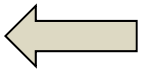
Master Thesis:

Optimization of QoS for Cloud-Based Services through Elasticity and Network Awareness

Alexander Fedulov

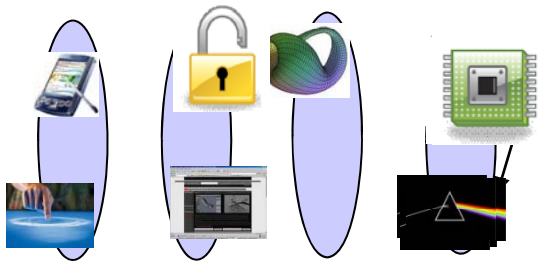
Agenda

- **BonFIRE Project overview** 
- Motivation
- General System Architecture
  - Monitoring in the Cloud environment
  - Software Load Balancers
  - Elasticity Engine
- Elasticity: Upscaling/Downscaling
- Network implications
- Summary

BonFIRE Project Overview

Testing experiments selected in open calls

Service Components in virtual machines to be deployed on the Cloud

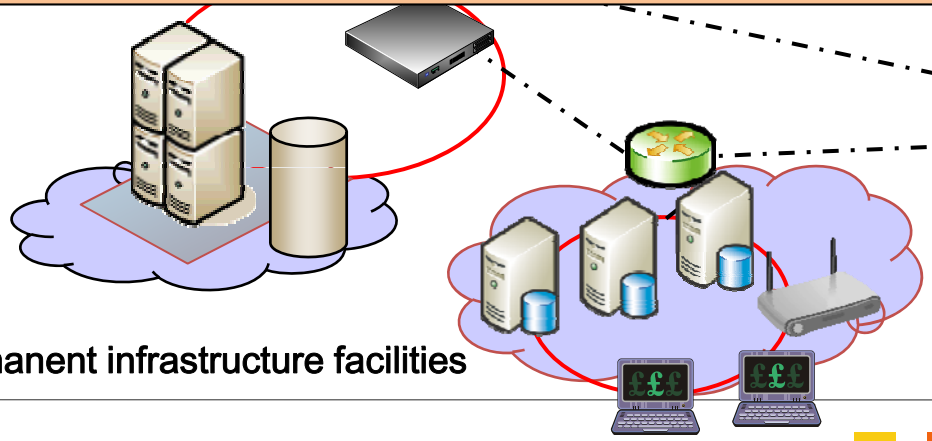


Service combination in validation

La

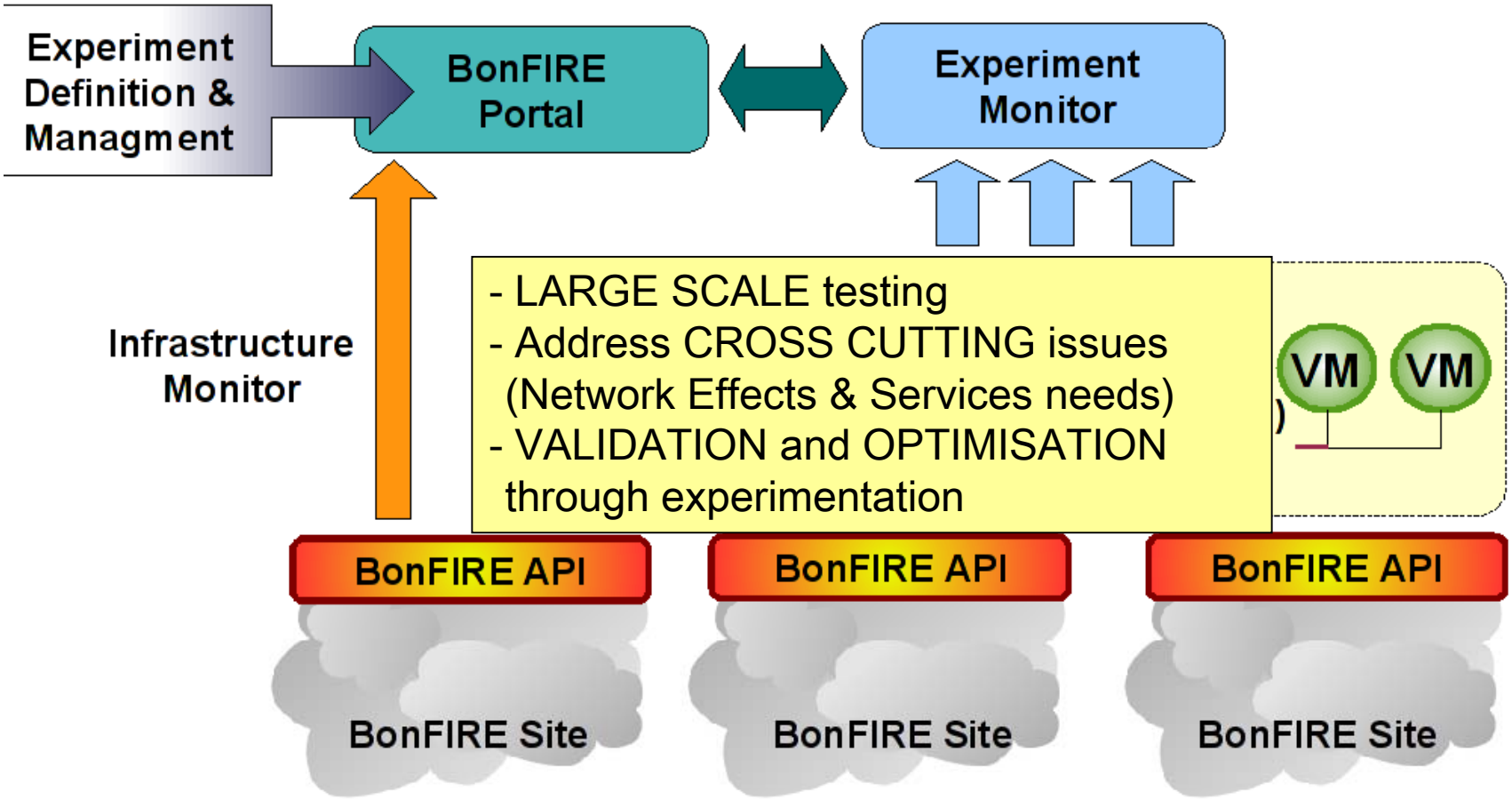
Netw

The **BonFIRE** (Building service testbeds for Future Internet Research and Experimentation) project will design, build and operate a multi-site cloud-based facility to support research across applications, services and systems **targeting services research community** on Future Internet.

