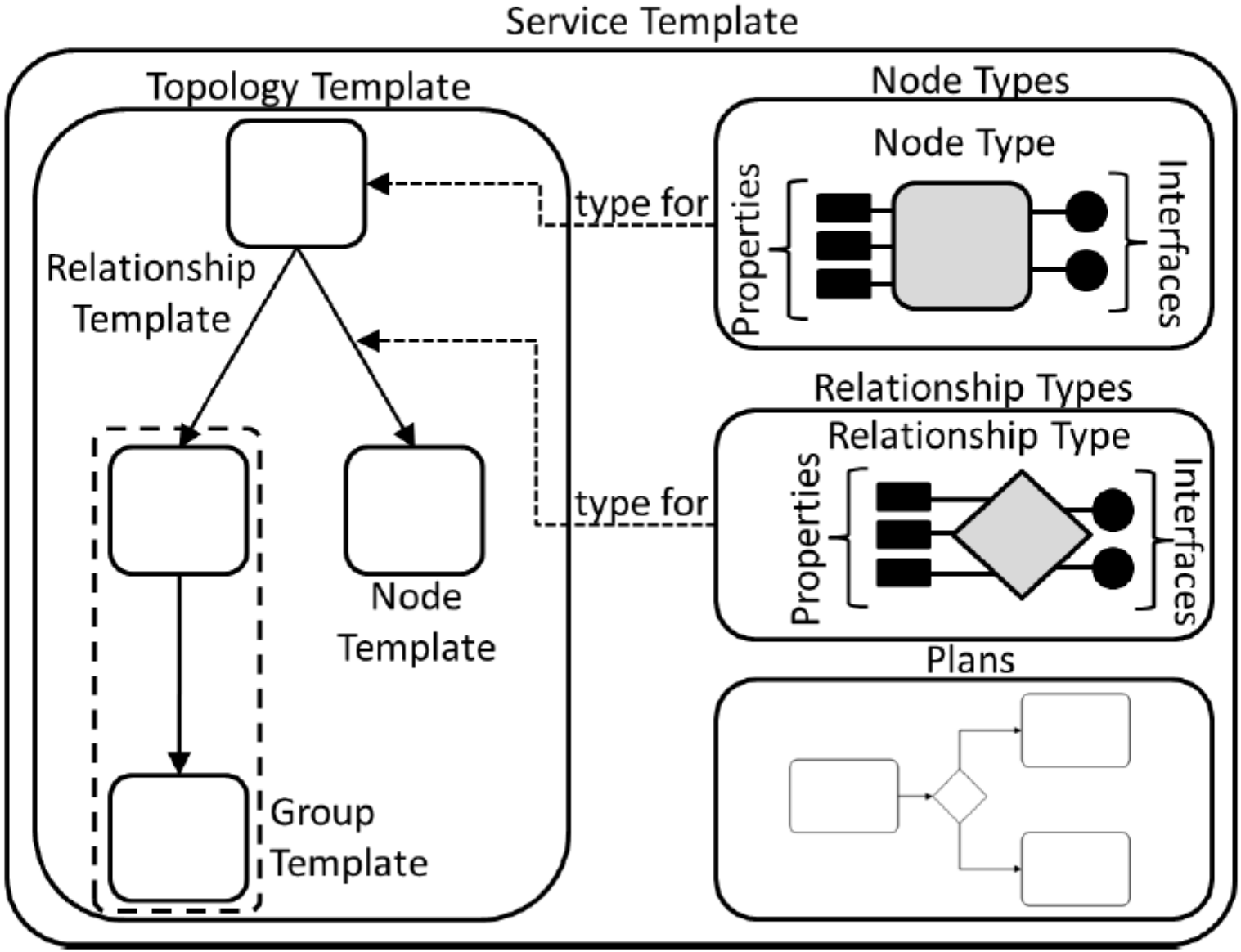
- **Triggering an Action on a resource instance**
  - > POST /vms/foo/vm1?action=stop HTTP/1.1
  - > [...]
  - > Category: stop; scheme="[...]"; class="action";
  - > X-OCCI-Attribute: method="poweroff"
  - < HTTP/1.1 200 OK
  - < [...]

OCCI Rendering

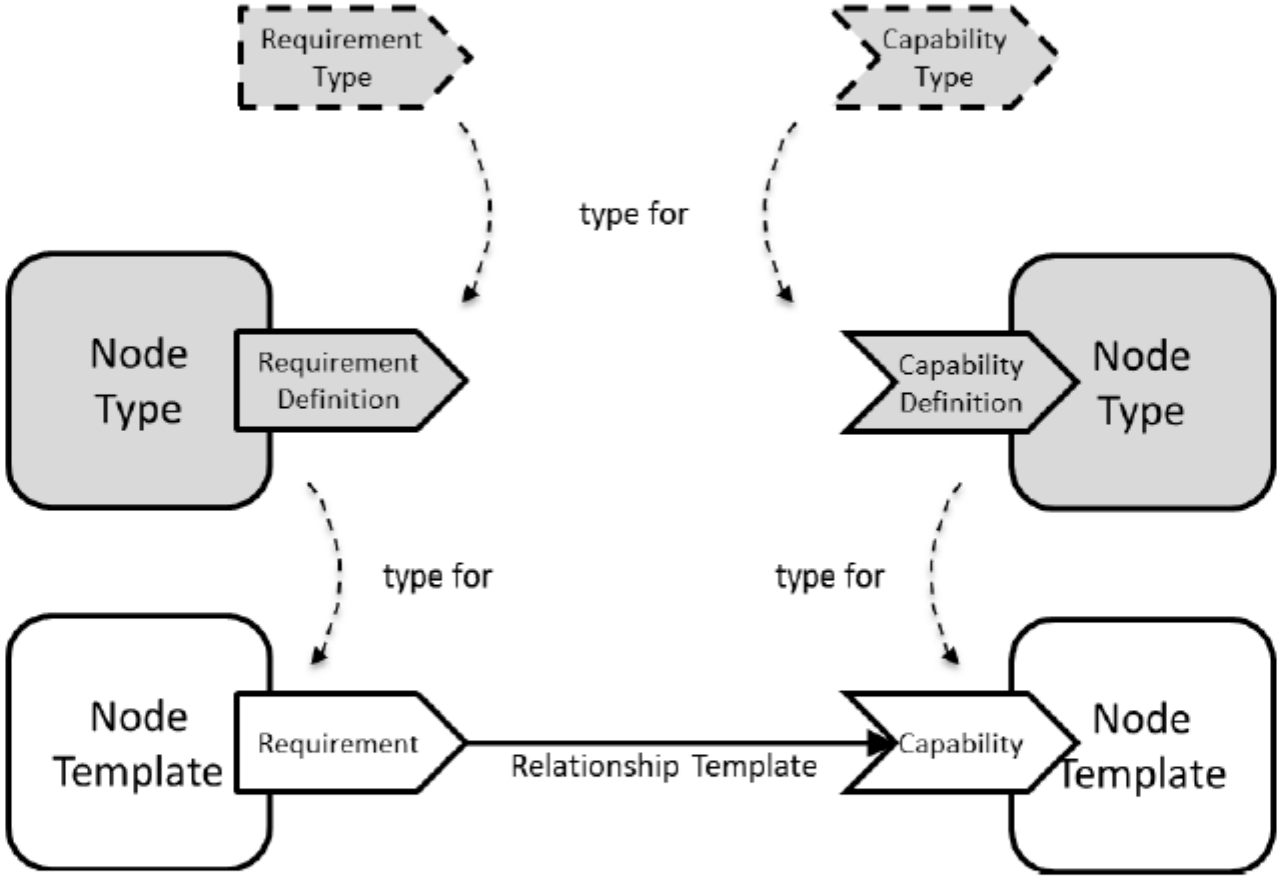
- **Inline Creation of a Link Instance**
  - > POST /compute/ HTTP/1.1
  - > [...]
  - > Category: compute;
  - scheme="http://schemas.ogf.org/occi/infrastructure#";
  - class="kind";
  - > Link: </network/123>;
  - rel="http://schemas.ogf.org/occi/infrastructure#network";
  - category="http://schemas.ogf.org/occi/infrastructure#networkinterface";
  - occi.networkinterface.interface="eth0";
  - occi.networkinterface.mac="00:11:22:33:44:55";
  - > X-OCCI-Attribute: occi.compute.cores=2
  - > X-OCCI-Attribute: occi.compute.hostname="foobar"
  - > [...]
  - < HTTP/1.1 200 OK
  - < [...]
  - < Location: http://example.com/vms/foo/vm1


Resource
+ summary : String
1
+links
+target
*
Link
+source
{ unique }
OCCI Infrastructure
Compute
Network
Storage
<<interface>>
Network Interface
<<interface>>
Storage Link
OCCI Platform
Router
Normalized Message Router
Apache JK
<<interface>>
BC
<<interface>>
NMR
<<interface>>
Container
<<interface>>
Delivery Channel
Container
DataBase
Component-based Engine
Service Engine
Bindings Component
SQL
NoSQL
Service Assembly
MySQL
MangoDB
Jonas Server
Micro-container
WS-BC
RMI

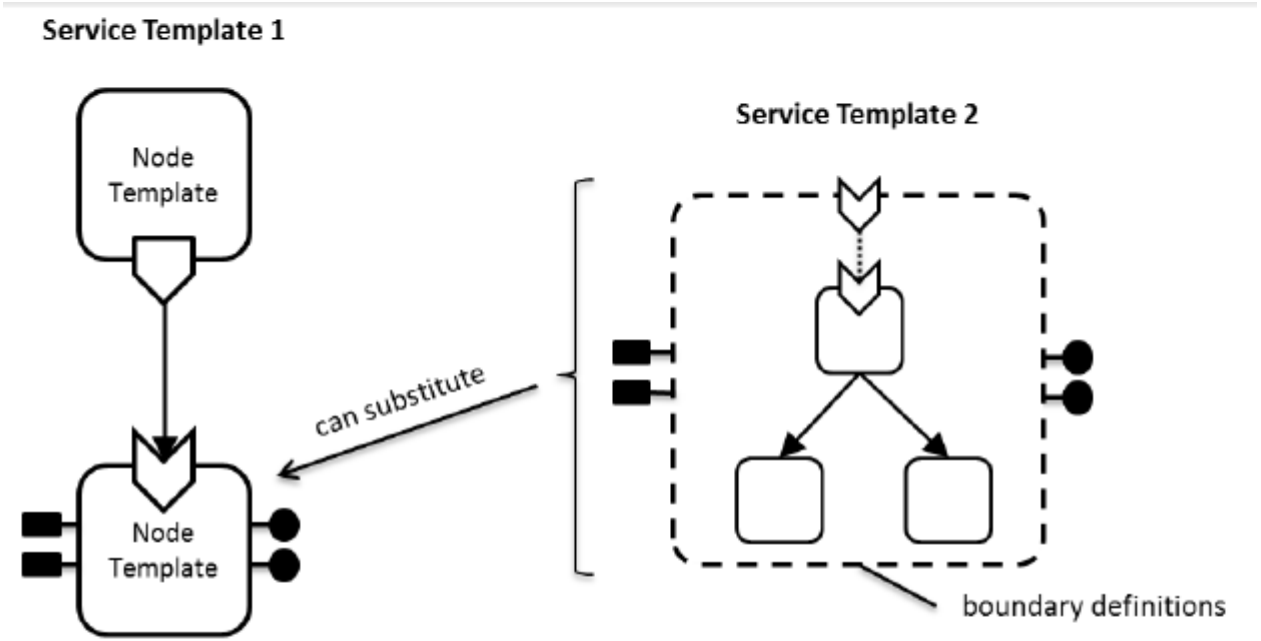
TOSCA: Structural Elements of a Service Template and their Relations



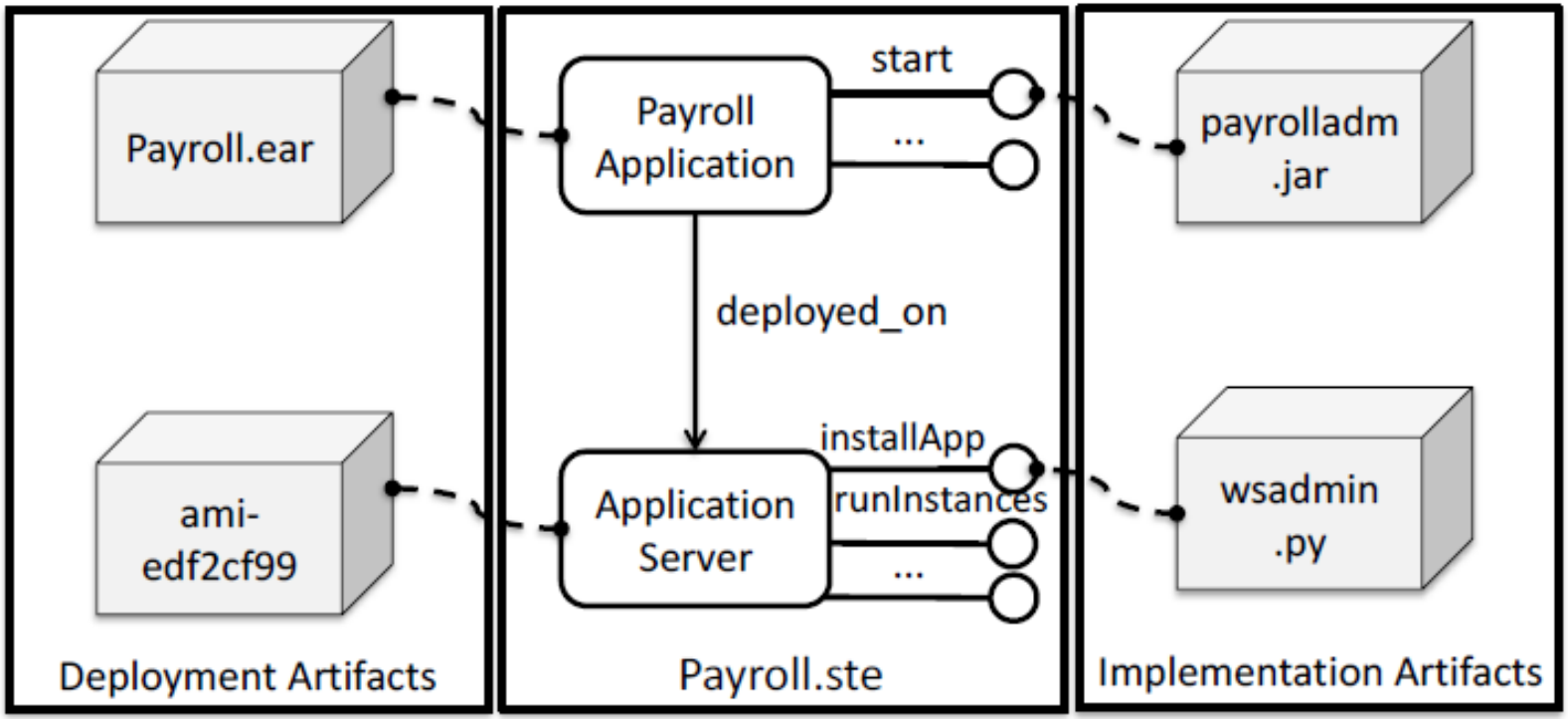
TOSCA: Requirements and Capabilities



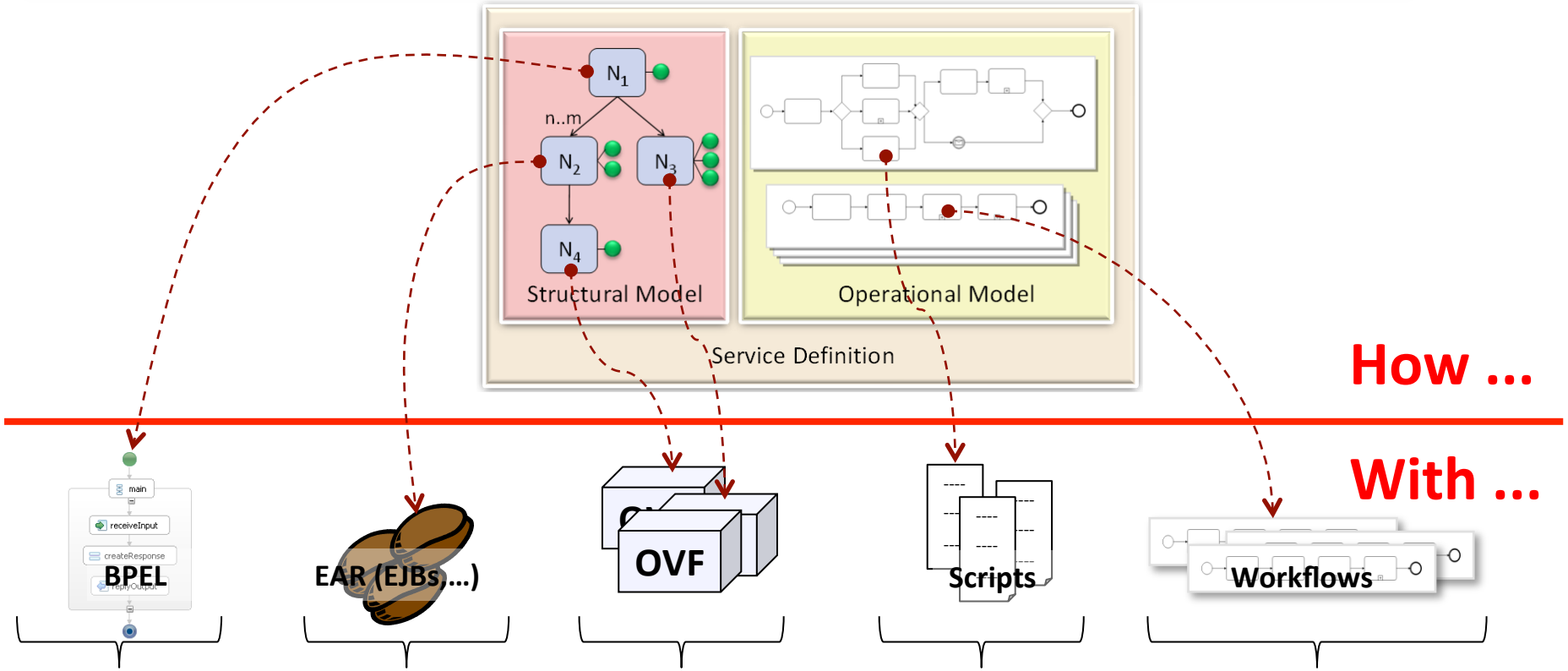
TOSCA: Composition of Service Templates



TOSCA: sample



TOSCA Refined View



The business processes of the application (BPEL, BPMN, Human Tasks,...)

The business logic of the application, e.g. EJBs, JSPs, JPEG,...