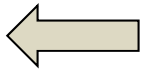


Permanent infrastructure facilities

Operational Infrastructure: Hardware, and Software

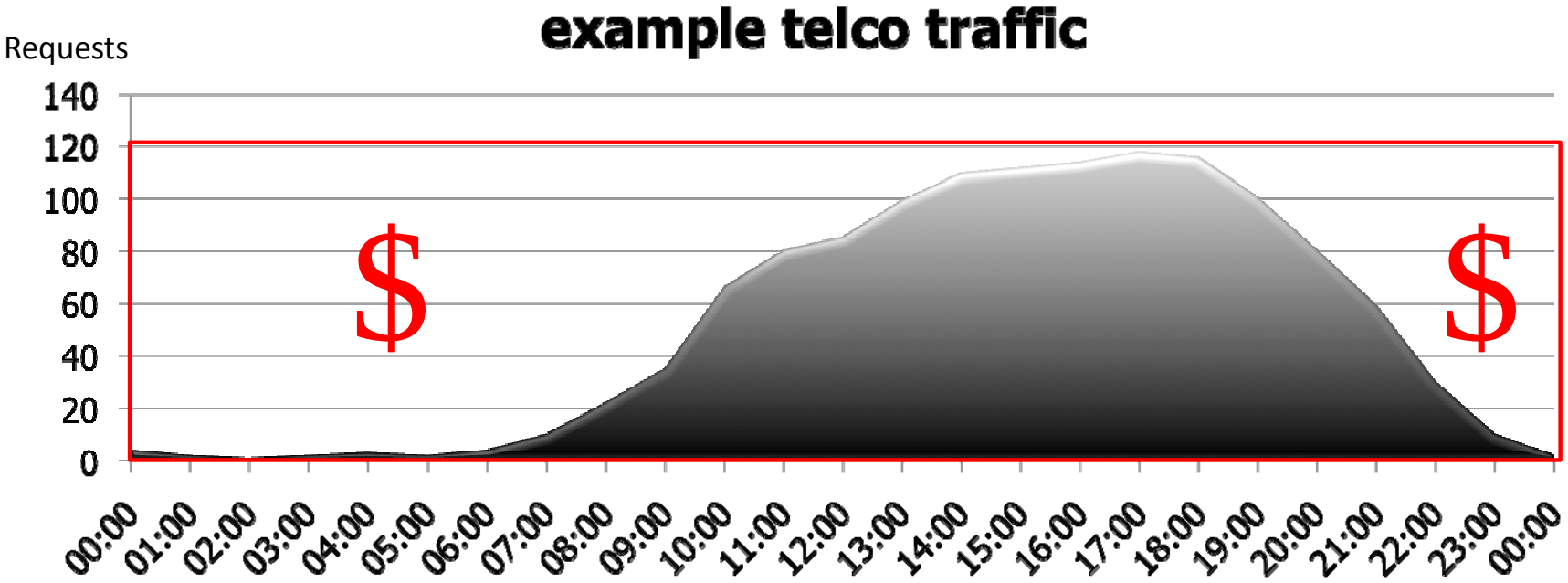


Agenda

- BonFIRE Project overview
- **Motivation** 
- General System Architecture
  - Monitoring in the Cloud environment
  - Software Load Balancers
  - Elasticity Engine
- Elasticity: Upscaling/Downscaling
- Network implications
- Summary

Motivation

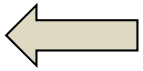
- Cloud & SOA-based Environments should provide elasticity
  - dynamical change of the environment according to the actual demand



Motivation

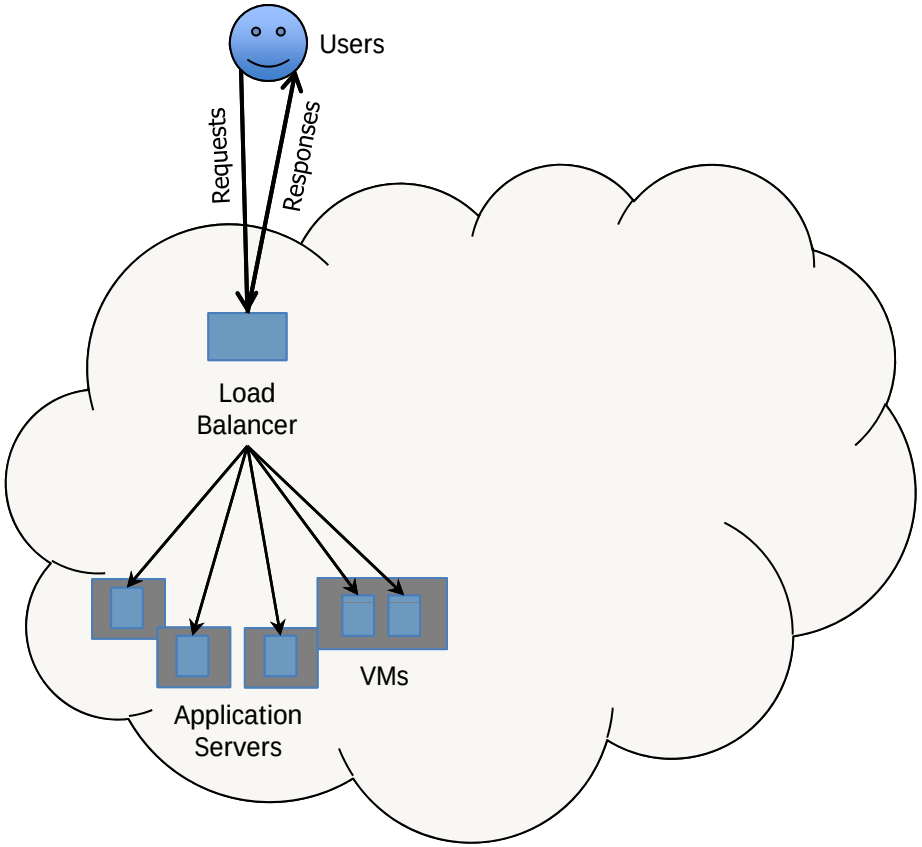
- Elasticity:
  - essential property of a Cloud Environment, allowing to dynamically scale utilized resources in accordance to the momentary demand
- Automated Elasticity:
  - elasticity, executed and controlled autonomously based on Key Performance Indicators (KPIs)
- KPIs that can be taken into account are:
  - CPU Utilization
  - Memory Utilization
  - Response Time
  - Network parameters
    - Delay
    - Packet loss
    - etc.

Agenda

- BonFIRE Project overview
- Motivation
- **General System Architecture** 
  - Monitoring in the Cloud environment
  - Software Load Balancers
  - Elasticity Engine
- Elasticity: Upscaling/Downscaling
- Network implications
- Summary

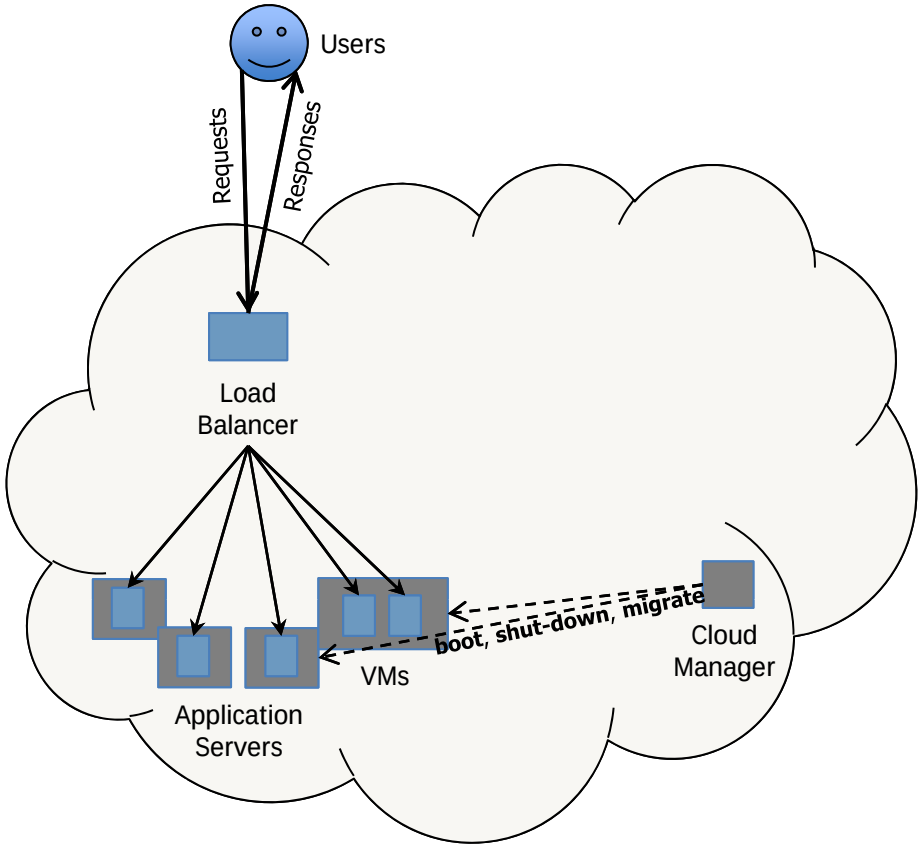
Cloud-based Service Environment – (1/3) – Load Balancing

- In a typical, cloud-based service environment, the load is shared between application servers dynamically
- A Load Balancer represents a key component, allowing for dynamic load sharing between multiple serving nodes (application servers)
- Different load-sharing algorithms exist, e.g. round robin, weighted round robin, random, sticky session etc.
- Load-balancers can be application specific (L7 HTTP load sharing) , or application agnostic (L4 load sharing)



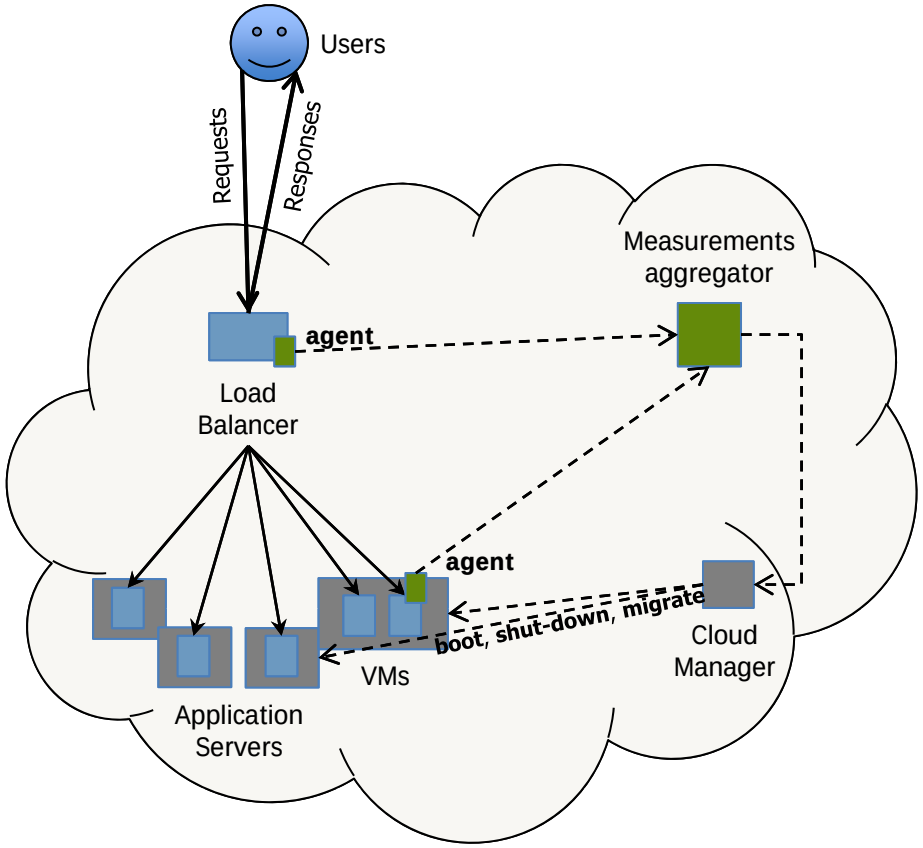
Cloud-based Service Environment – (2/3) – Cloud Management

- In a typical cloud-based service environment, application servers run on top of VMs
- Virtual machines can be dimensioned in a flexible way regarding allocated CPU, memory and storage
- A Cloud Management System is capable of dynamically booting, migrating and deleting virtual machines running on heterogeneous hypervisors (KVM, XEN, VMware etc.)