The images of the middleware (DB2, Websphere,...) required to run the application

(Existing) scripts used by task of plans to manage the cloud application

(Existing) workflows used by subprocess-tasks of plans

Cloud Platforms: Open issues

Open issues

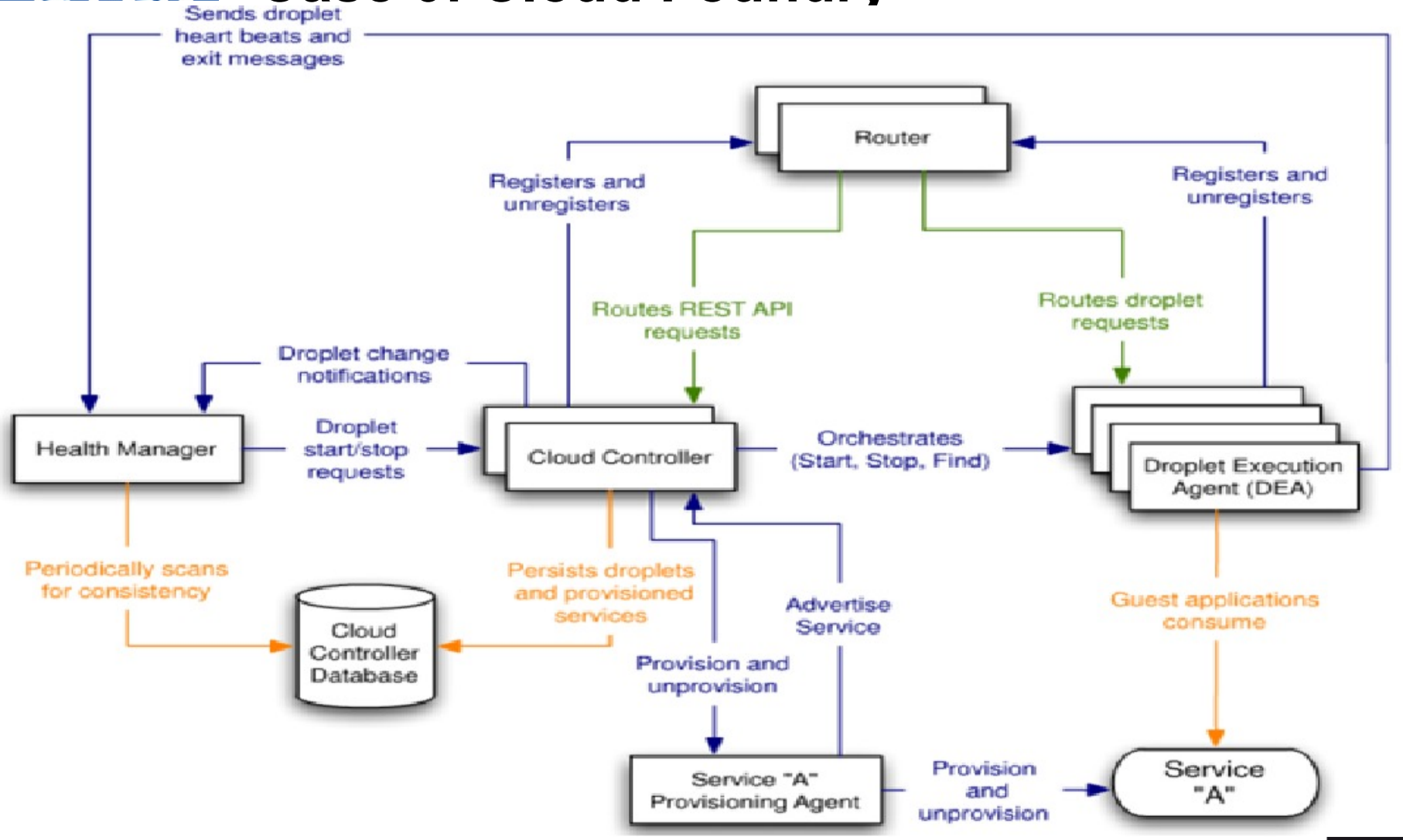
- Design time
  - Software development
- Deployment
  - Several public clouds
  - Hybrid clouds
- Run time
  - Elasticity
  - Federation
- Portability, Migration & Mobility
- Simulation

Cloud Platforms: Software development

Application development

- **Traditional platforms**
  - Focus non business
  - Benefit from non functional properties
- **Cloud platform**
  - Case 0: development unchanged
    - Elasticity (engine granularity)
  - Case 1: API-based developments
    - Lock-in
  - Case 2: plain development
    - Change of practices
  - Case 3: hybrid ?

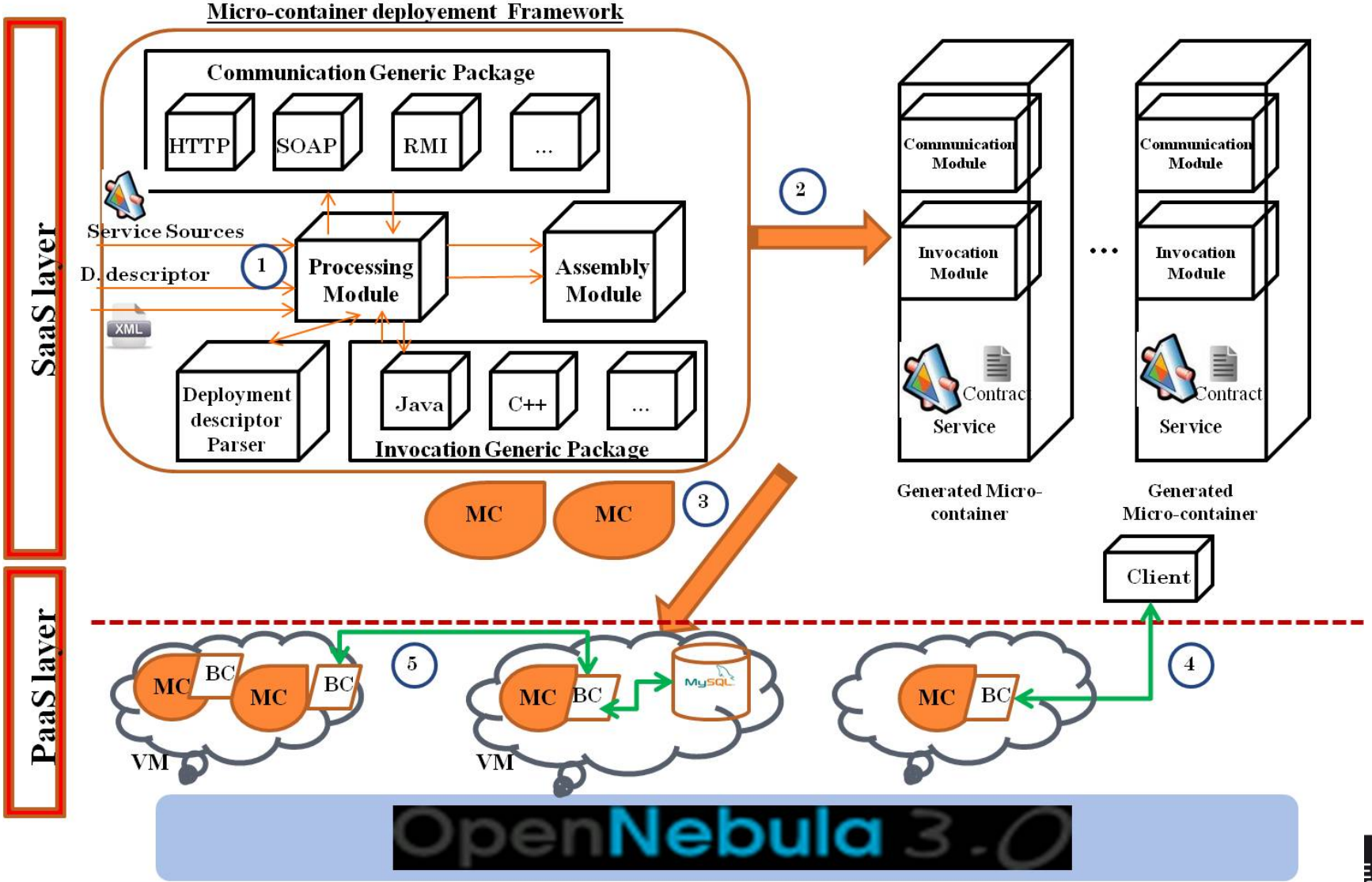
Case 0: Cloud Foundry



Case 1: API-based developments

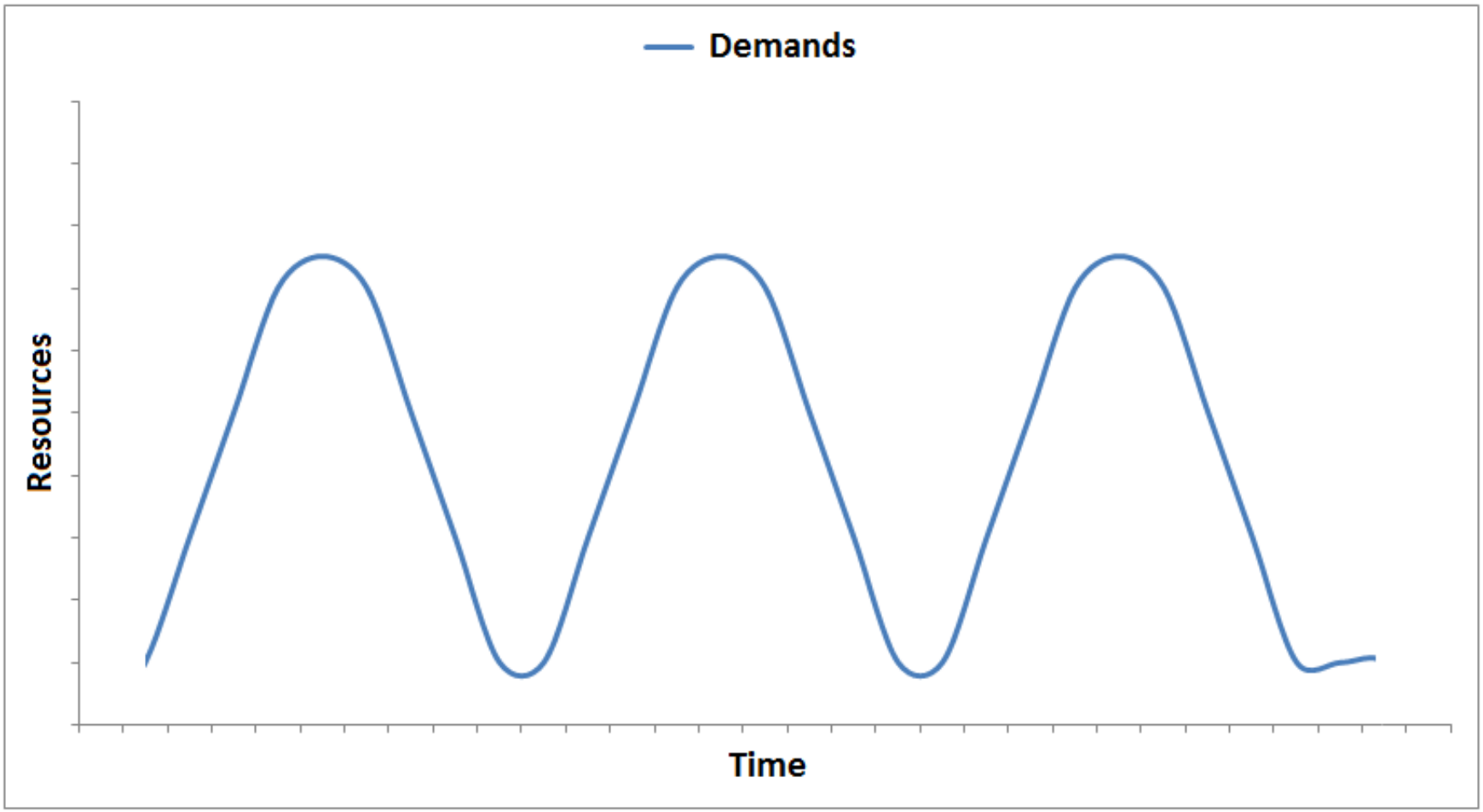
- **Google App Engine API**
  - provisioning, reporting, and migration, as well as manipulating data in Calendar and Spreadsheets