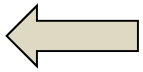


Cloud-based Service Environment – (3/3) – Service Monitoring

- In order to provide dynamic cloud elasticity, different measurements (e.g. VM, service and even network) have to be taken into account
- By aggregating and analyzing real-time measurements, an elasticity management system is capable of intelligently controlling cloud management systems
- Based on performance thresholds (e.g. service execution time, network performance, host/VM performance), triggering the setup, migration and destruction of distributed VMs can be performed



Agenda

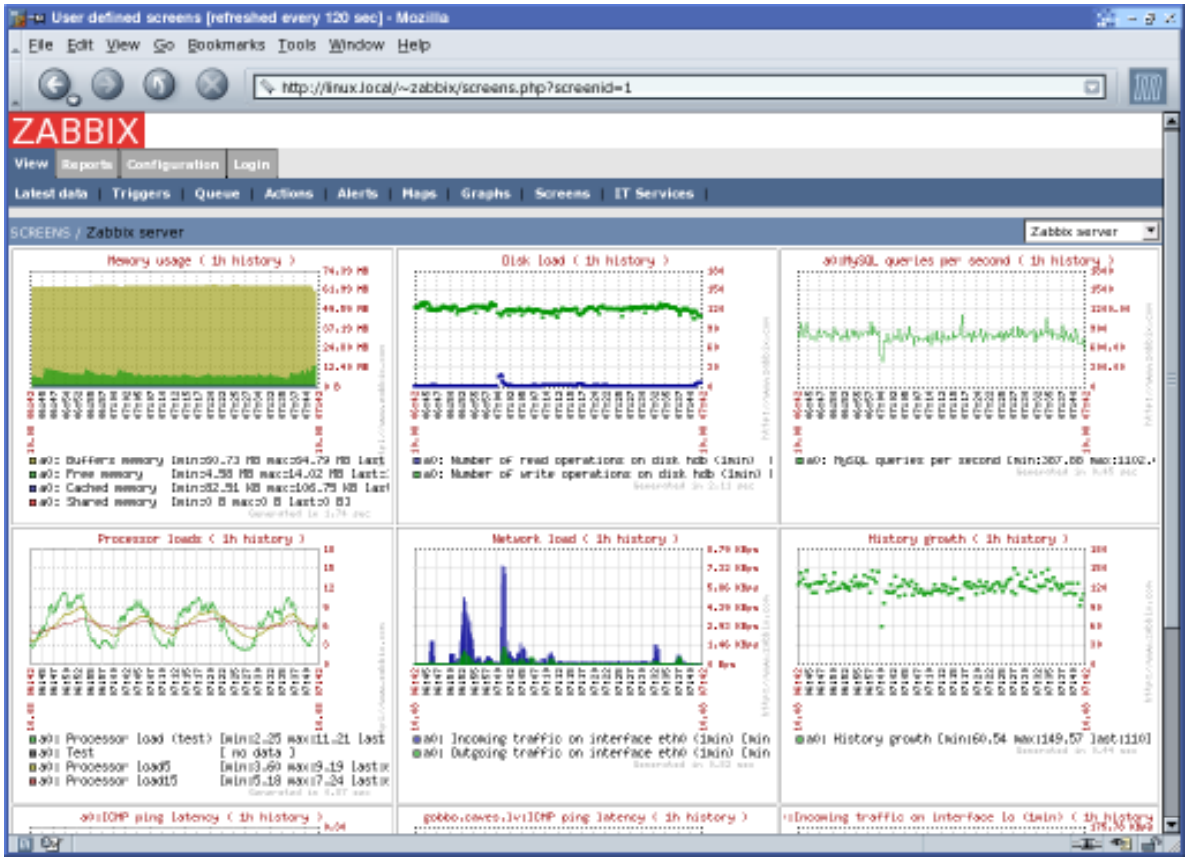
- BonFIRE Project overview
- Motivation
- General System Architecture
  - **Monitoring in the Cloud environment** 
  - Software Load Balancers
  - Elasticity Engine
- Elasticity: Upscaling/Downscaling
- Network implications
- Summary

Monitoring Systems Comparison chart

Name	IP SLA Reports	Logical Grouping	Trending	Trend Prediction	Auto Discovery	Agent	SNMP	Syslog	Plugins	Triggers / Alerts	WebApp	Distributed Monitoring	Inventory	Data Storage Method	License	Maps	Access Control	IPv6
Ganglia	No	Yes	Yes	No	Via gmond check in	Yes	Via plugin	No	Yes	No	Viewing	Yes	Unknown	RRDtool, in memory	BSD	Yes	No	Unknown
OpenNMS	Yes	Yes	Yes	Unknown	Yes	Supported	Yes	Yes	Yes	Yes	Full Control	Yes	Limited	RRD, PostgreSQL	GPL	Yes	Yes	Limited
Zabbix	Yes	Yes	Yes	Yes	Yes	Supported	Yes	Yes	Yes	Yes	Full Control	Yes	Yes	Oracle MySQL PostgreSQL SQLite	GPL	Yes	Yes	Yes
Nagios	Via plugin	Yes	Yes	No	Via plugin	Supported	Via plugin	Via plugin	Yes	Yes	Full Control	Yes	Via plugin	Flat file, SQL	GPL	Yes	Yes	Yes
Name	IP SLA Reports	Logical Grouping	Trending	Trend Prediction	Auto Discovery	Agent	SNMP	Syslog	Plugins	Triggers / Alerts	WebApp	Distributed Monitoring	Inventory	Data Storage Method	License	Maps	Access Control	IPv6