Case 2: CloudServ

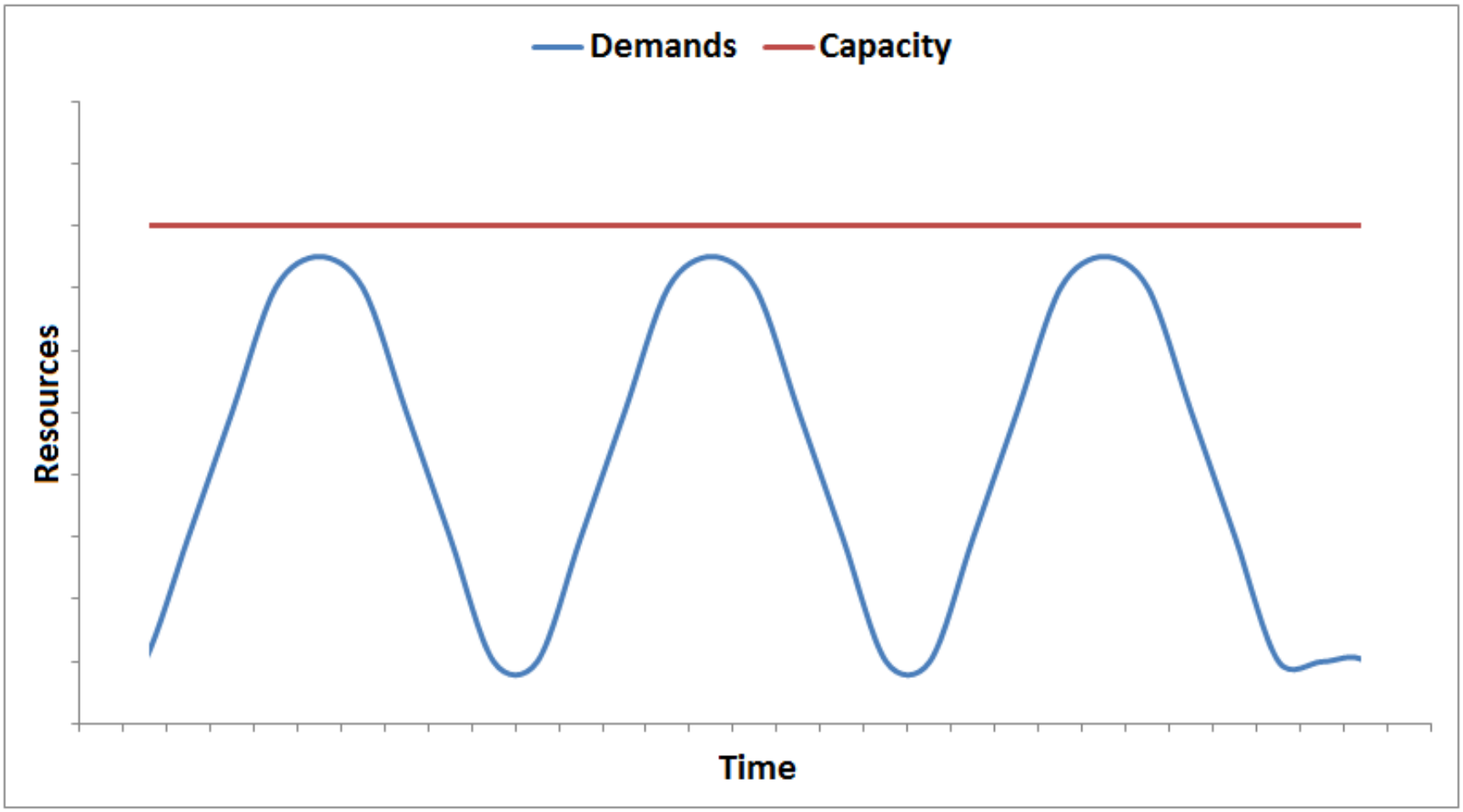


Cloud Platforms: Elasticity

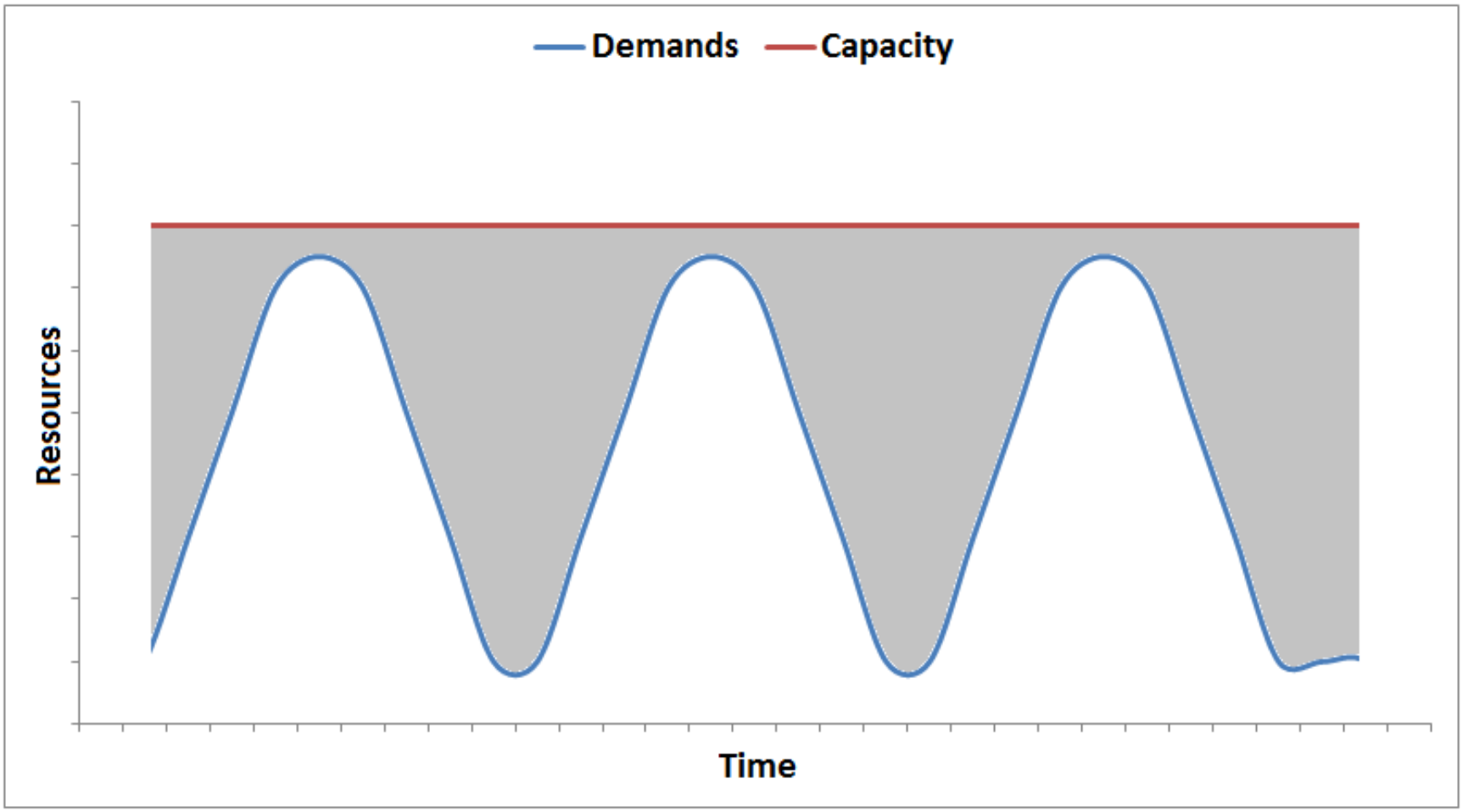
Elasticity principals



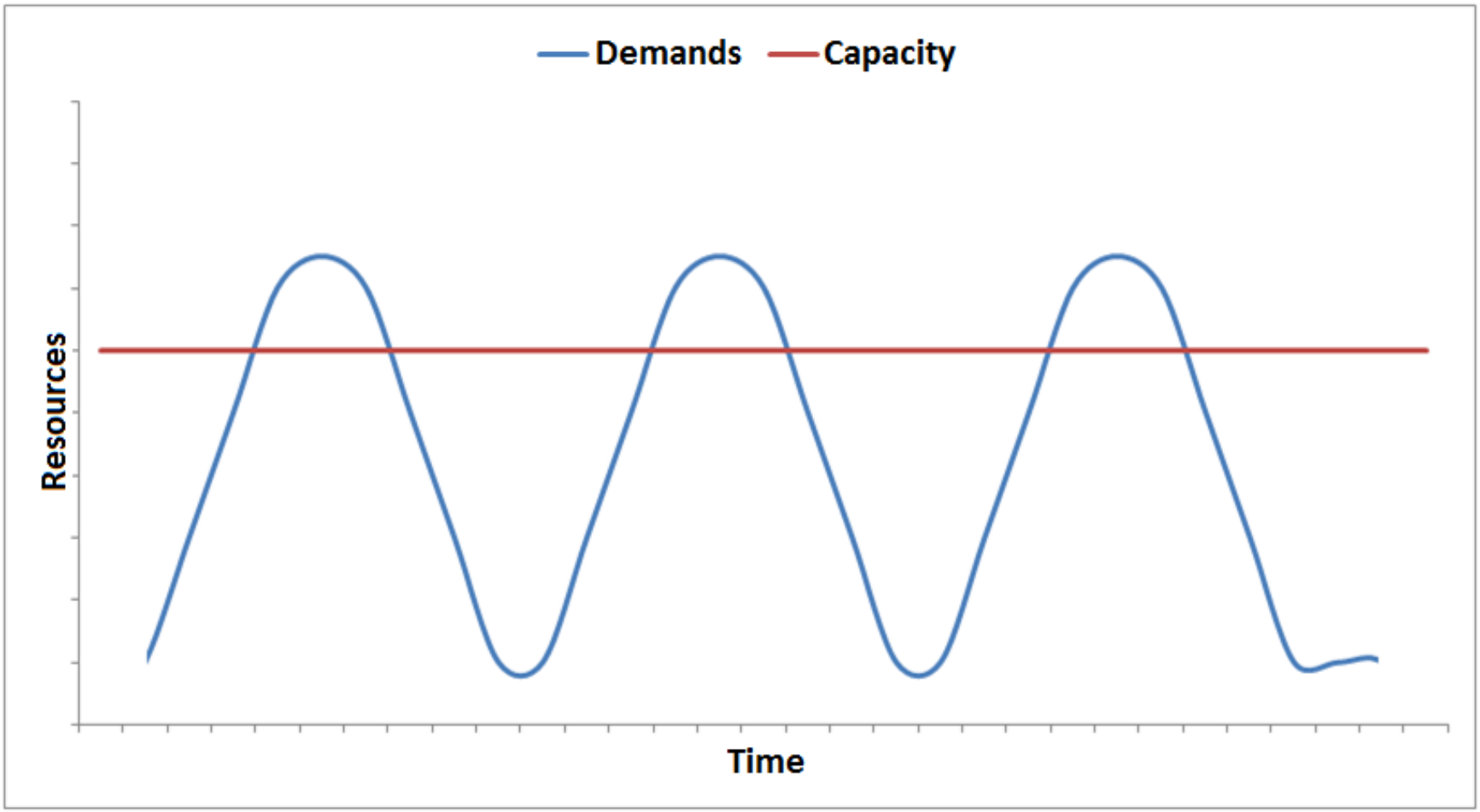
Elasticity principals



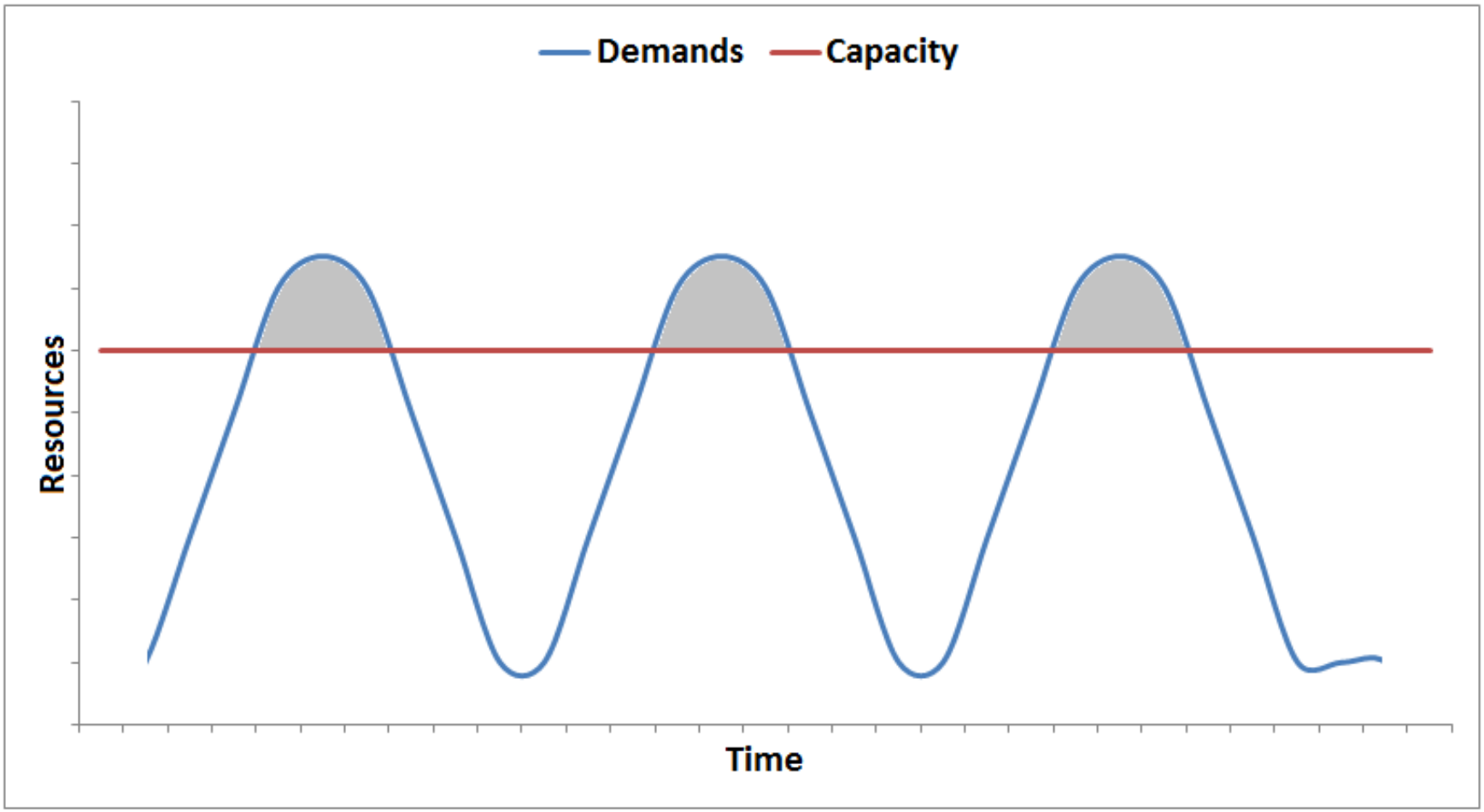
Elasticity principals



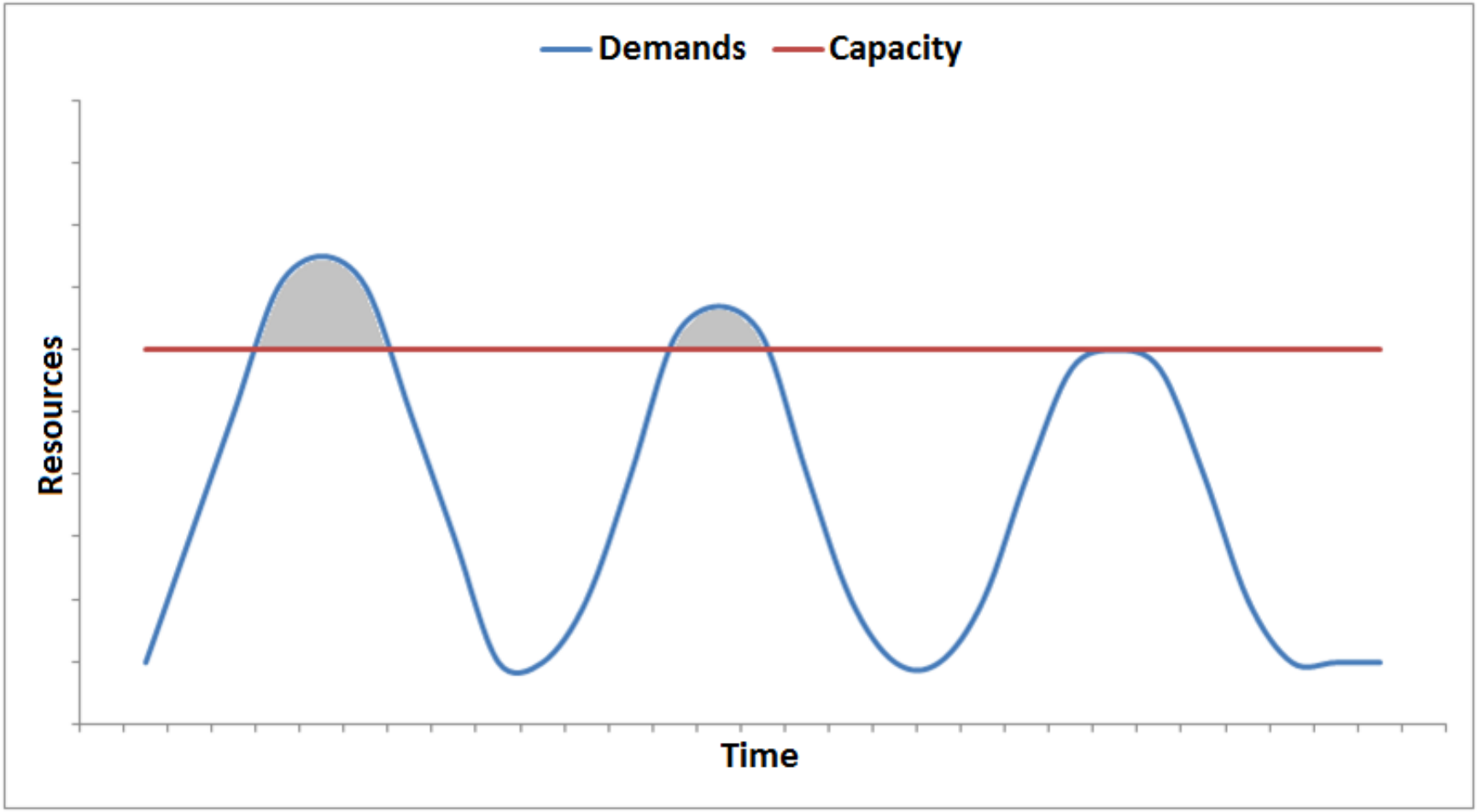
Elasticity principals



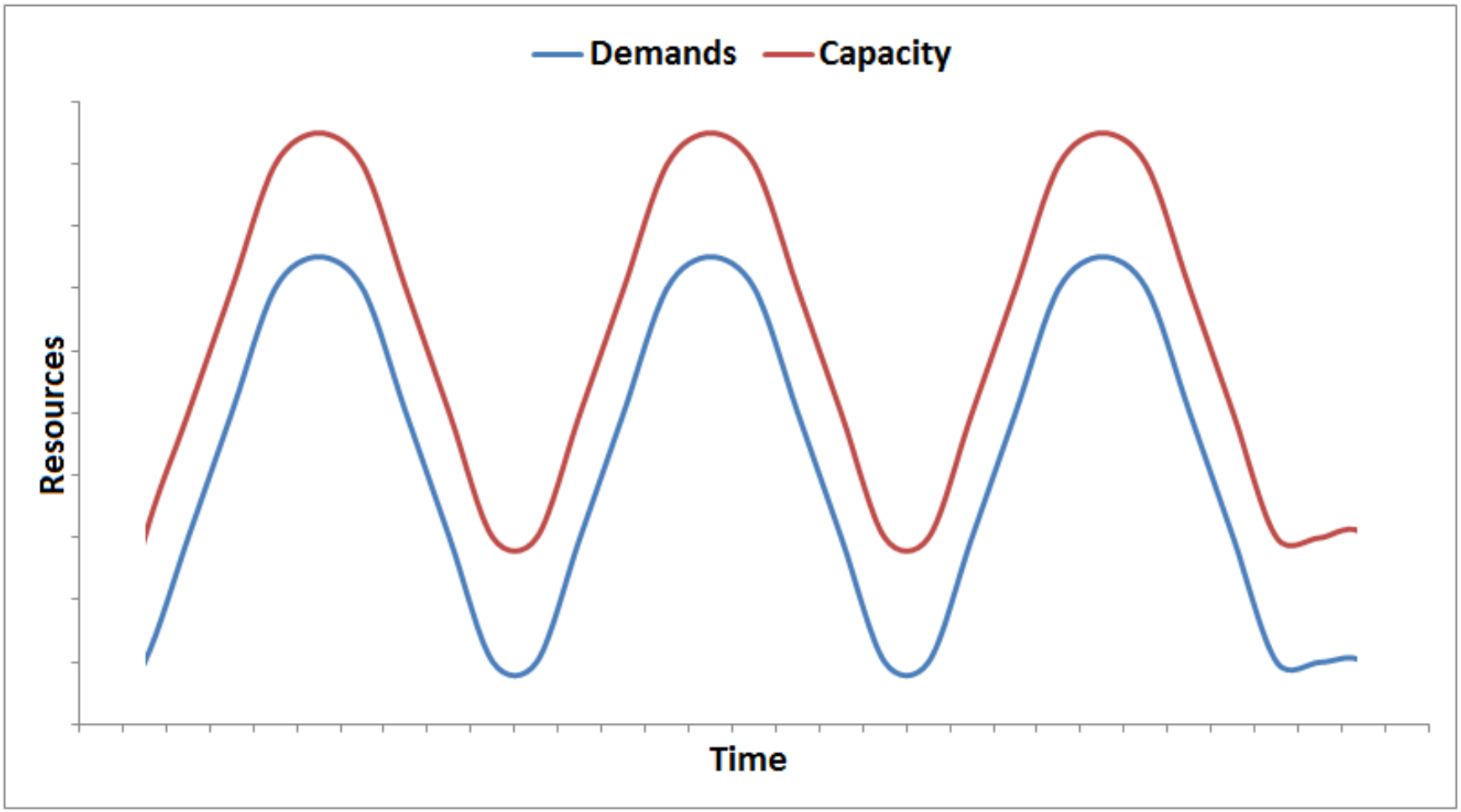
Elasticity principals



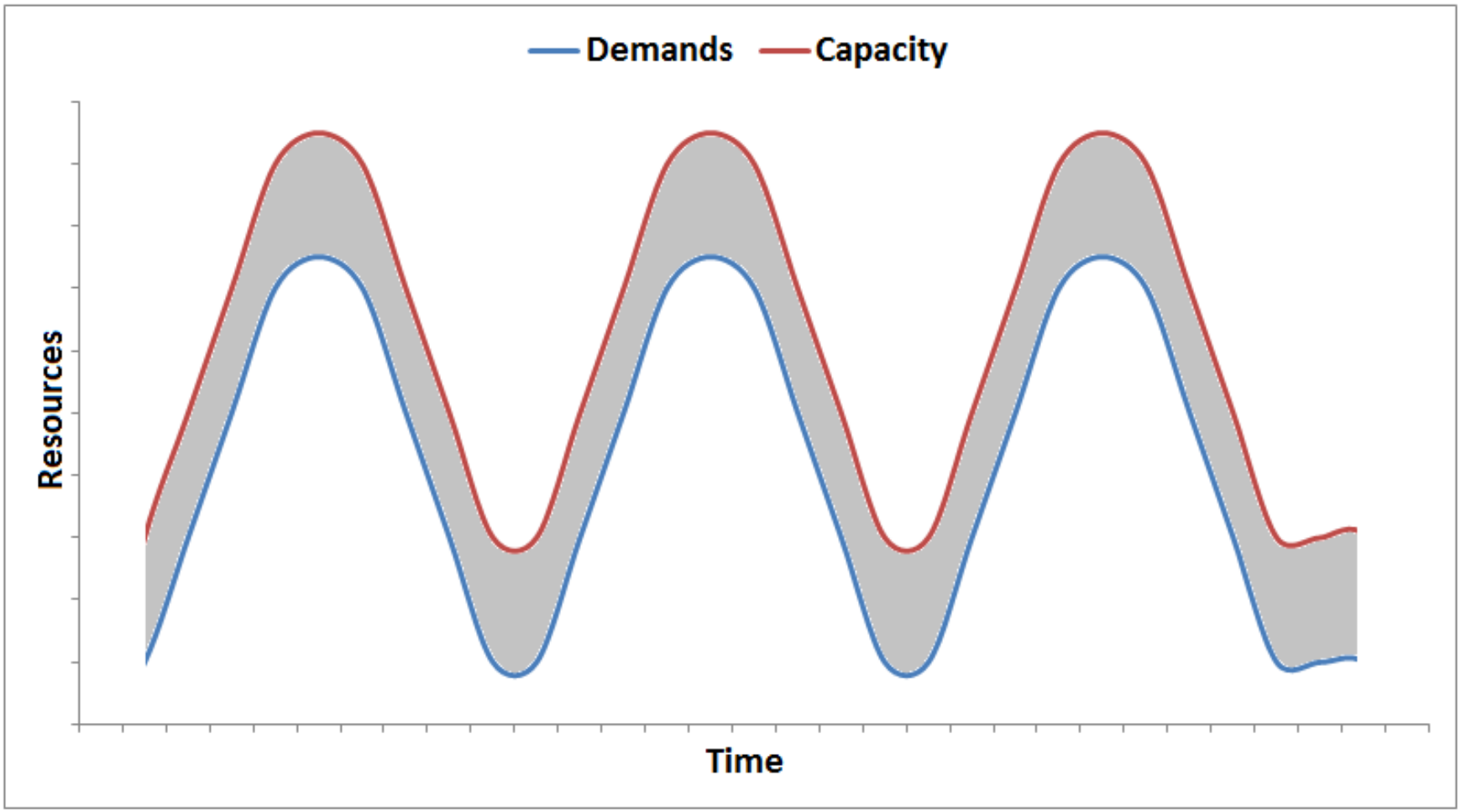