Source: http://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems

Zabbix Monitoring System

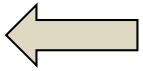


Source: http://en.wikipedia.org/wiki/Zabbix

Zabbix Monitoring System

- Client-server based monitoring
- Active and passive modes (active mode used for trespassing domain Firewalls)
- Flexibility in adding user-defined Metrics by executing shell scripts
- Auto-registration/linkage of newly added hosts (Virtual Machines)
- (!) Well-defined API for performing Create, Update, Delete (CRUD) operations on almost all inner data structures (Items, Hosts, Host Groups, Templates, Triggers, Actions etc.)
- Storing metrics data in databases (not files, like Nagios) → simplified data processing

Agenda

- BonFIRE Project overview
- Motivation
- General System Architecture
  - Monitoring in the Cloud environment
  - **Software Load Balancers** 
  - Elasticity Engine
- Elasticity: Upscaling/Downscaling
- Network implications
- Summary

Load Balancers: comparison chart

	HAProxy	Squid+MRTG	Ultra Monkey	NGINX	Apache mod_proxy balancer
Supported Layer	4/7	7	4/7	7	7
Actuality	2010	2010	2008	2010	
License	GPL v2	GPL	GPL	2-clause BSD-like	Apache License v2
Documentation	++	++	+	++	+
Inbuild Monitoring	-	MRTG	-	-	-

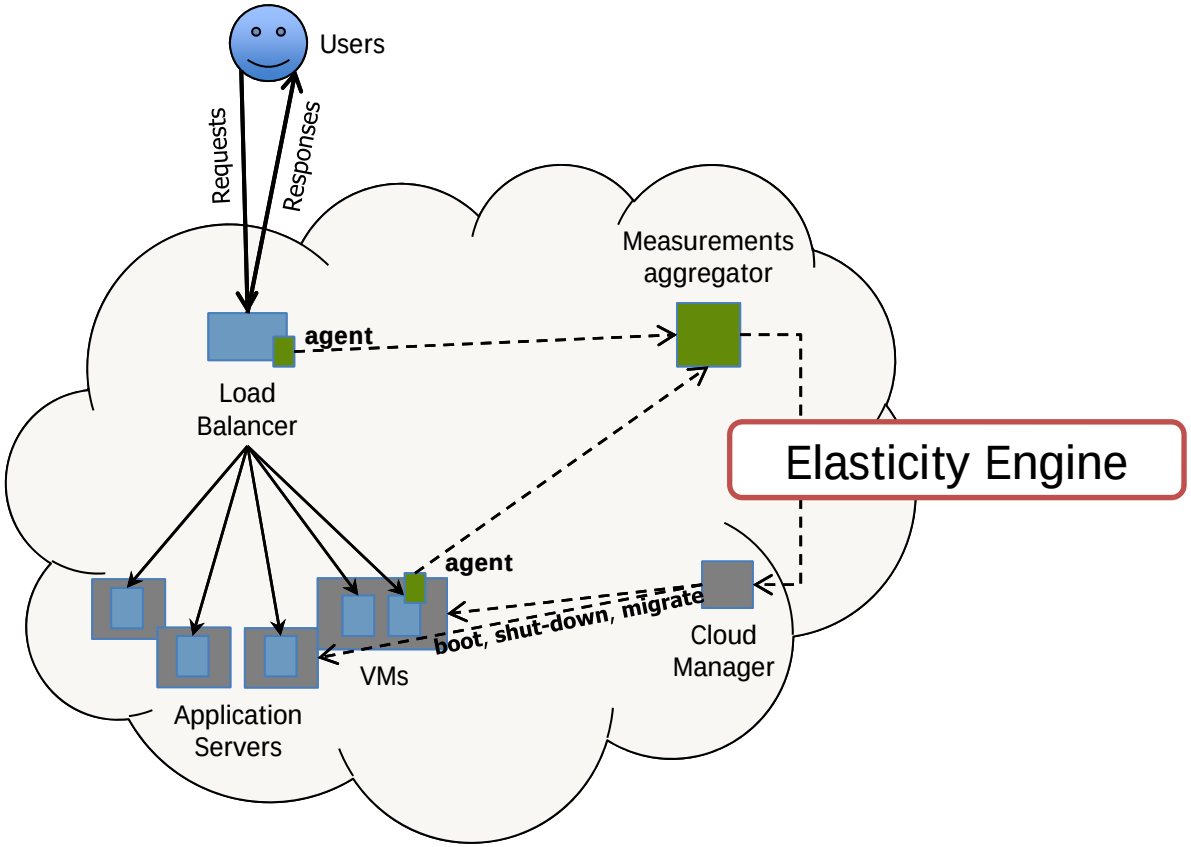
Load Balancers

- Requirements:
  - dynamically add/remove virtual machines
  - dynamically assign different “weights” to the running servers
- Non of the investigated load balancers had API for remote reconfiguration
- Simple REST interface was developed for this purpose for NGINX and HAProxy load balancers

Agenda

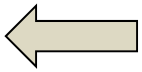
- BonFIRE Project overview
- Motivation
- General System Architecture
  - Monitoring in the Cloud environment
  - Software Load Balancers
  - **Elasticity Engine**
- Elasticity: Upscaling/Downscaling
- Network implications
- Summary