Elasticity principals



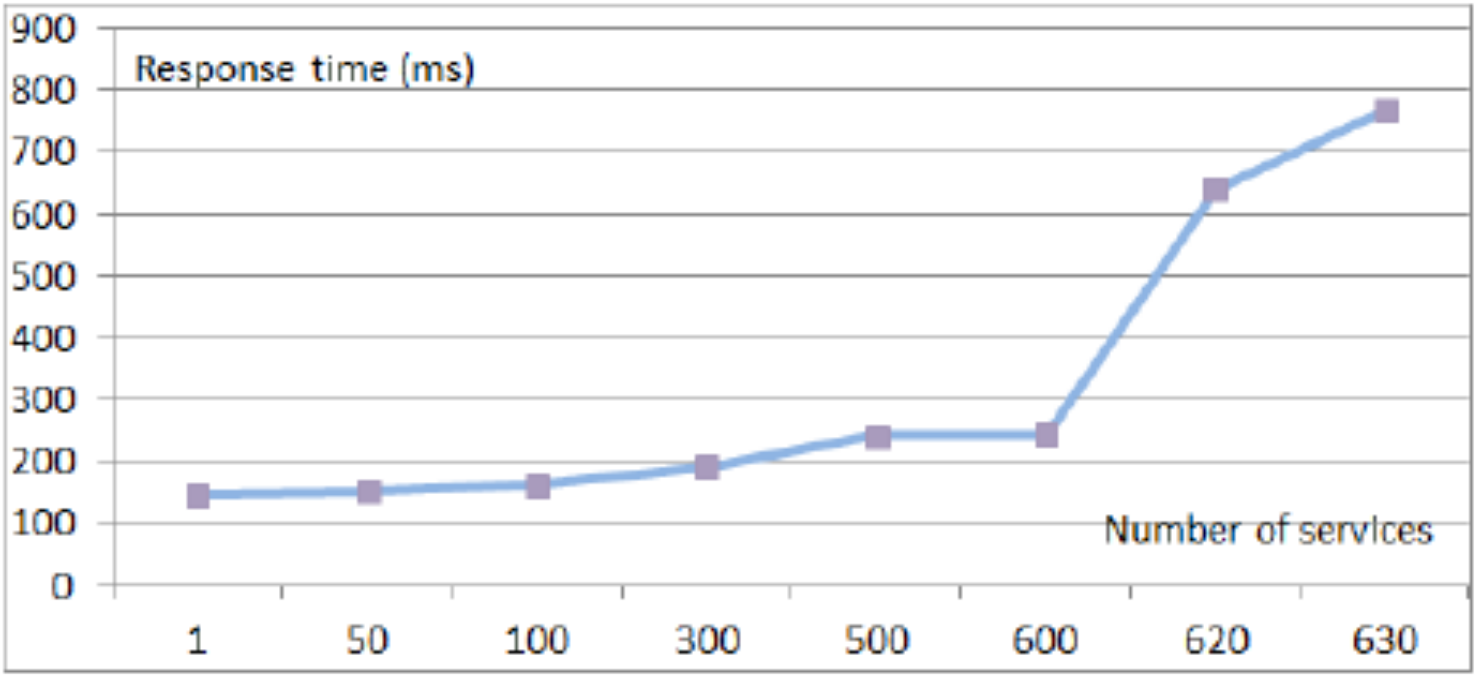
Elasticity principals



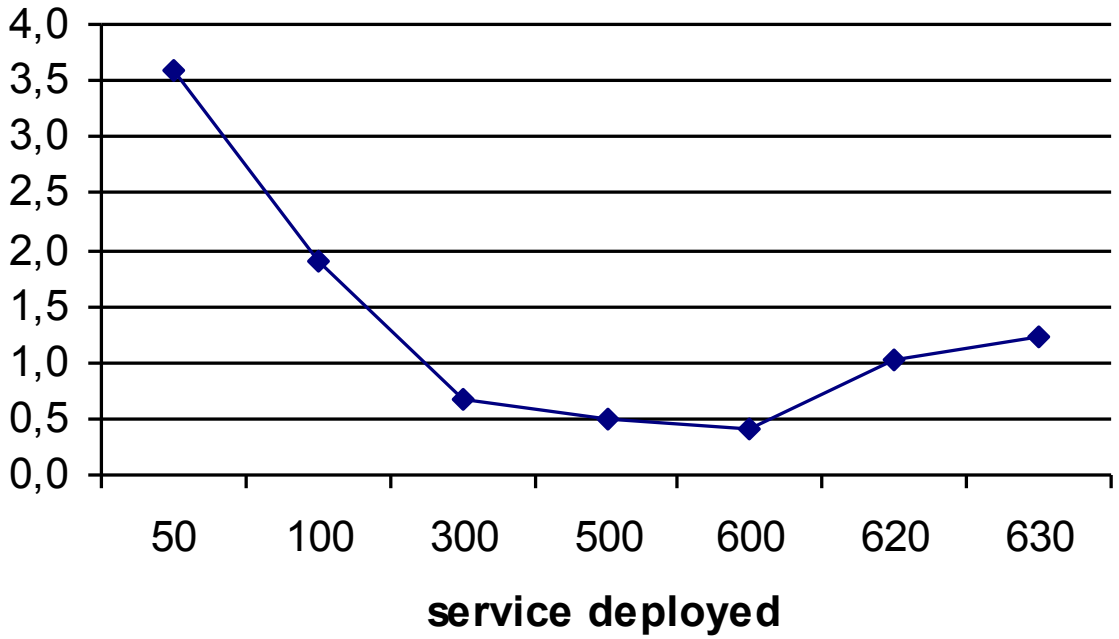
Elasticity principals



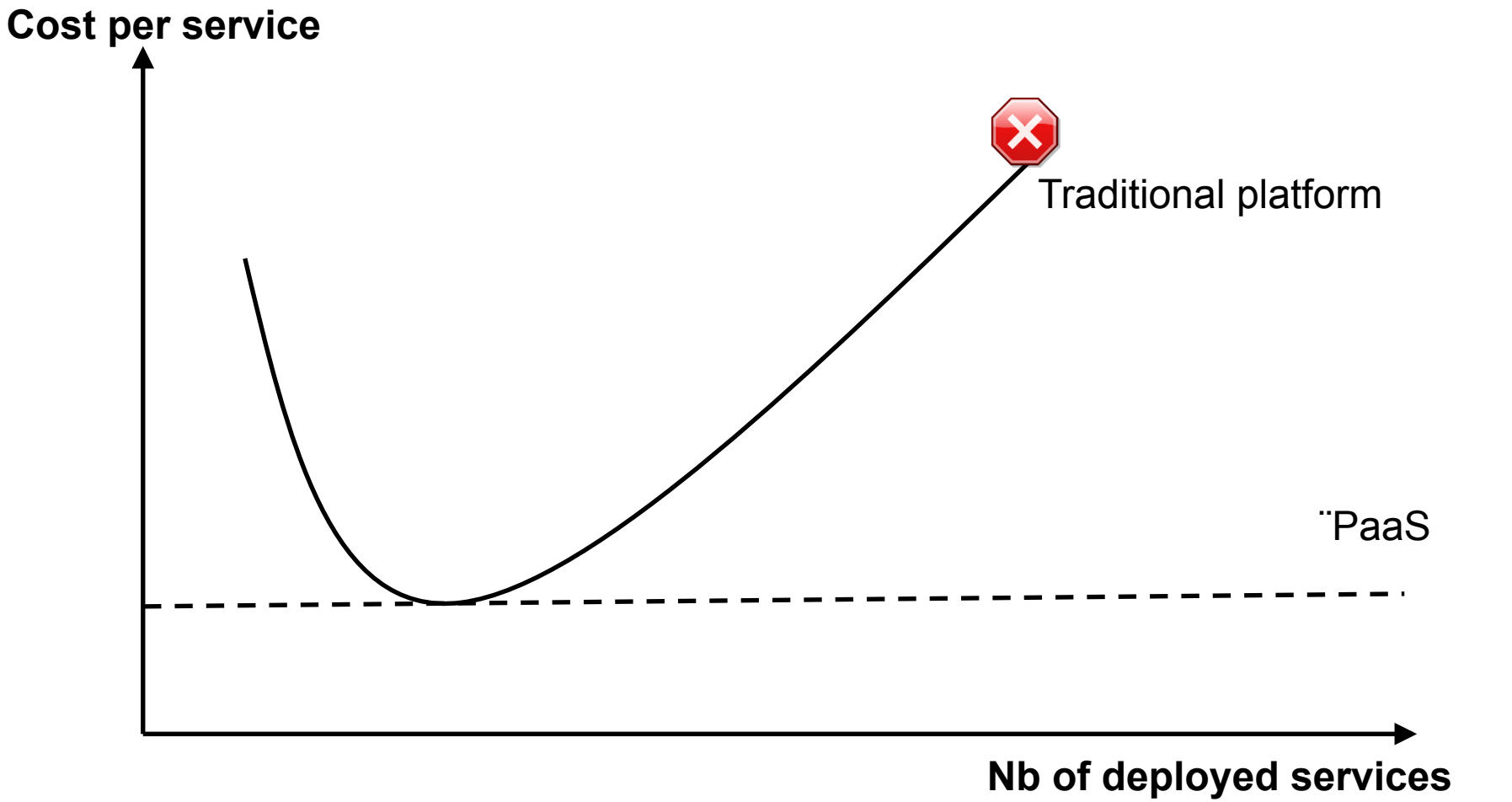
The case of a Web service container



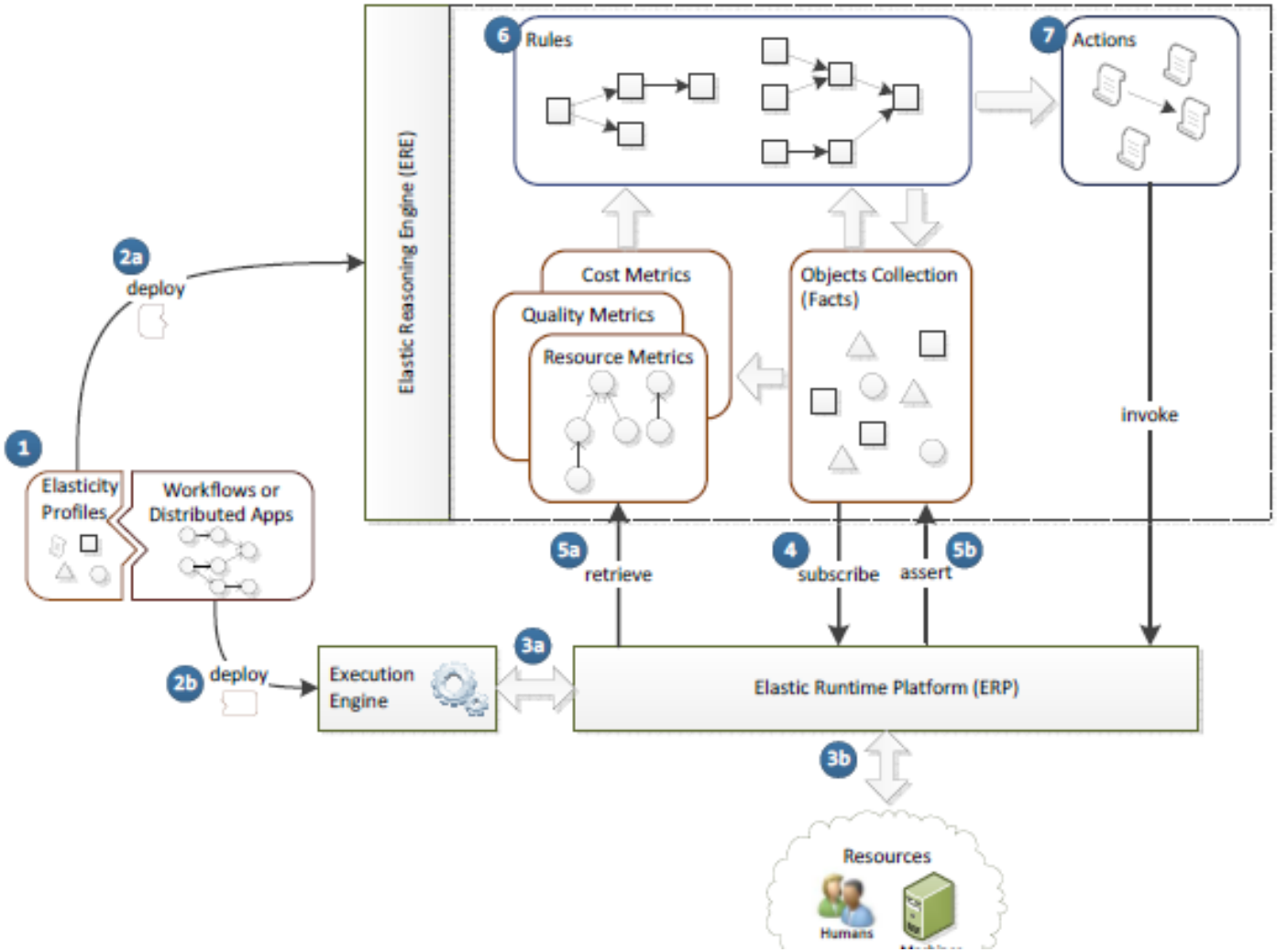
Cost per service ?



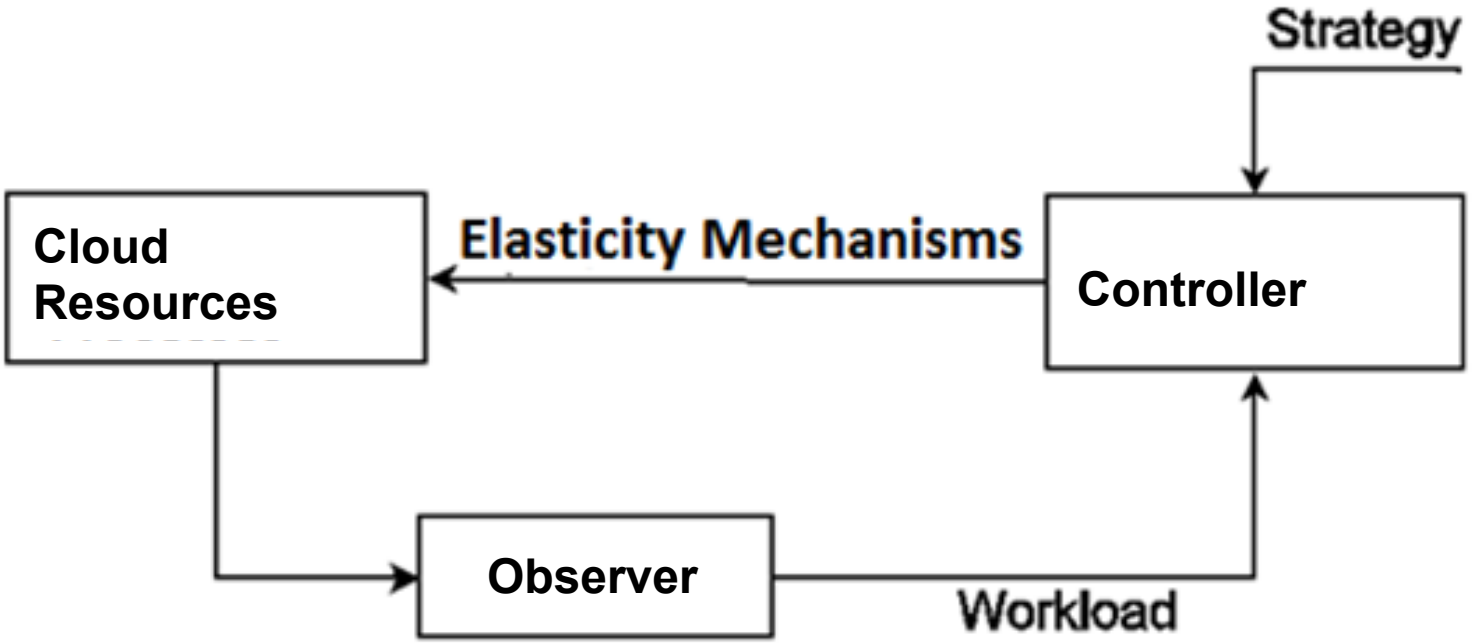
Platform signature



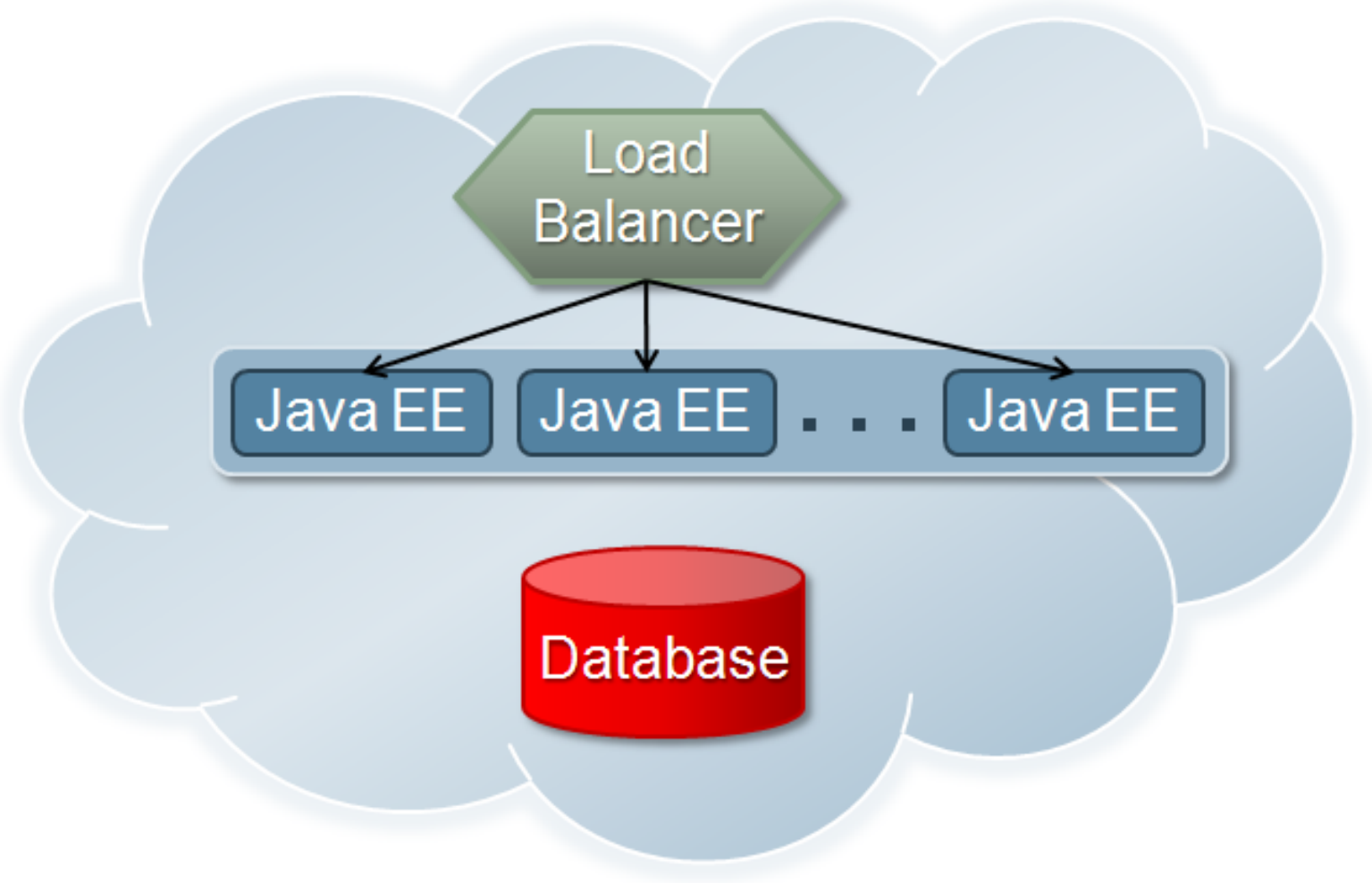
Elasticity Modeling Conceptual Framework (VUT)