Elasticity Engine



Agenda

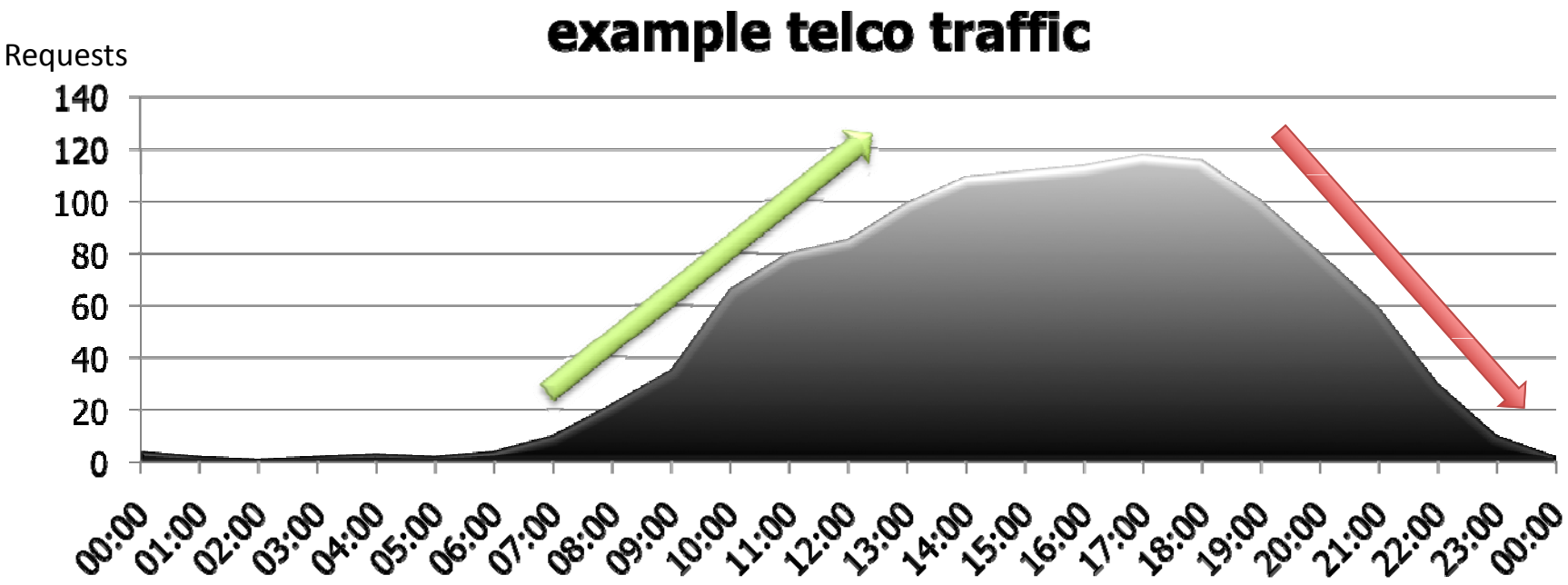
- BonFIRE Project overview
- Motivation
- General System Architecture
  - Monitoring in the Cloud environment
  - Software Load Balancers
  - Elasticity Engine
- **Elasticity: Upscaling / Downscaling**
- Network implications
- Summary



Assumptions:

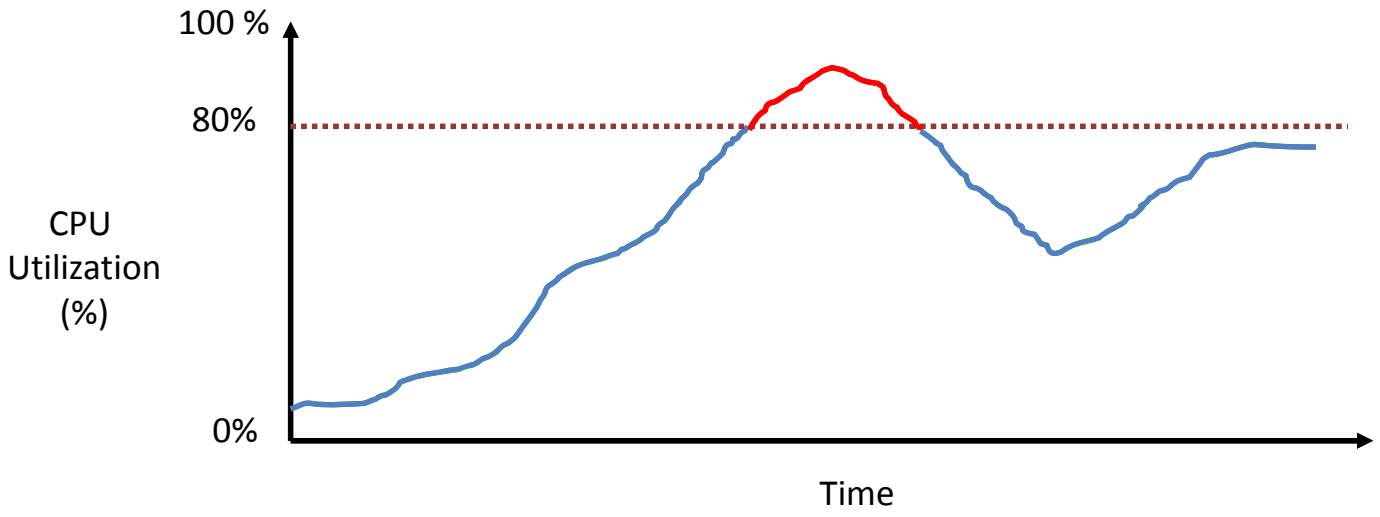
- VMs containing an application server with a test application
- Simple stateless application (can be a WebService)
- Test application mainly utilizes CPU resources
- No issues like databases replication are taken into account
- Concept of the VM Group – a set of load balanced machines with the same base image (running the same application, performing the same task)