Elasticity principal



Example: JEE Elasticity

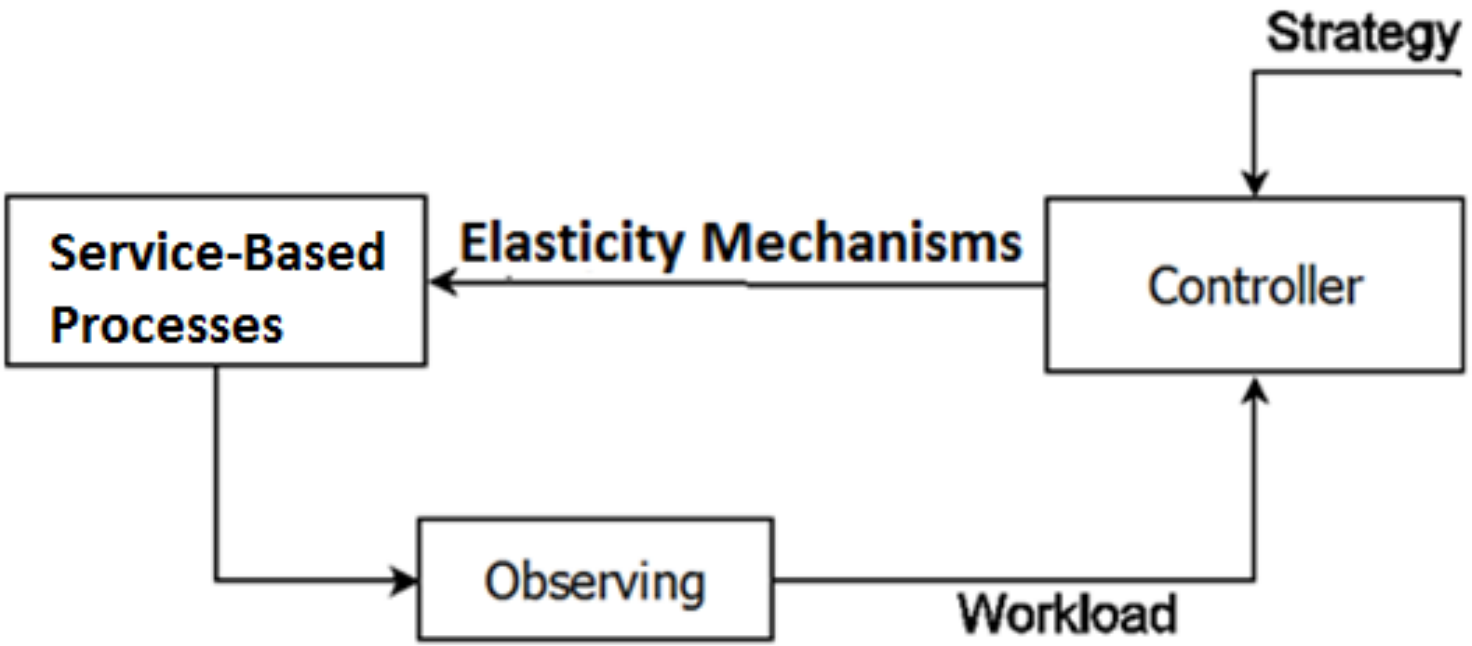


Elasticity in GlasFish

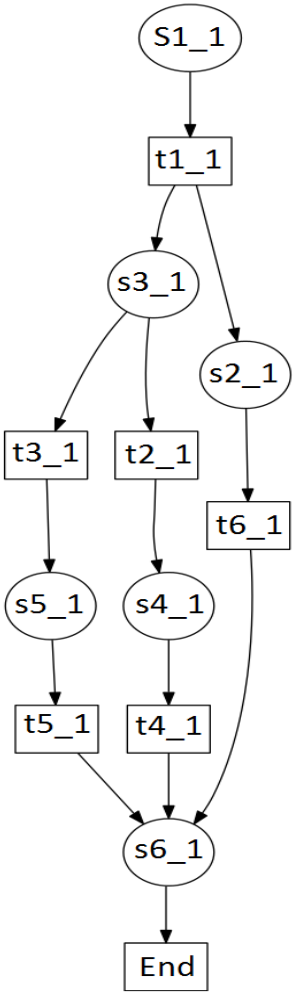
- **Dynamic Service Provisioning:**
  - service dependencies are discovered by introspecting the application archive
  - required services such as Java EE, Database, and Load Balancer are provisioned.
    - These services can be dedicated to the application or shared by different applications

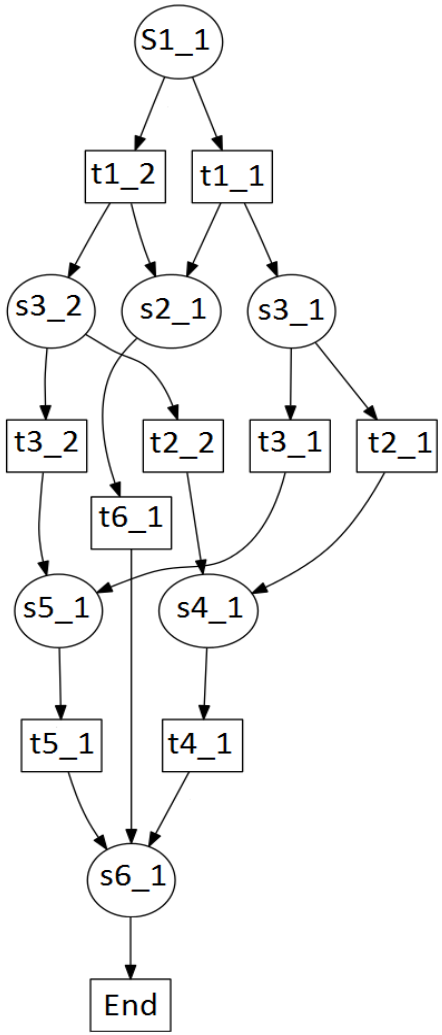
    .
- **Elasticity using Auto-scaling**
  - Monitoring application flows
  - Indicators aggregation
  - Automatically resizing to meet the growing demands using auto-scaling.

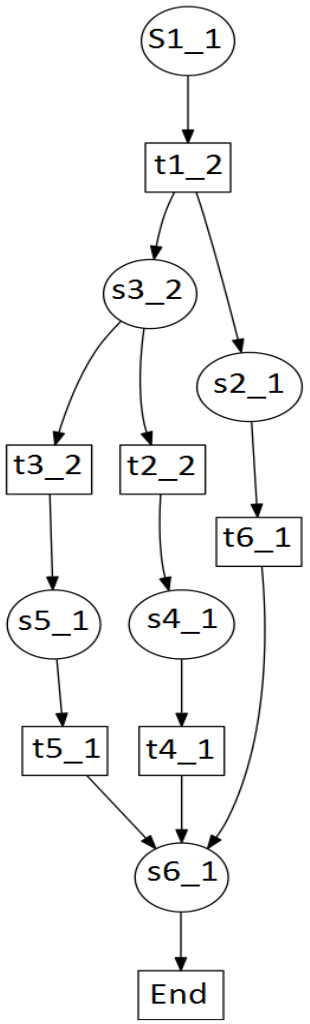
Example: BPM elasticity



Elasticity at the SaaS level

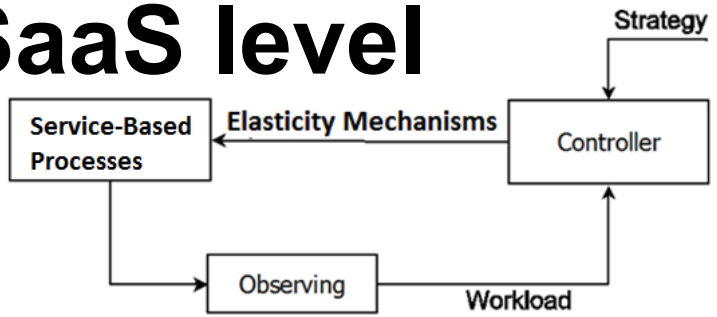


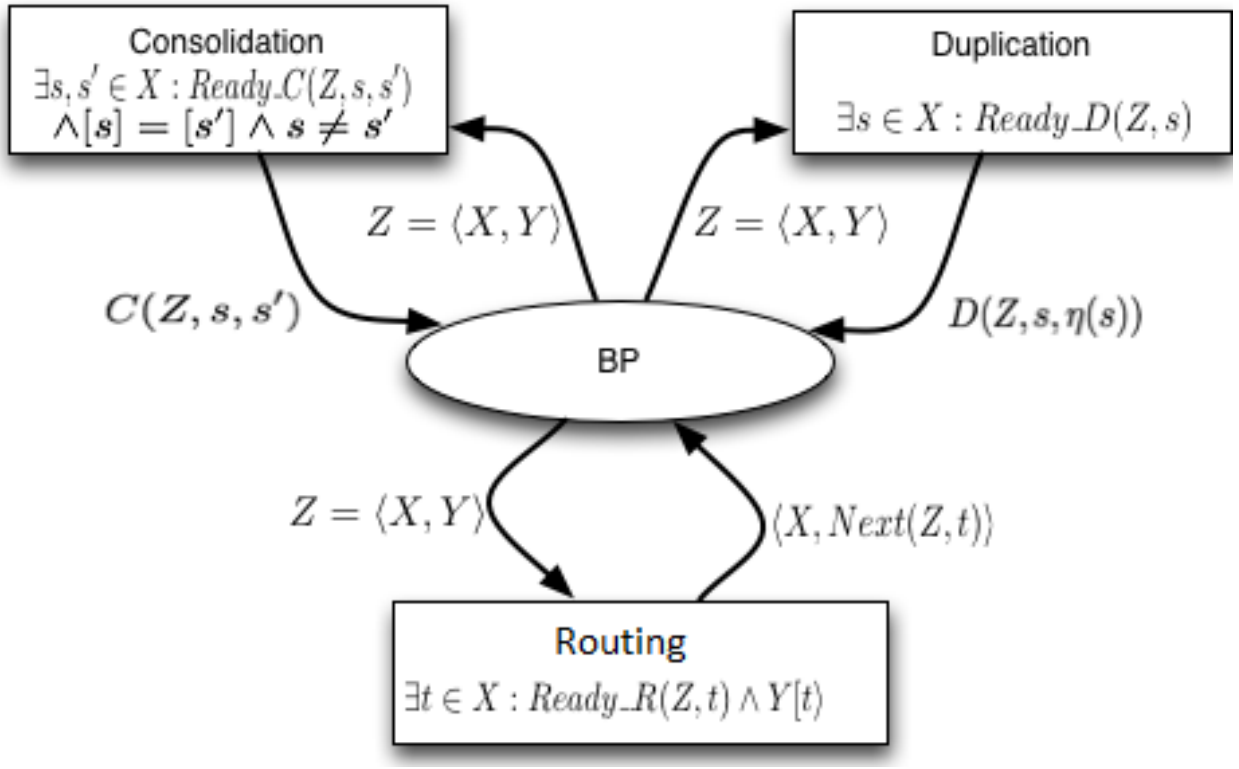




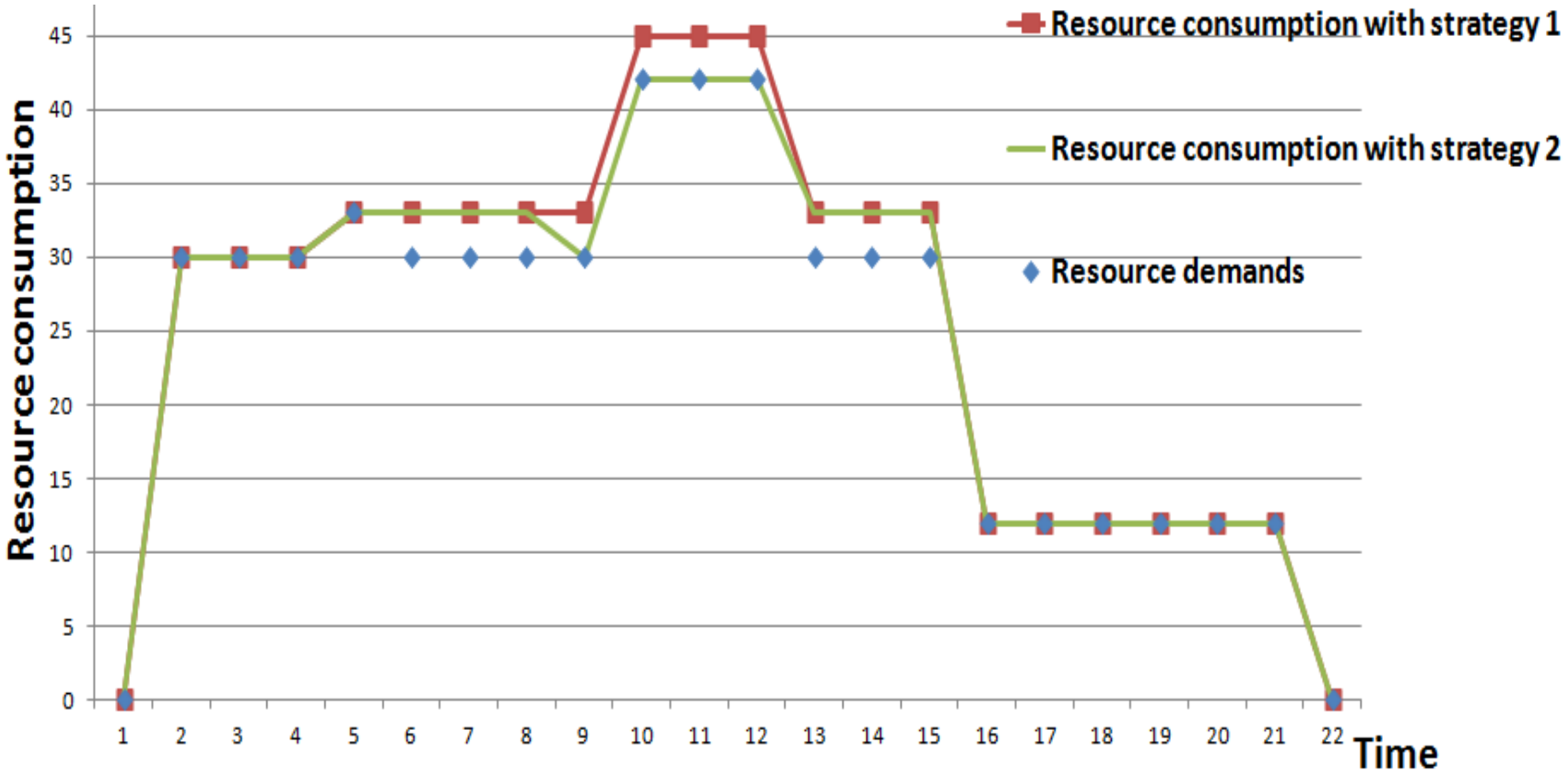
Initial state **Duplication(S, s3_1, s3_2)** **Consolidation(S, s3_2, s3_1)**

Elasticity at the SaaS level

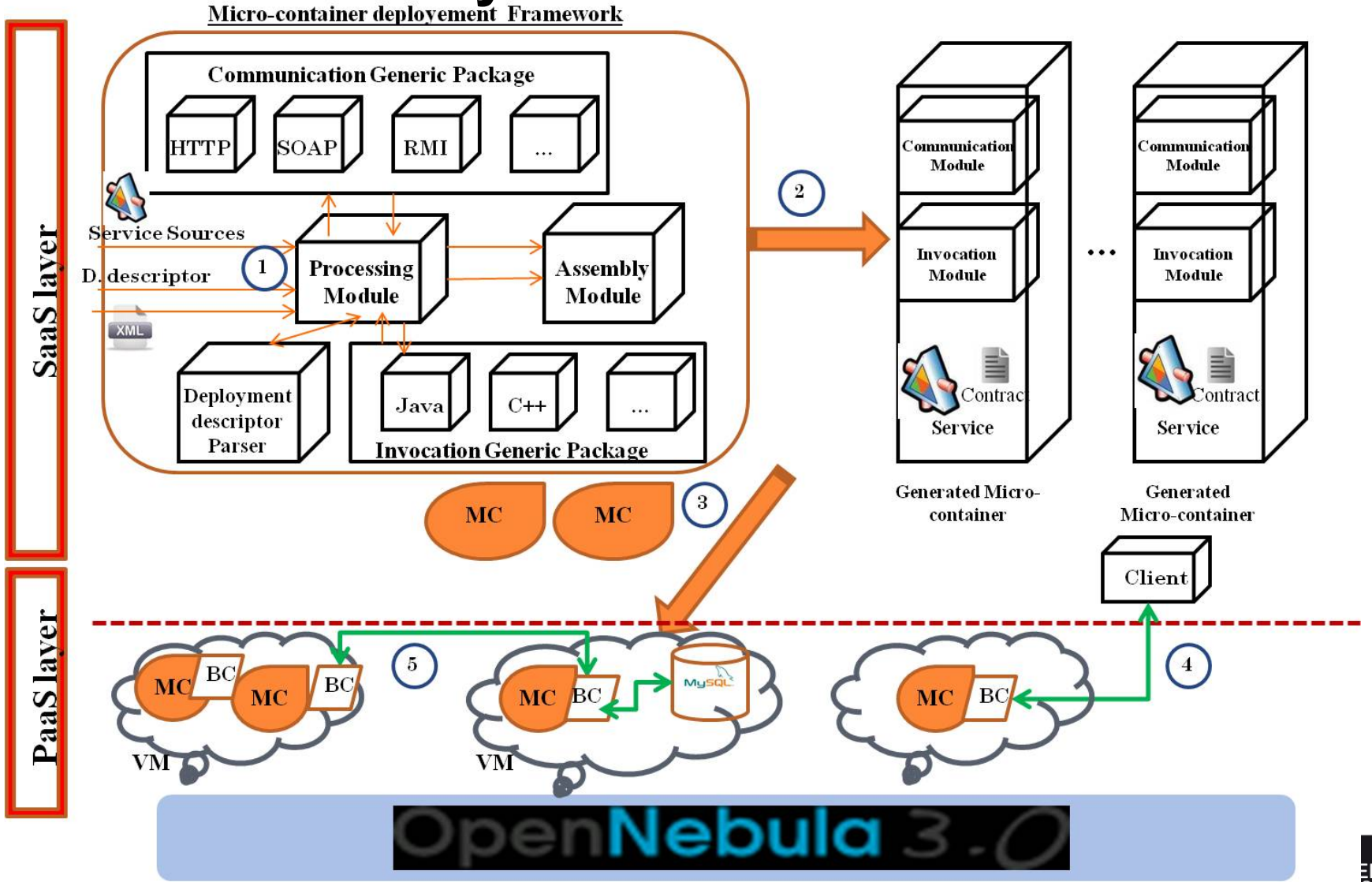




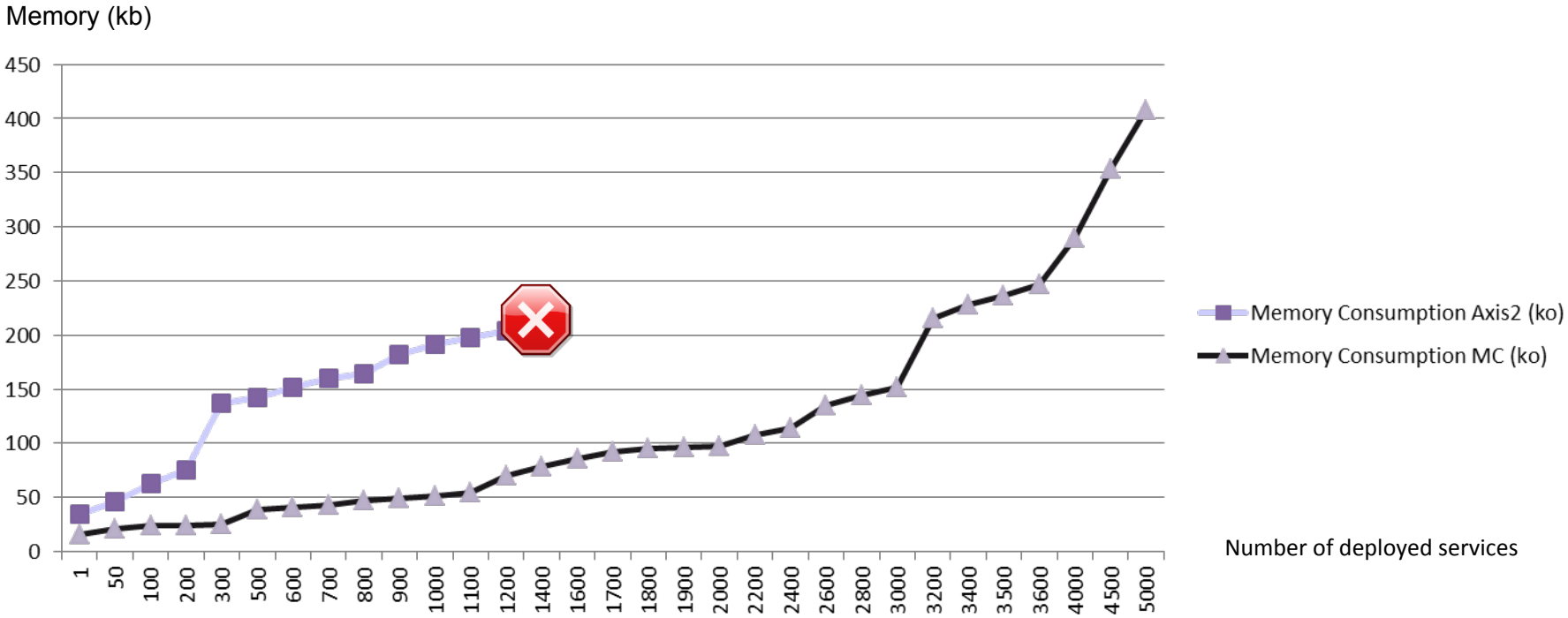
Elasticity at the SaaS level



Elasticity in CloudServ



CloudServ Experiments

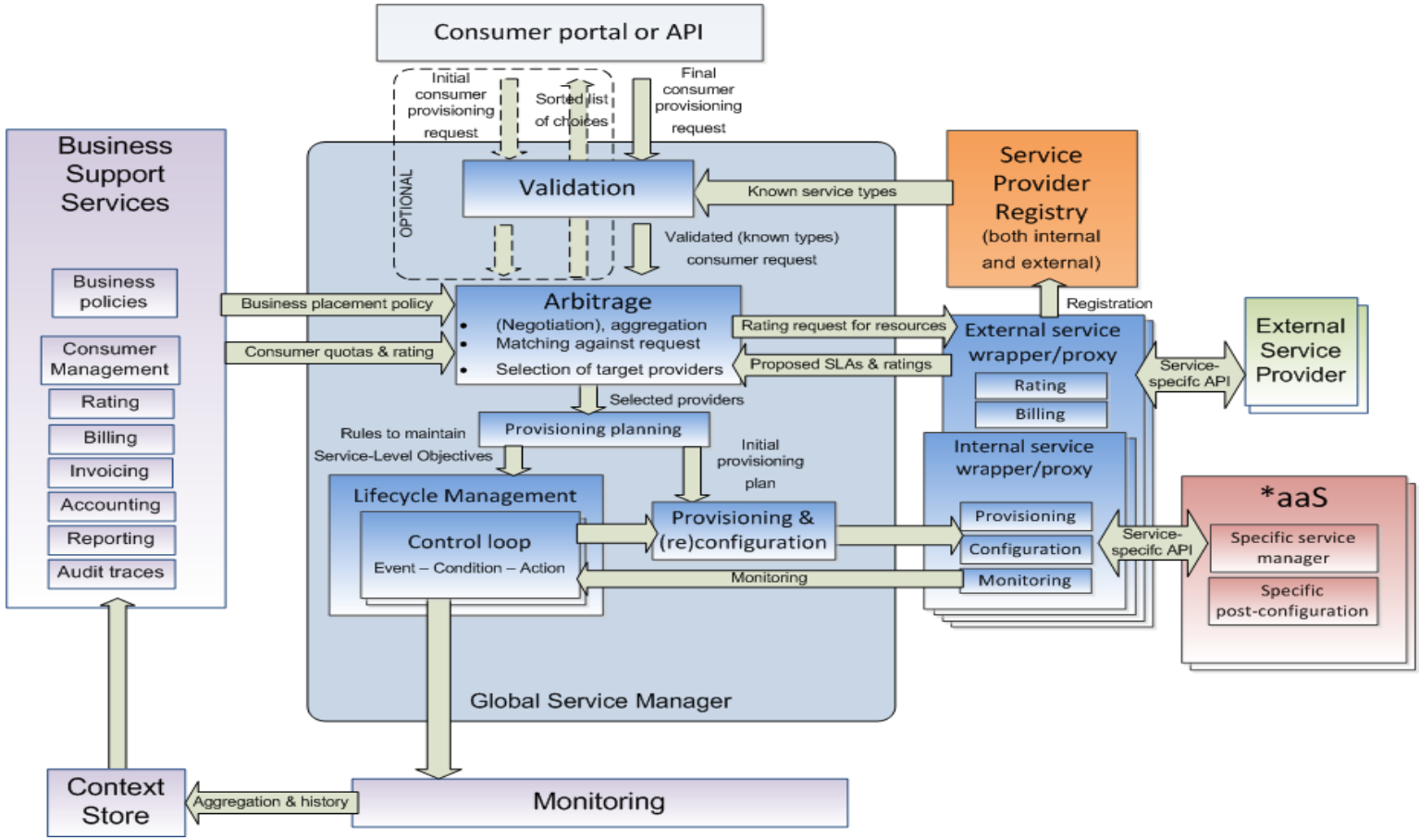


Number of virtual machines = 4 VM
Total memory used = 4 * 512 kb

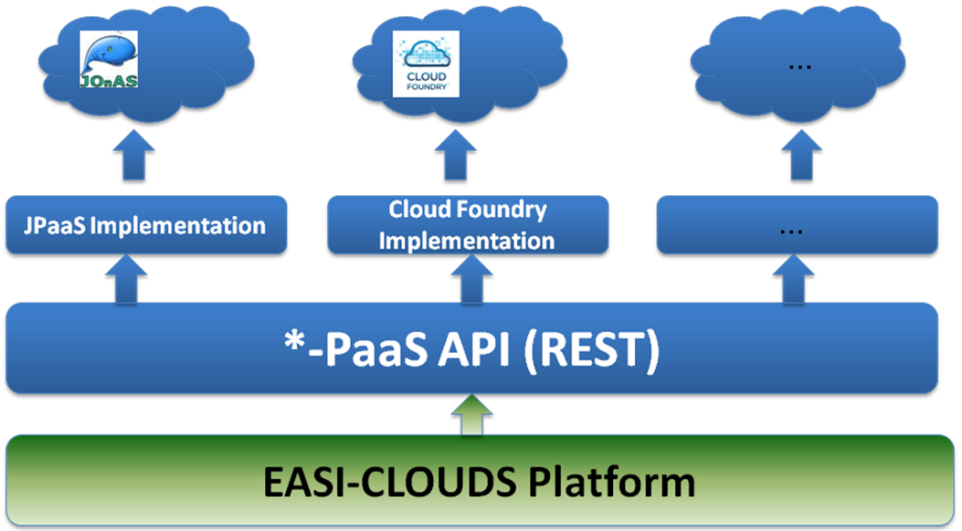
Cloud Platforms: Deployment

The need for a generic *-PaaS API

- **Interaction with several PaaS providers:**



The *-PaaS API



- User API
  - Application Management Operations (create, delete, start, etc.)
  - Environment Management Operations (create, delete, deploy, etc.)
  - Monitoring & Logging Management Operations (describe events, logs, etc.)
- Admin API
  - User Management (create, delete and describe user)
  - PaaS Management (deploy, start, stop, provision, etc.)
- **The specification is available at:**
  http://www-inf.it-sudparis.eu/~sellami/PaaS-API-Specification.pdf

Cloud Platforms: Deployment on hybrid clouds

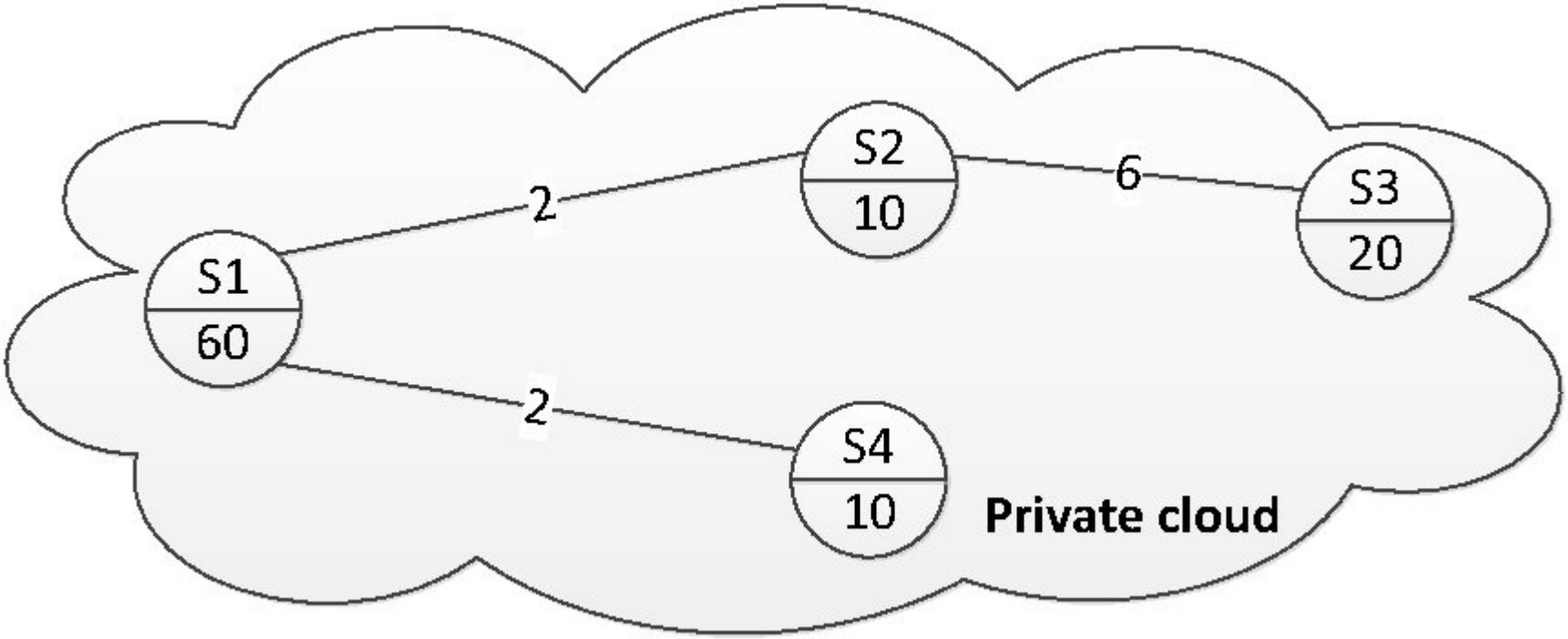
Deployment on hybrid clouds: when?

- **For matter of resources**
  - Already deployed applications request more resources the private cloud could not provide.
  - Already deployed applications release resources so that a re-deployment can be envisaged to release allocated resources in the public cloud.
  - New deployment requests to be fulfilled can not be satisfied by the private cloud.

- **For matter of good properties**
  - Good non functional properties provided by public cloud which can not be enforced/maintained in private cloud (e.g. persistency)

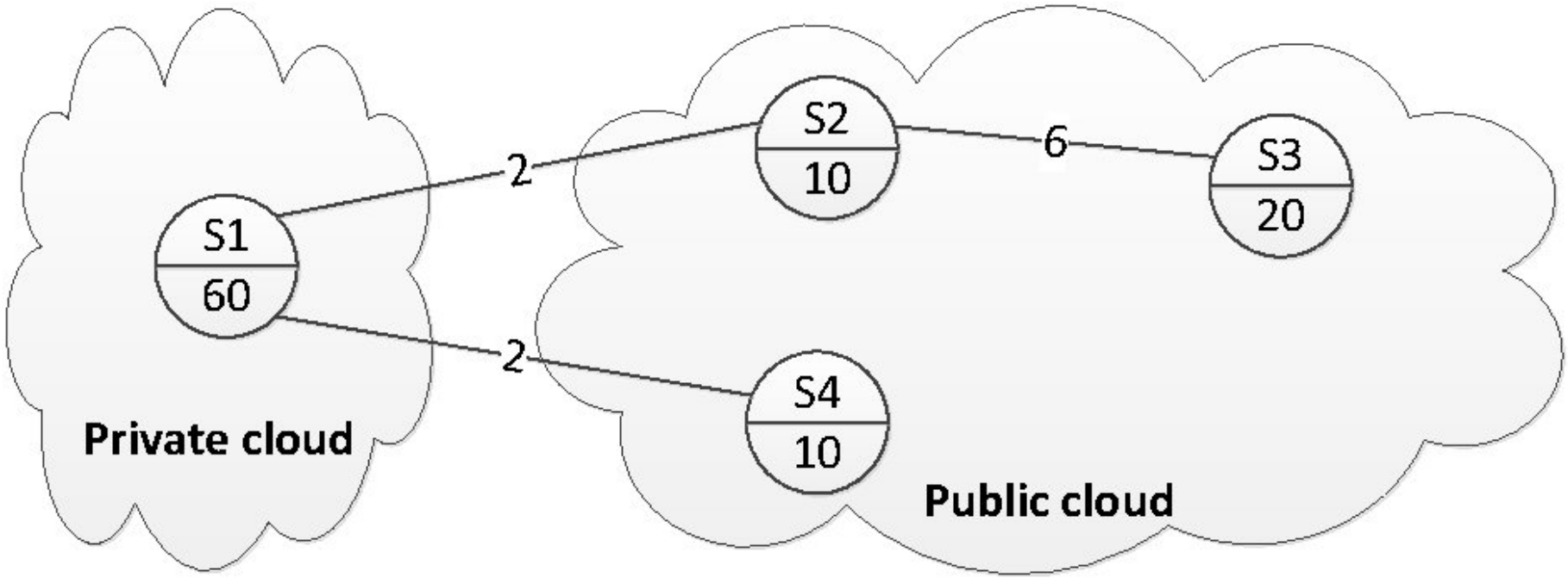
Deployment on hybrid clouds: Example (1/2)

SBA graph of the on-line store



Deployment on hybrid clouds: Example (2/2)

Deployment of service composition on hybrid cloud



Example: HQ= 25, α = 15, β1= 1 and β2 = 10

Deployment on hybrid clouds: objective ?

- **Parameters**
  - The need of resources from public cloud (units of platform resources required) noted **HQ**, unit cost noted α
  - communications between services inside the public cloud, unit cost noted β**1**
  - communication between private and public services, unit cost noted β**2**
  - Security, Privacy, etc
- **Objective**
  1. Minimizing costs of hosting and communications
  2. While the quantity of required platform resources of the chosen services has to be greeter or equal to HQ
  3. Non functional properties are enforced/maintained

Deployment on hybrid clouds: objective ?

Minimize: $H + PC + HC$

Subject to:

$\sum_{i=1}^{n} h(s_i) \times l(s_i) \geq HQ$

Where:

$H = \sum_{i=1}^{n} \alpha \times h(s_i) \times l(s_i)$

$PC = \sum_{e=\langle s_i, s_j \rangle \in E} \beta_2 \times c(e) \times l(s_i) \times l(s_j)$

$HC = \sum_{i=1}^{n} \sum_{j \ \ s.t. \ \ e=\langle s_i, s_j \rangle \in E} \beta_1 \times c(e) \times l(s_i) \times (1 - l(s_j))$

2-step algorithm

- **Step 1: while HQ is not fulfilled do**
  - Calculate, for each service S_i in the private cloud, the cost caused by its moving to the public cloud
    - Cost of inter-cloud communication, intra-public cloud communication and hosting in the public cloud
  - Choose the service with the minimum cost caused

- **Step 2:**
  - move back services from Public to Private cloud while
    - Reducing global cost and
    - enforcing that the hosting values of services in Public set are greater or equal to HQ

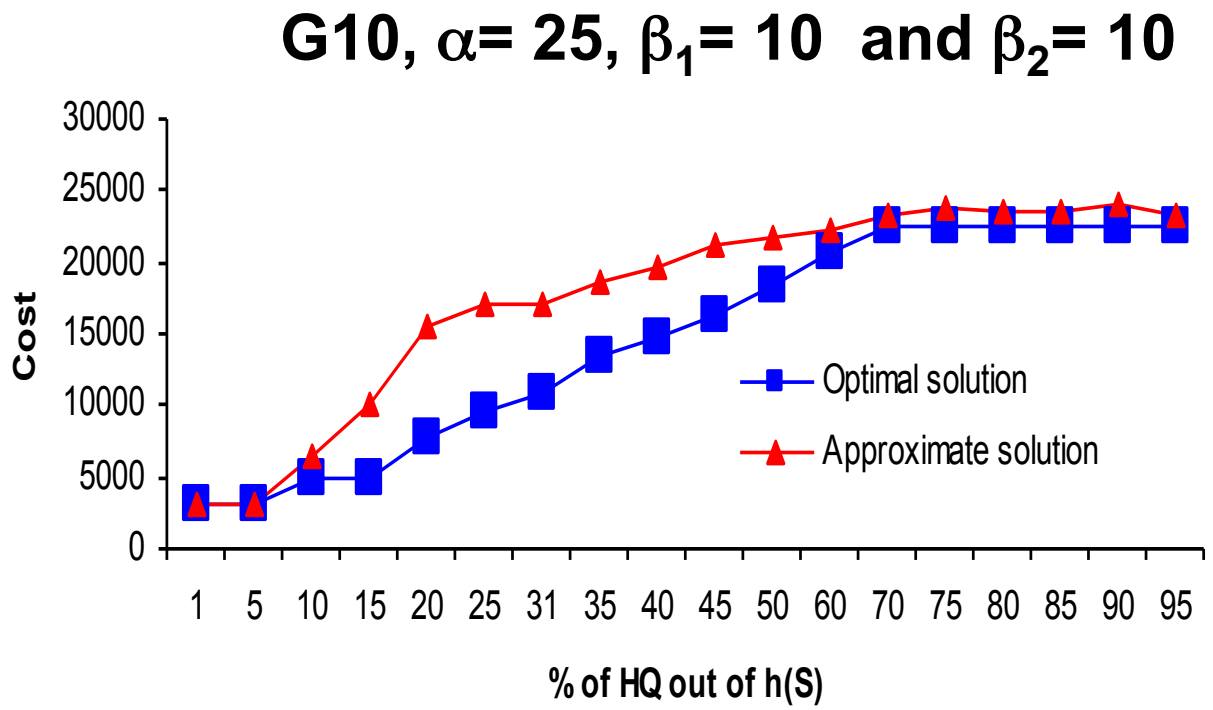
2-step algorithm: Experiments (1/6)

- More than 1000 experiments
- 10 graphs

Graphs	Nodes	Edges	Hosting needed	Density
G1	15	14	802	13,3%
G2	11	11	305	20%
G3	11	17	350	30,9%
G4	13	45	578	54,6%
G5	12	42	575	63,6%
G6	15	71	713	64,6%
G7	14	61	685	67%
G8	16	81	848	67,5%
G9	18	130	1017	84,9%
G10	16	120	785	100%

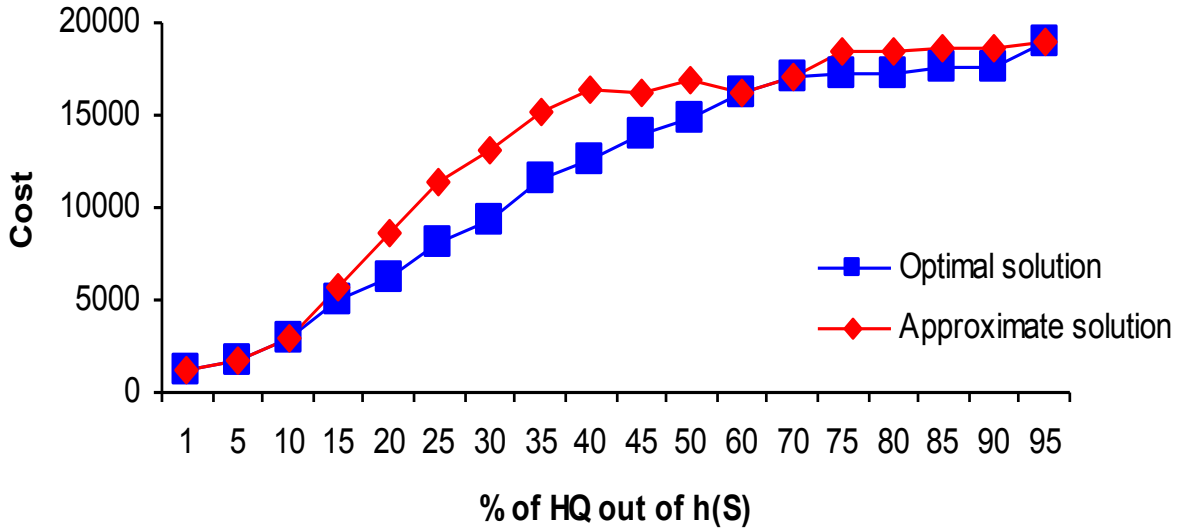
- α = 6, 8, 12, 15, 20, 25
- β_1= 1 and β_2 = 10
- HQ, 18 different values (varied from 1% to 95%)