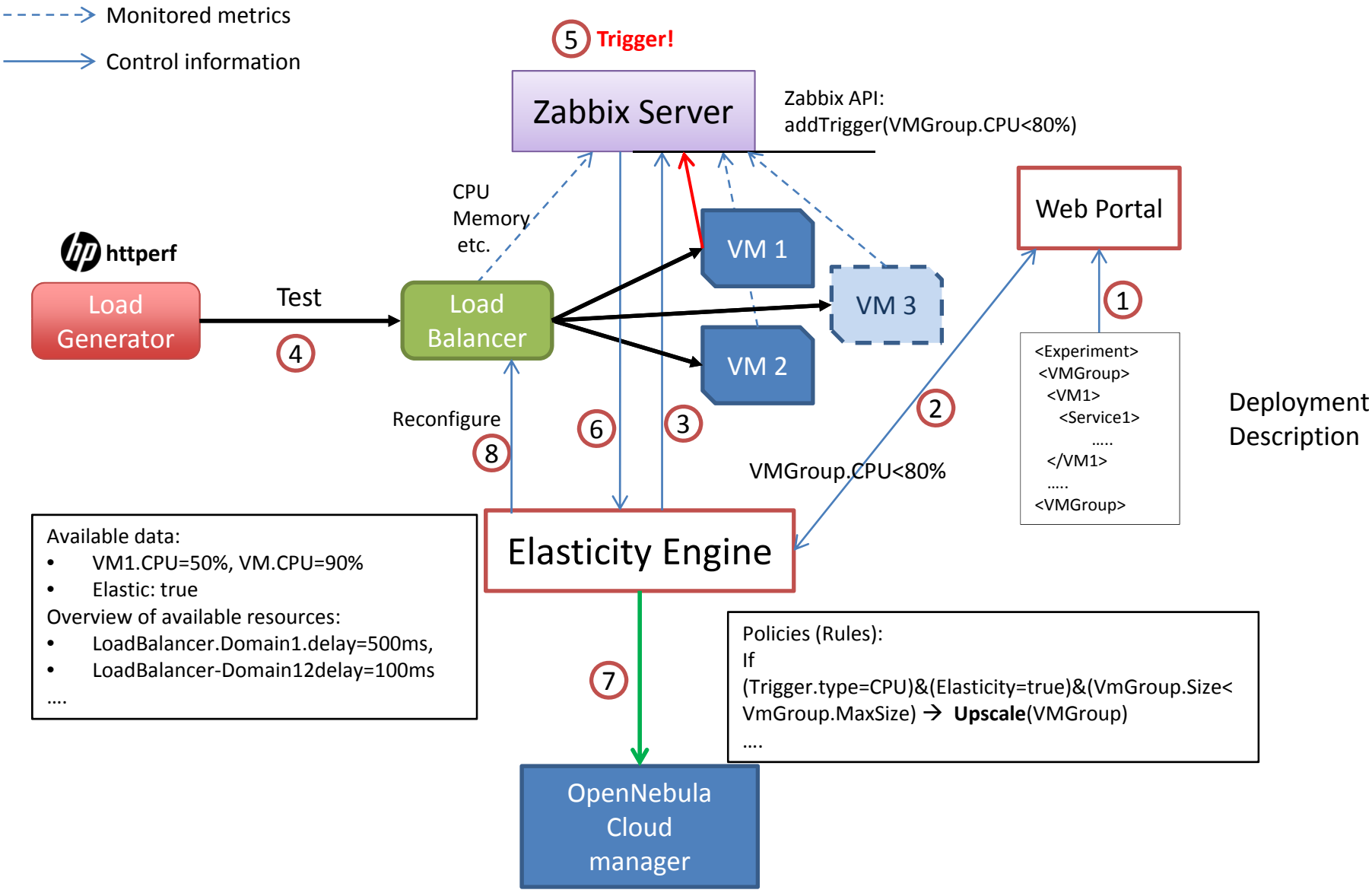
Upscaling/Downscaling



Upscaling: CPU-based

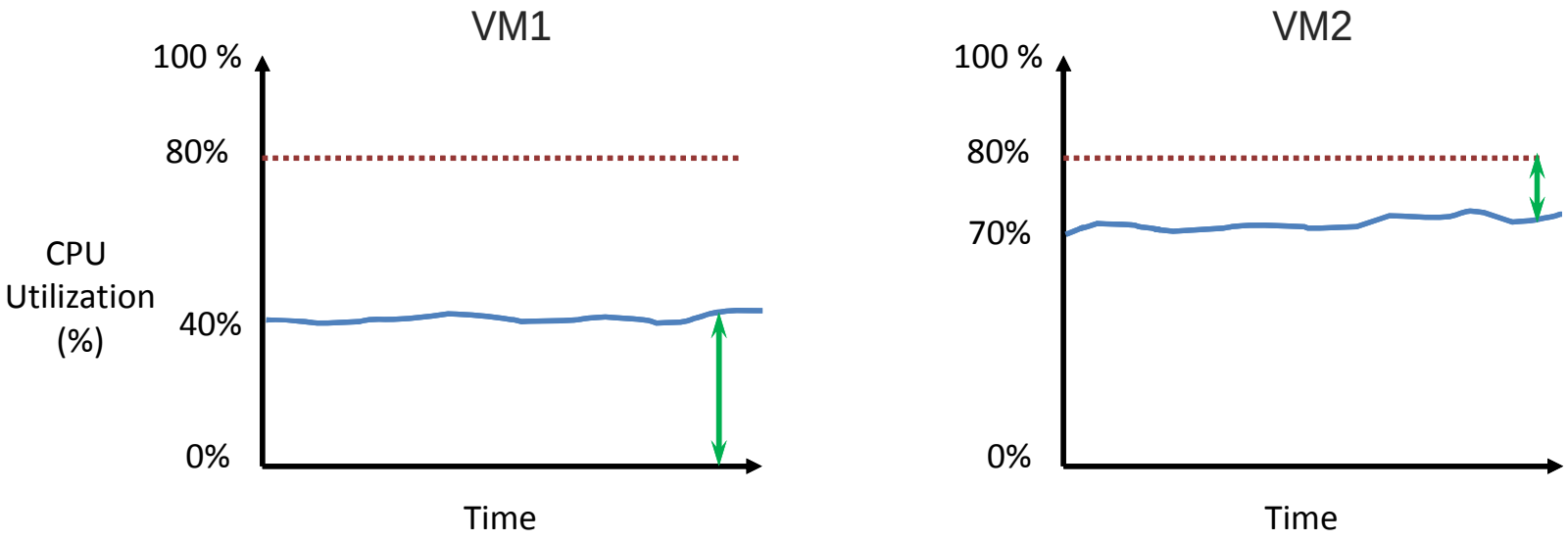
- Performed on the bases of actual resources utilization
- Meaningful thresholds have to be defined to specify the moment of reacting
- Zabbix provides flexible triggers mechanism




Monitored metrics
Control information
5 Trigger!
Zabbix Server
Zabbix API:
addTrigger(VMGroup.CPU<80%)
CPU
Memory
etc.
Web Portal
httperf
Load Generator
Test
4
Load Balancer
VM 1
VM 3
VM 2
1
<Experiment>
<VMGroup>
<VM1>
<Service1>
.....
</VM1>
.....
<VMGroup>
Deployment Description
2
VMGroup.CPU<80%
Reconfigure
8
6
3
Available data:
• VM1.CPU=50%, VM.CPU=90%
• Elastic: true
Overview of available resources:
• LoadBalancer.Domain1.delay=500ms,
• LoadBalancer-Domain12delay=100ms
....
Elasticity Engine
7
Policies (Rules):
If
(Trigger.type=CPU)&(Elasticity=true)&(VmGroup.Size<
VmGroup.MaxSize) → Upscale(VMGroup)
....
OpenNebula Cloud manager