2-step algorithm: Experiments (2/6)
Varying HQ on a complete graph



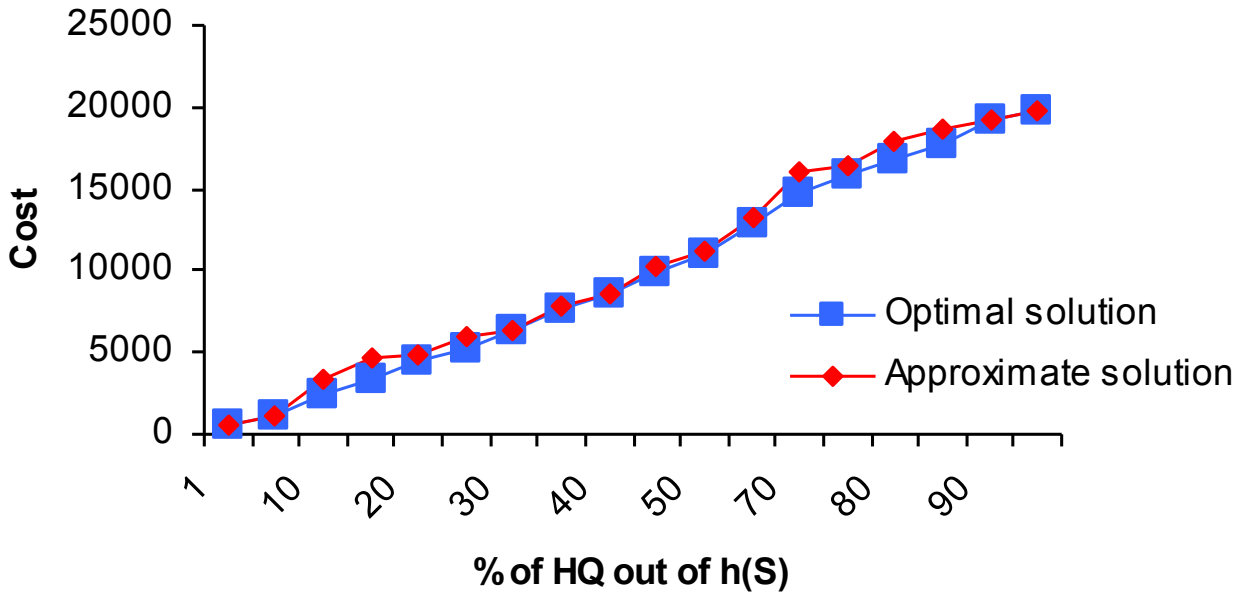
2-step algorithm: Experiments (3/6)
Varying HQ on a dense graph

G7, α= 25, β_1= 10 and β_2= 10

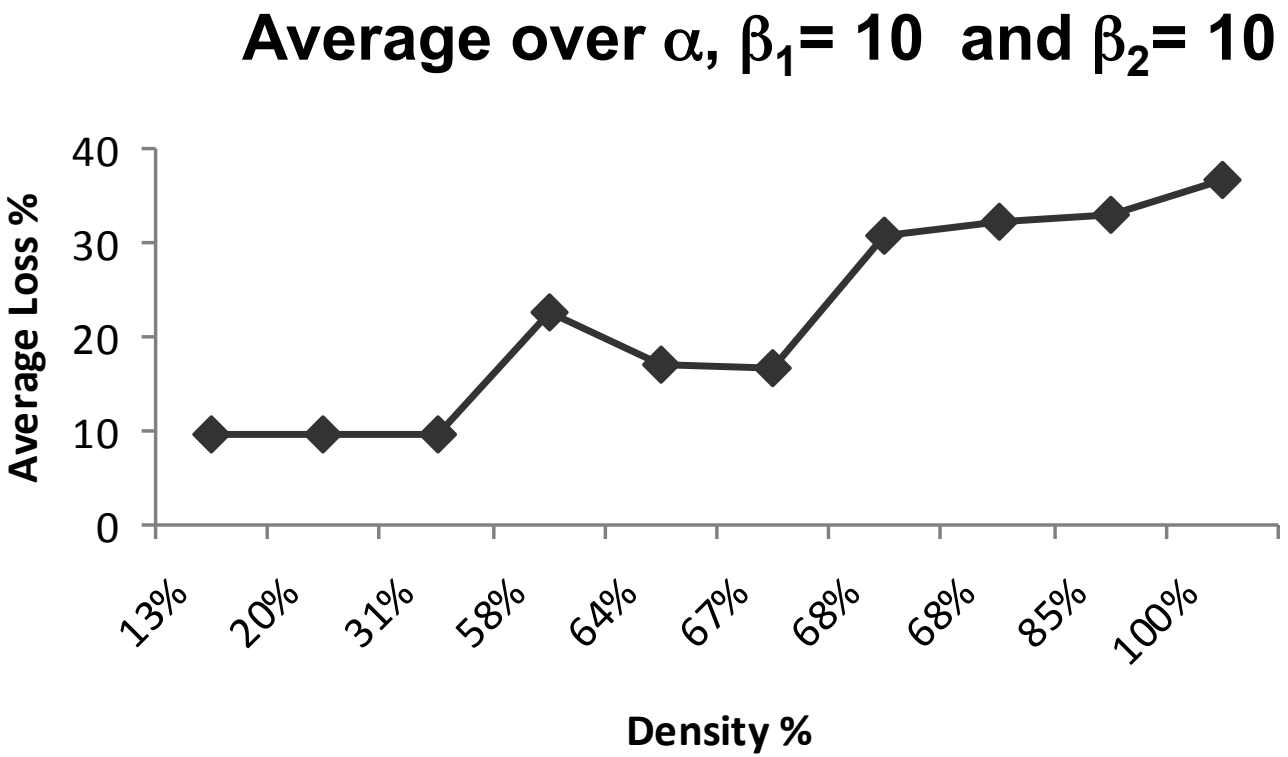


2-step algorithm: Experiments (4/6)
Varying HQ a sparse graph

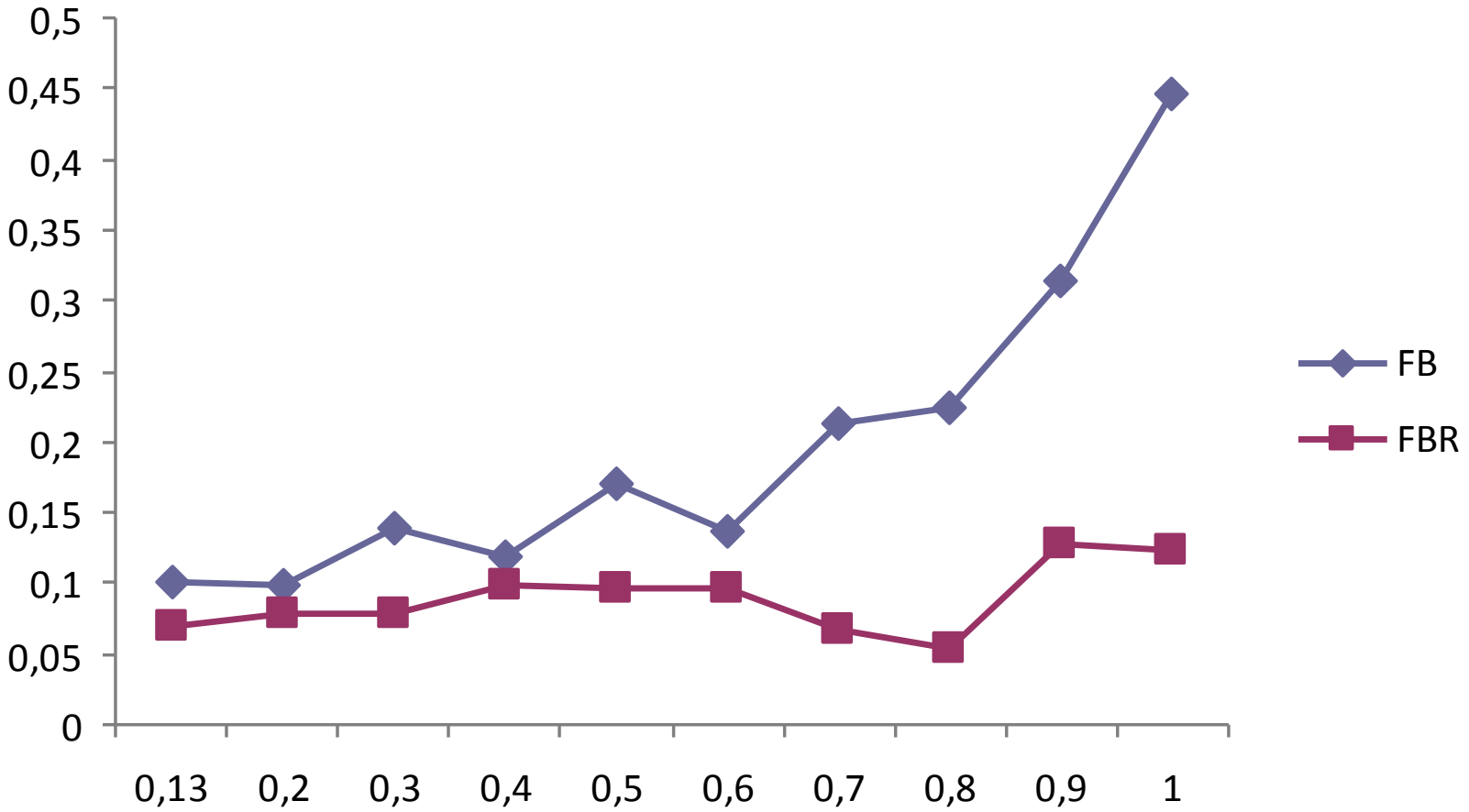
G1, α= 25, β_1= 10 and β_2= 10



2-step algorithm: Experiments (5/6)
Varying types of graphs



3-steps algorithm vs. 2-steps algorithm



Cloud Platforms:
Portability, Migration & Mobility

Portability

- **Portability**
  - the capability to operate software on different platforms without the need for changes

- **Problems***
  - Different products, different programming models
  - Different implementations of similar PaaS products
  - A great diversity in frameworks, toolsets and SDKs (proprietary)
  - Lack of common and standardized PaaS APIs to access to the PaaS Cloud
  - Heterogeneous data types and storing

* Cloud4SOA project

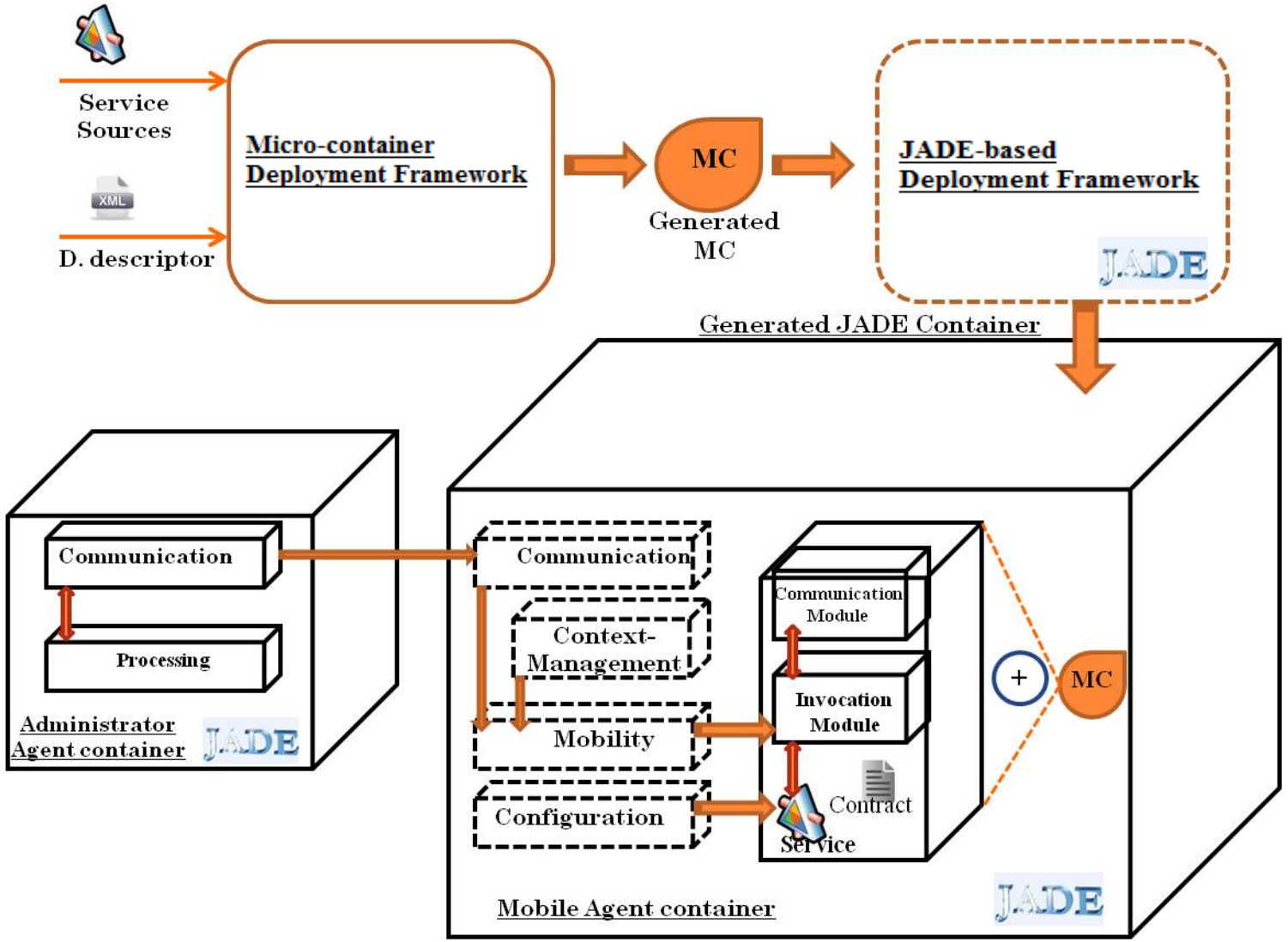
Mobility

- **Dynamicity of Cloud environment → Mobility techniques**
- **Mobility & Migration at IaaS level (VM migration)**
  - Moving VM
  - Consequences : moving all its platform
- **Mobility & Migration at PaaS level (platform migration)**
  - Moving platform
  - Consequences : moving all its applications
- **Mobility & Migration at SaaS level (service migration)**
  - Consider constraints on the destination
  - Need of mobility to deal with heterogeneity at a fine granularity

2 approaches for service mobility

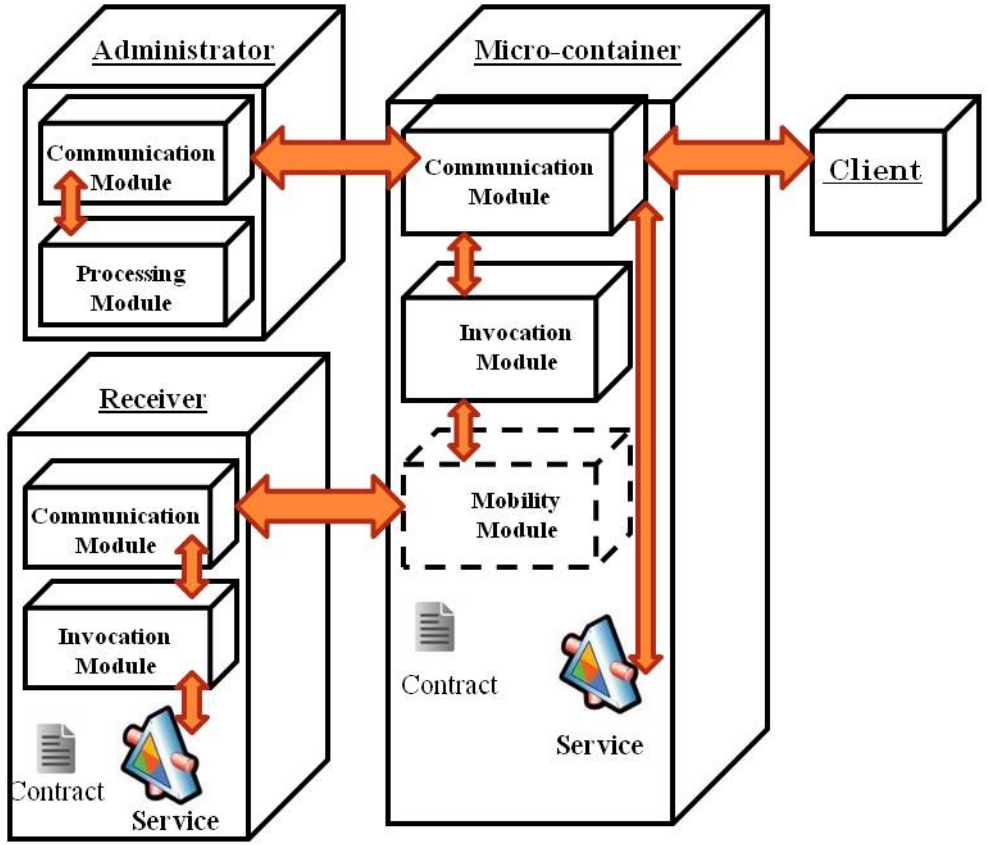
- **JADE-based mobility of services in the Cloud**
  - Encapsulate micro-container into mobile agent
- **Mobile Micro-container**
  - Add mobility component for micro-container

Approach 1: JADE-based mobility



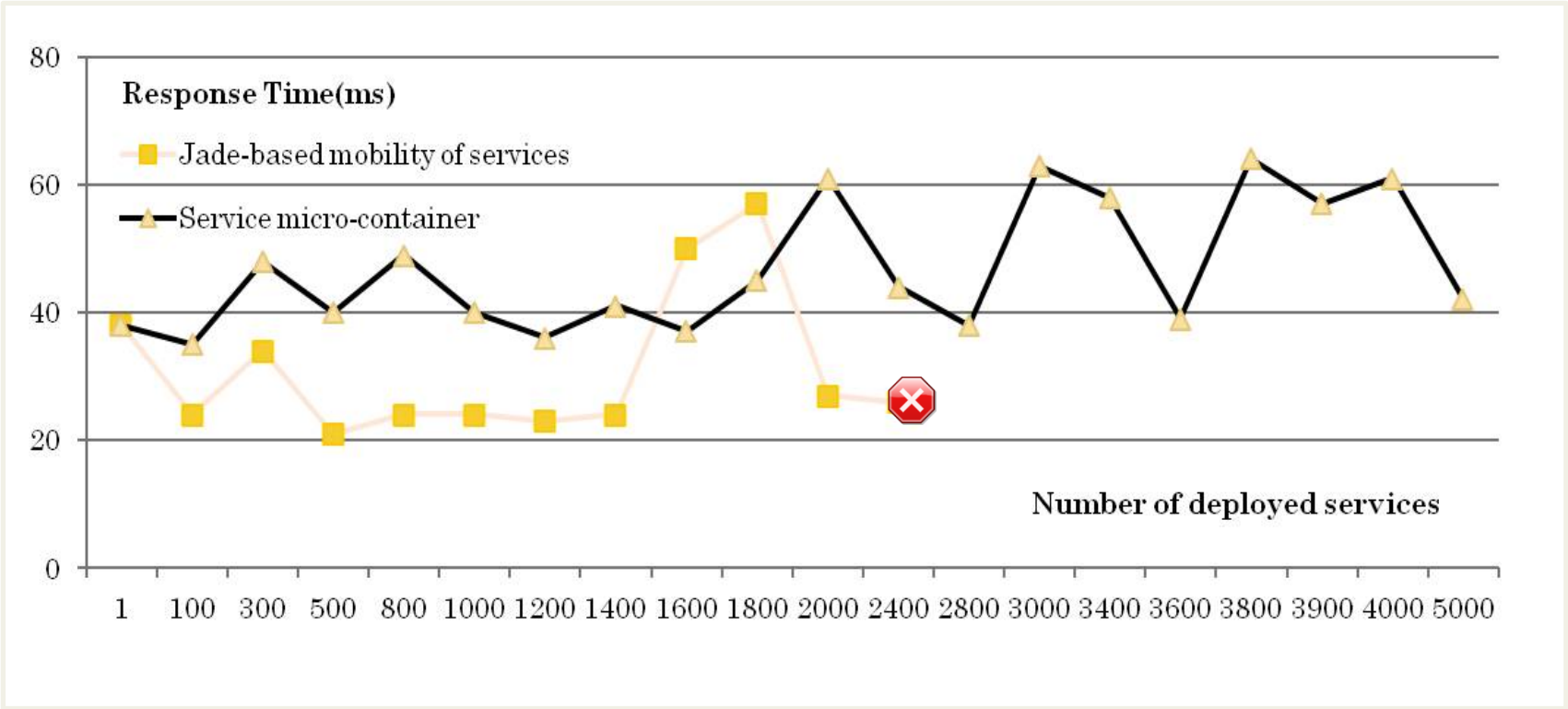
Approach 2: Mobile Micro-container

- **An administrator**
- **A Micro-container**
- **A Receiver:**
  - Communication module
  - Invocation module
  - Service module
- **A thin client**



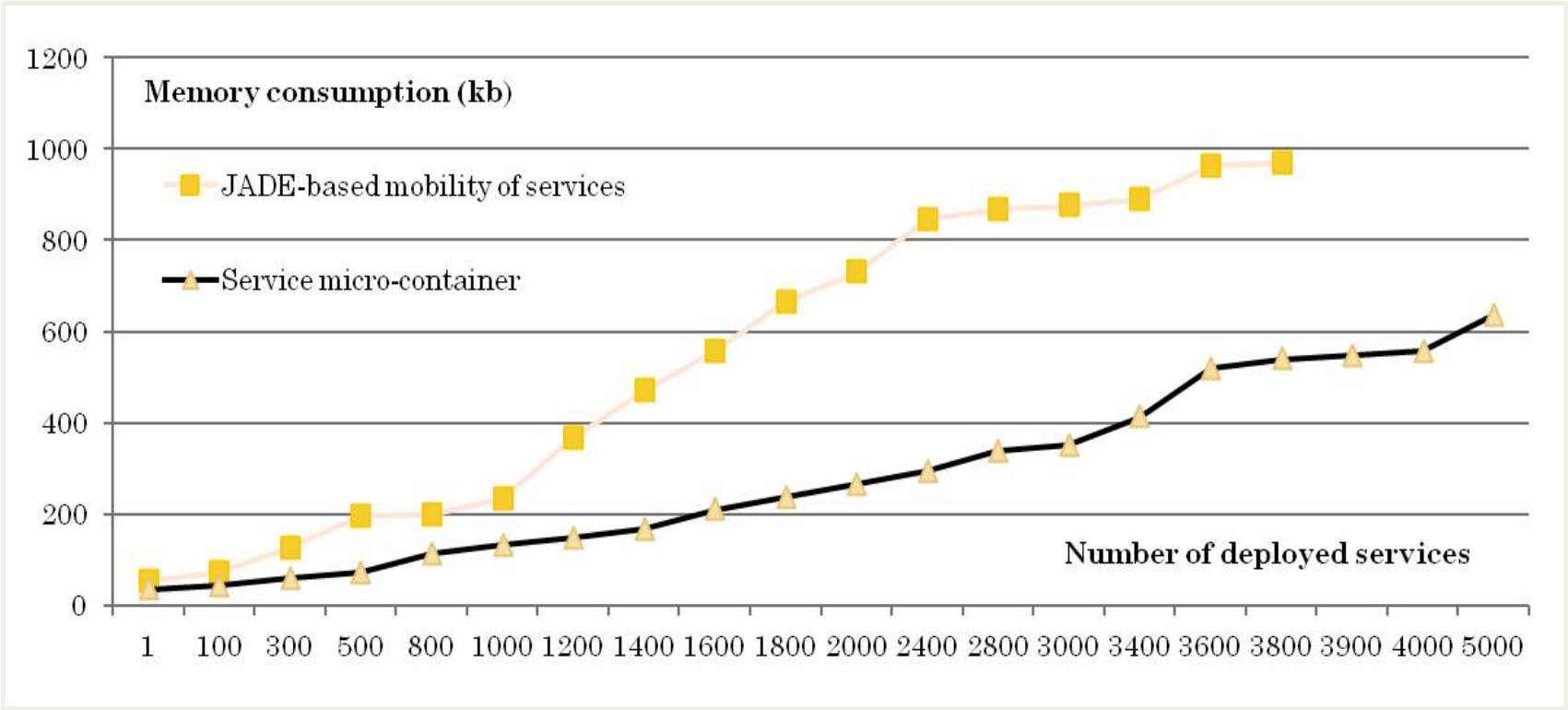
Approach 1

Mobile Agent VS Micro-Container (Response Time)



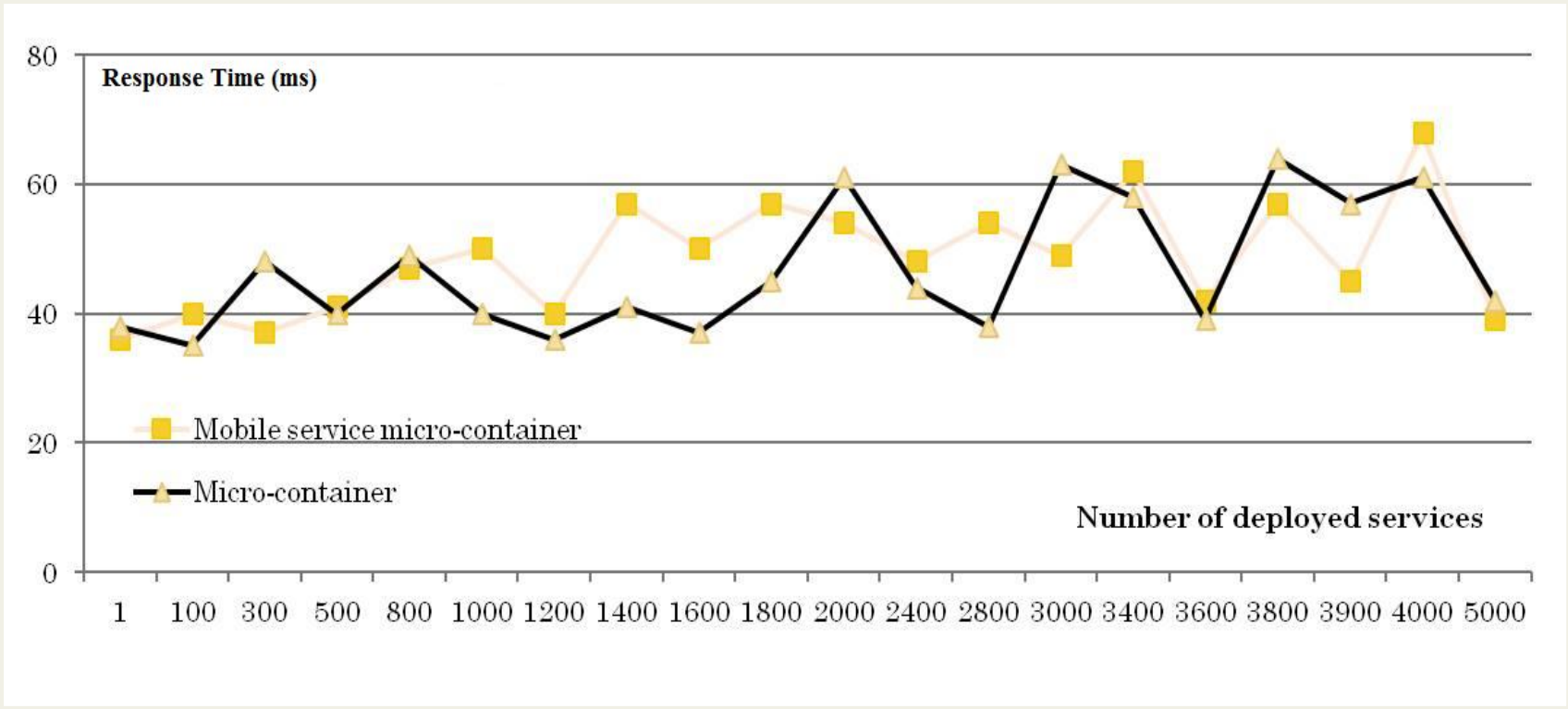
Approach 1

Mobile Agent VS Micro-Container (Memory consumption)



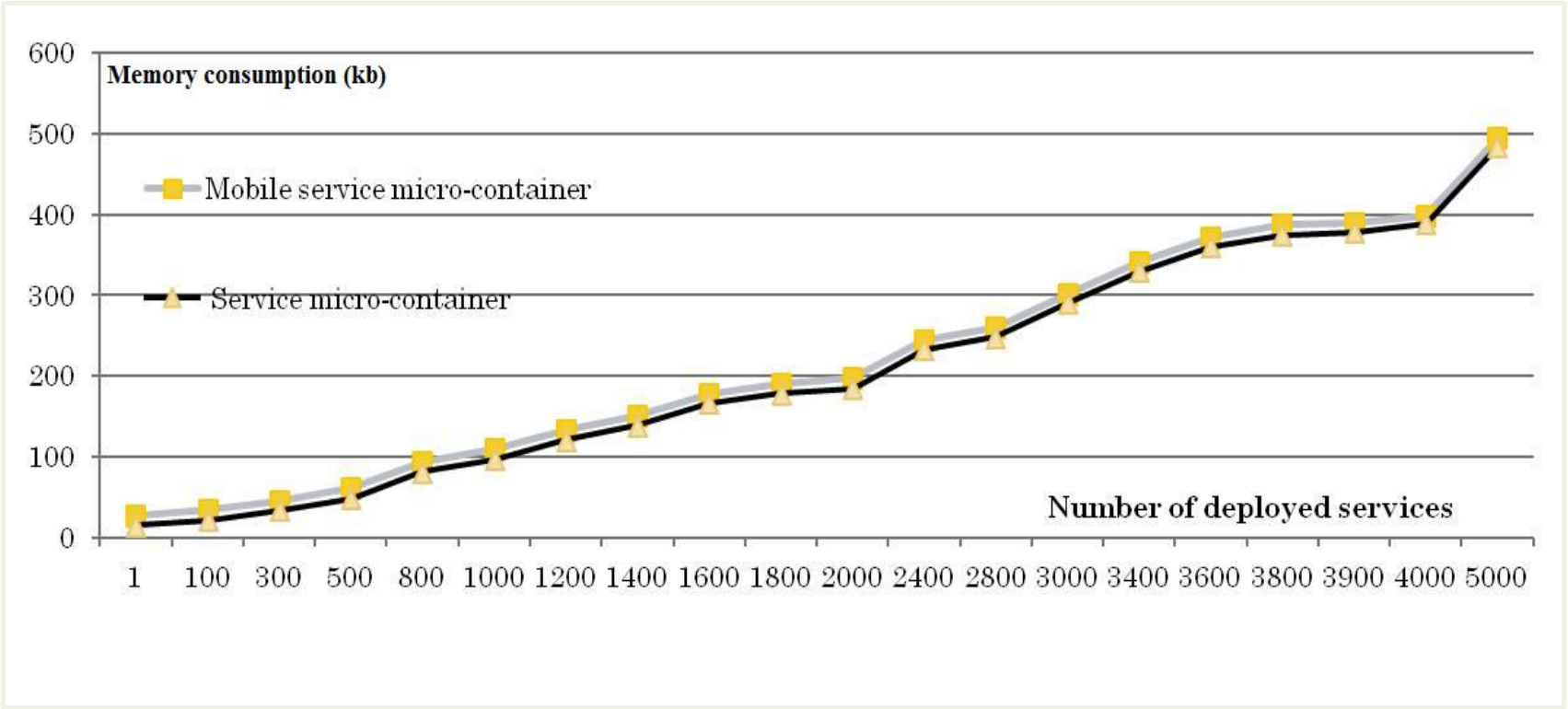
Approach 2

Mobile Micro-Container VS Micro-Container (Response Time)



Approach 2

Mobile Micro-Container VS Micro-Container (Memory consumption)



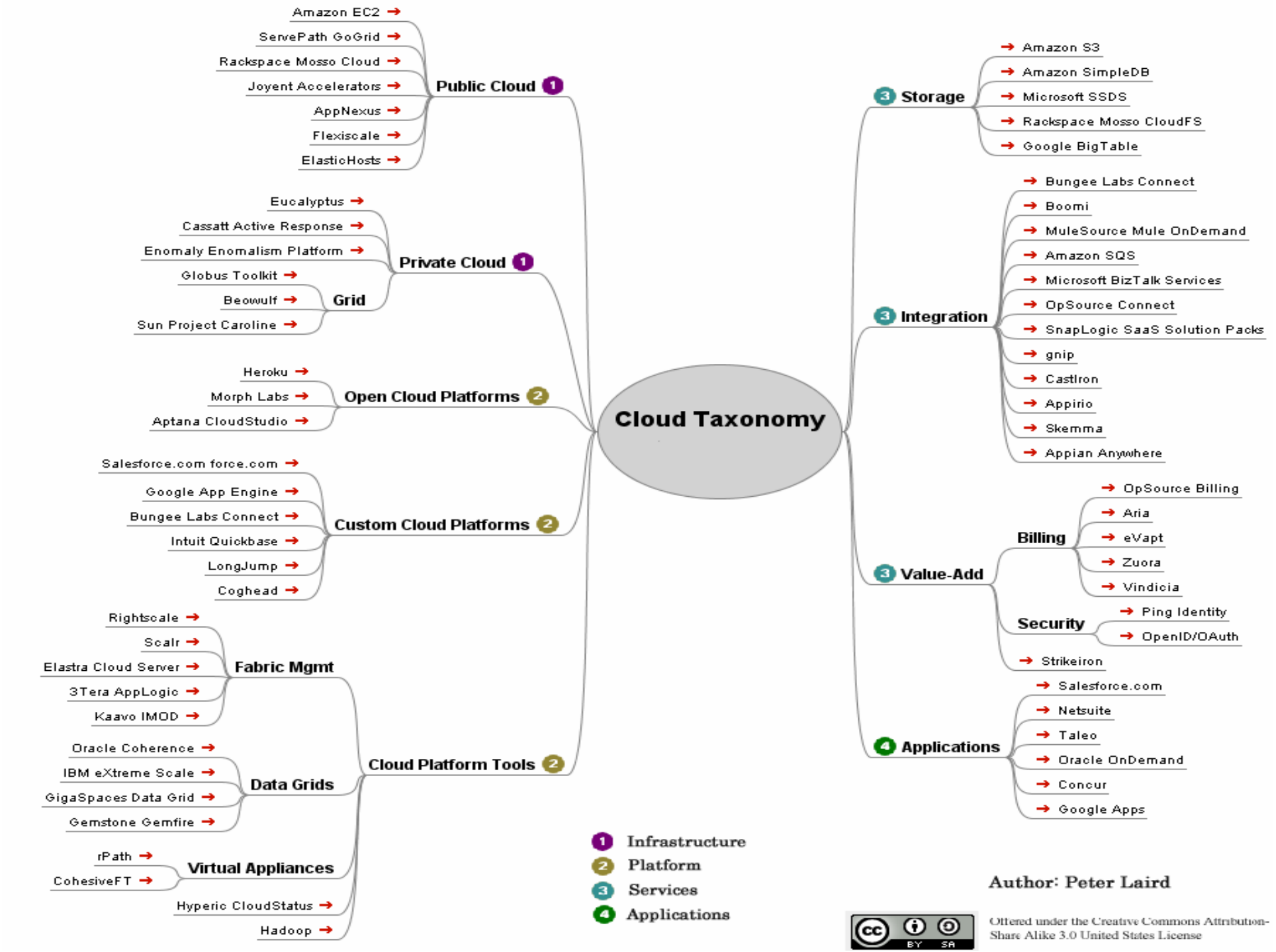
Conclusion and future work

- **Mobility for micro-containers**
  - Mobility techniques on Cloud environment don't consider the heterogeneity of services composing applications
  - Design, Implementation and experiments of a mobile micro-container
- **Next ?**
  - Extend mobility module to allow stateful service migration
  - Develop and integrate a decision module

Cloud Platforms:
Conclusion

Conclusion

- **PaaS for industrial**
  - Big companies
    - Resources sharing
  - SMEs
    - Explore market opportunities without investment risk
    - Focus on core competencies (special for non computer-based businesses)
- **PaaS for researcher**
  - Rethink our findings
    - Key concept: elasticity
  - E.g. challenges related to software and application architecture, PaaS


Cloud Taxonomy
Public Cloud 1
Amazon EC2
ServePath GoGrid
Rackspace Mosso Cloud
Joyent Accelerators
AppNexus
Flexiscale
ElasticHosts
Private Cloud 1
Eucalyptus
Cassatt Active Response
Enomaly Enomalism Platform
Grid
Globus Toolkit
Beowulf
Sun Project Caroline
Open Cloud Platforms 2
Heroku
Morph Labs
Aptana CloudStudio
Custom Cloud Platforms 2
Salesforce.com force.com
Google App Engine
Bungee Labs Connect
Intuit Quickbase
LongJump
Coghead
Cloud Platform Tools 2
Fabric Mgmt
Rightscale
Scalr
Elastra Cloud Server
3Tera AppLogic
Kaavo IMOD
Data Grids
Oracle Coherence
IBM eXtreme Scale
GigaSpaces Data Grid
Gemstone Gemfire
Virtual Appliances
rPath
CohesiveFT
Hyperic CloudStatus
Hadoop
3 Storage
Amazon S3
Amazon SimpleDB
Microsoft SSDS
Rackspace Mosso CloudFS
Google BigTable
3 Integration
Bungee Labs Connect
Boomi
MuleSource Mule OnDemand
Amazon SQS
Microsoft BizTalk Services
OpSource Connect
SnapLogic SaaS Solution Packs
gnip
CastIron
Appirio
Skemma
Appian Anywhere
3 Value-Add
Billing
OpSource Billing
Aria
eVapt
Zuora
Vindicia
Security
Ping Identity
OpenID/OAuth
Strikeiron
4 Applications
Salesforce.com
Netsuite
Taleo
Oracle OnDemand
Concur
Google Apps
1 Infrastructure
2 Platform
3 Services
4 Applications
Author: Peter Laird
Offered under the Creative Commons Attribution-Share Alike 3.0 United States License

PaaS examples

PaaS vendor	What it offers
Amazon Web Services (AWS)	Amazon Elastic Beanstalk of AWS allows developers to host, deploy, and manage applications in the cloud.
Google	Google's PaaS platform, App Engine, allows organizations to build and host their web applications. The vendor promises centralized administration, 99.9% uptime, and security. It charges $8 per user per month (with maximum of $1000 a month) for its PaaS platform.
OrangeScape	OrangeScape claims to offer, what it calls, a Cross-Cloud PaaS platform by integrating the resources of Google App Engine, Microsoft Azure, IBM Smart Cloud, and Amazon EC2.

PaaS examples

PaaS vendor	What it offers
Microsoft	Microsoft offers PaaS platform services using Windows Azure AppFabric. These cloud middleware services include Service Bus, Access Control, Caching, Integration, and Composite App. The vendor assures compatibility with all programming languages and development frameworks including .NET, Java, Ruby, and PHP.
Salesforce.com	PaaS platform offerings of Salesforce.com include Social Application Platforms, Raw Compute Platforms, Web Application Platforms, and Business Application Platforms.
Sify Technologies	Sify's PaaS services are offered using its On-Demand Hosting Platform powered by HP's converged infrastructure components.