Downscaling: CPU-based

- Checks performed regularly (no Zabbix triggers used, data is polled)
- Decisions made on the basis of estimation



Downscaling Model

U_i – CPU utilization of the i'th VM (%)
C_i – cooeficient, proportional to the i'th VM's CPU capacity (MHz)
N – current number of VMs in a group
T_{up} – threshold for the upscale trigger (%)

amount of load to be redistributed

$$\frac{U_k \cdot C_k}{N-1} + U_i \cdot C_i \leq T_{up} \cdot C_i, \quad \forall i \in N$$

existing load on the i'th VM

k – index of the VM being estimated as a candidate for shutdown

Downscaling

U_i – CPU utilization of the i'th VM (%)
C_i – cooeficient, proportional to the i'th VM's CPU capacity (MHz)
N – current number of VMs in a group
T_{up} –threshold for the upscale trigger (%)

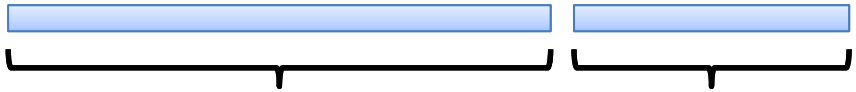
$$\frac{U_k}{N-1} \cdot \frac{C_k}{C_i} + U_i \leq T_{up}, \quad \forall i \in N$$

estimated utilization of i'th VM after downscaling (%)

k – index of the VM being estimated as a candidate for shutdown

Network implications

- Service response time perceived by the user:



Network delay (Round-Trip Time)

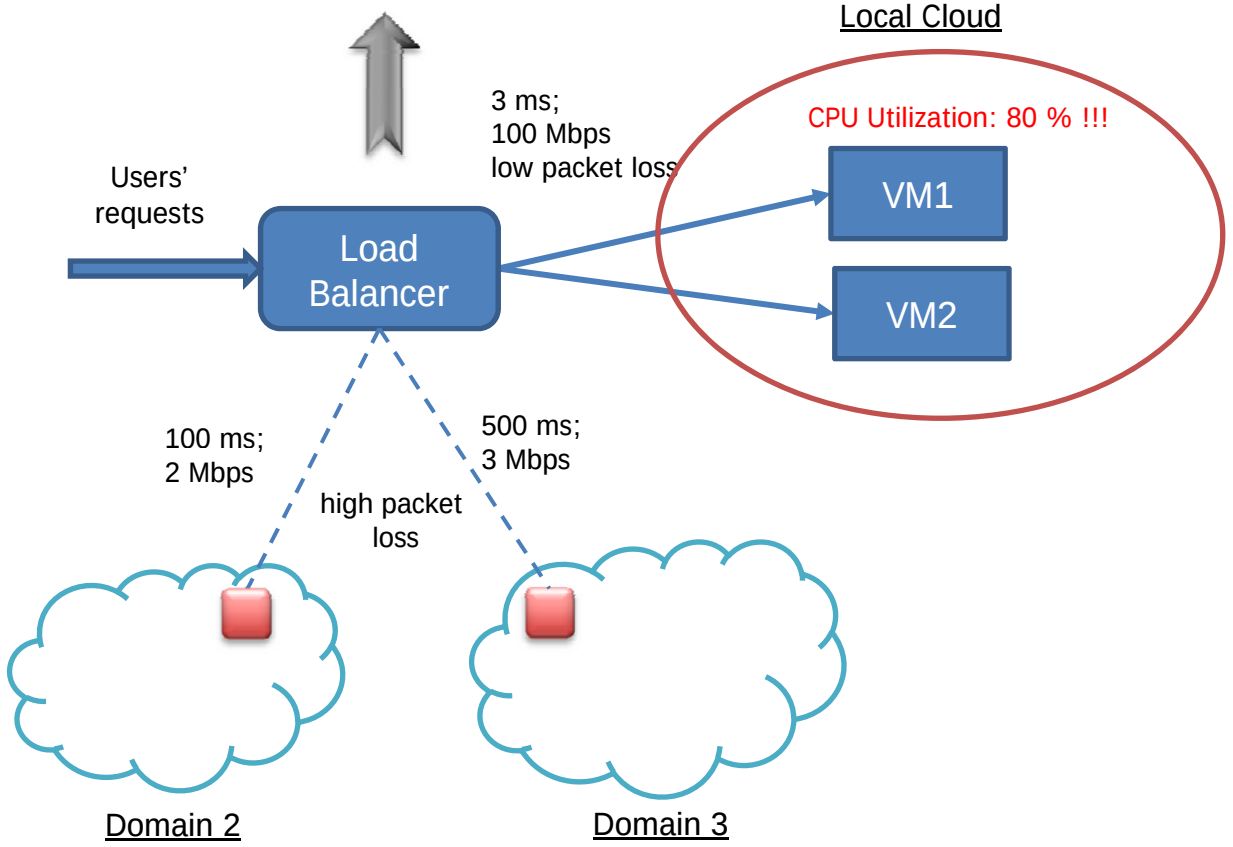
Application delay (Request processing time)

- Response time defines the Quality of Service and Quality of Experience
- Further assumptions: Cloud-Brokerage Scenario

Cloud Brokerage: Network Aspects

Domain agent:

```
Feb  6 12:14:14 localhost \
    haproxy[14389]: 10.0.1.2:8080 [06/Feb/2009:12:14:14.655] http-in \
    static/srv1 10/0/30/69/109 200 2750 - - ---- 1/1/1/1/0 0/0 {1wt.eu} \
    {} "GET /servlet1/getWeather.do?city=Berlin HTTP/1.1"
```



Summary

- There is a possibility to build a solid elasticity enabler based on the open-source components
- Providing a generic Elasticity framework
  - allows controlling resources on a broad range of cross-cutting monitored data (CPU, MEM, Storage, SET, Delay, Loss, etc.)
  - allows to improve the QoS and QoE while minimizing the costs
- There is a broad range of open questions regarding optimal policies and thresholds
  - VM deployment rules (e.g. size, location, migration)
  - load balancing rules (e.g.round robin DRR, weighted round robin WRR)
  - thresholds for triggers
  - network implications

Thank you for your attention!