Evaluating Cloud Elasticity of an 3-Tier Application with Experiments and Simulations

Christian PEREZ & Jonathan ROUZAUD-CORNABAS
http://avalon.ens-lyon.fr

2014, July 3rd

Agenda

Context

RUBBOS: a 3-Tier Benchmark

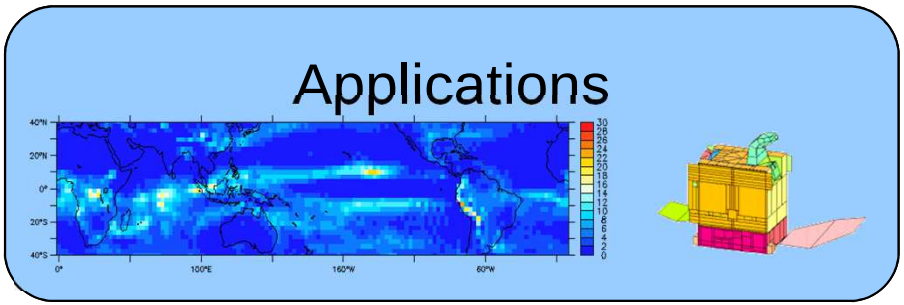
Grid'5000: An Hardware-as-a-Service Platform

Towards a performance model of RUBBOS

Using a performance model in SGCB A Cloud Simulator

Conclusion

Context: Efficiently Running an Application



Programming Models

Algorithmics

Application & Resource Models

Resource Management

CPU/data-intensive Applications
- Parallel applications
- Workflows
- Code coupling
- MapReduce

Computing platforms
- Different characteristics
  - Performance, energy, size, cost, reliability, QoS, etc.
- Hybridization
  - Sky computing, HPC@Cloud, Exascale, Spot instance

Objectives
- Expressiveness simplicity
- Application portability
- Resource specific optimizations
  - Elastic resource management
  - Energy consumption
  - Data management

Super-computers (Exascale)	Grids (EGI)	Desktop Grids	Clouds (IaaS, PaaS)
Large scale	Heterogeneity	Volatility	On demand

N-Tier Applications

N-Tier Applications

- Applications made of several (N) tiers
- Variation of workflow applications

Example

- **Classic 3-tier application: HTTP Server, Application Server, Database**
  - HTTP Server: Apache
  - Application server: Tomcat
  - DB: Mysql

- **Large part of applications running on Cloud**
  - With batch jobs such as analytics

3-Tier Applications and Cloud

Motivation

- Easy to deploy one tier per VM
  - e.g. 3 VMs: 1 HTTP Server VM , 1 App Server VM and 1 DB VM
- Easy to scale with add load balancer between instances of each tier VM
  - e.g. 1 HTTP Load balancer (such as Apache mod_proxy or nginx) can distribute the load between 100s of HTTP VMs.
- Difficult to scale DB tier
  - Not scalable because of ACID properties
  - But, large website (Facebook, Twitter, Airbnb) use NoSQL or limited ACID property (per line) to create scalable database

Cloud enables automatically adaptation to the number of request by adding and removing VMs at each tier.

- Perfect for startup without knowledge how quickly their mobile apps/website will be adopted

RUBBOS: A 3-Tier Use Case

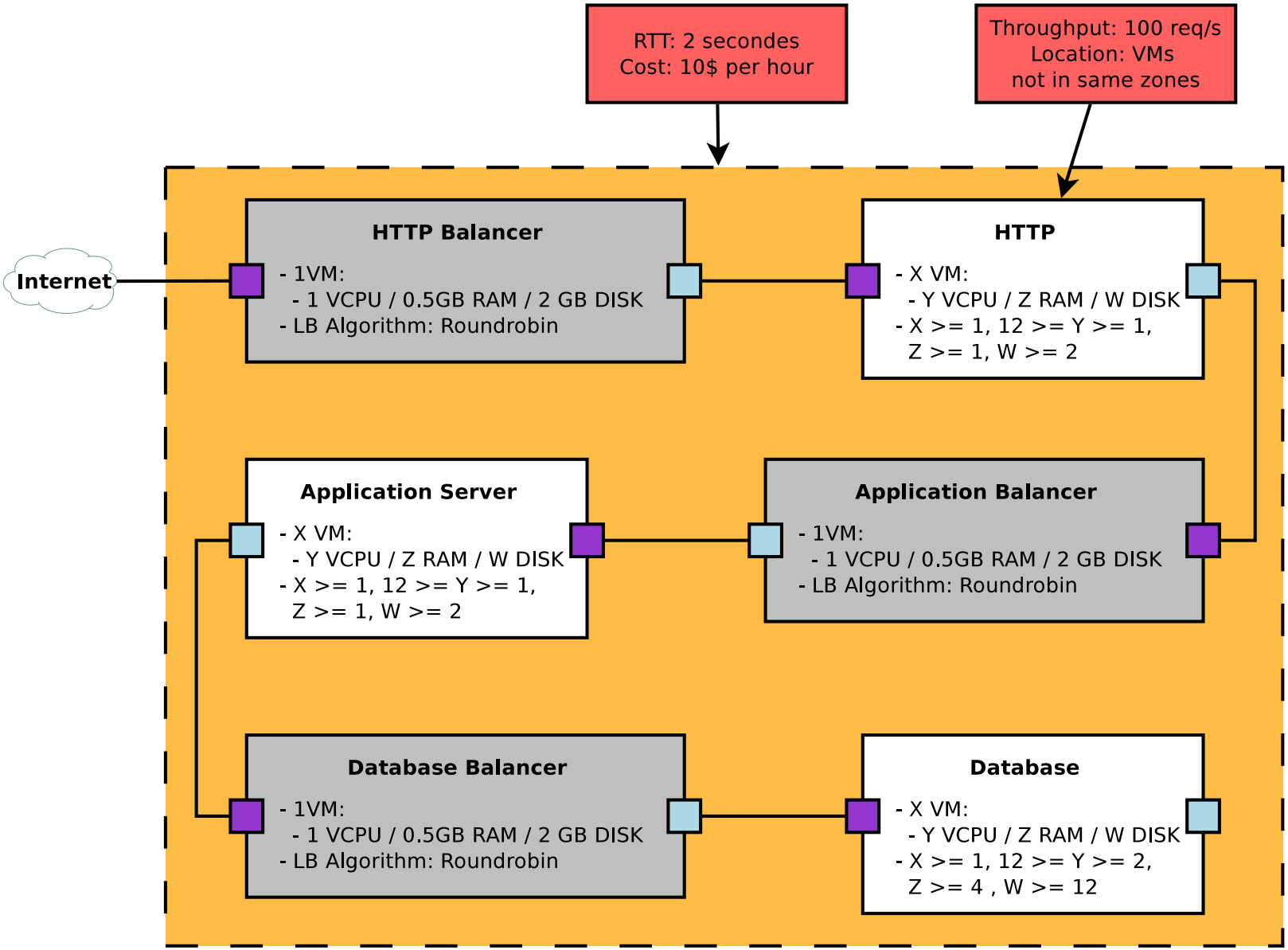
RUBBoS is a bulletin board benchmark

- Model an online news forum like Slashdot.
- A Java Middleware Open Benchmarking (JMOB) project of OW2
  - http://jmob.ow2.org/rubbos.html

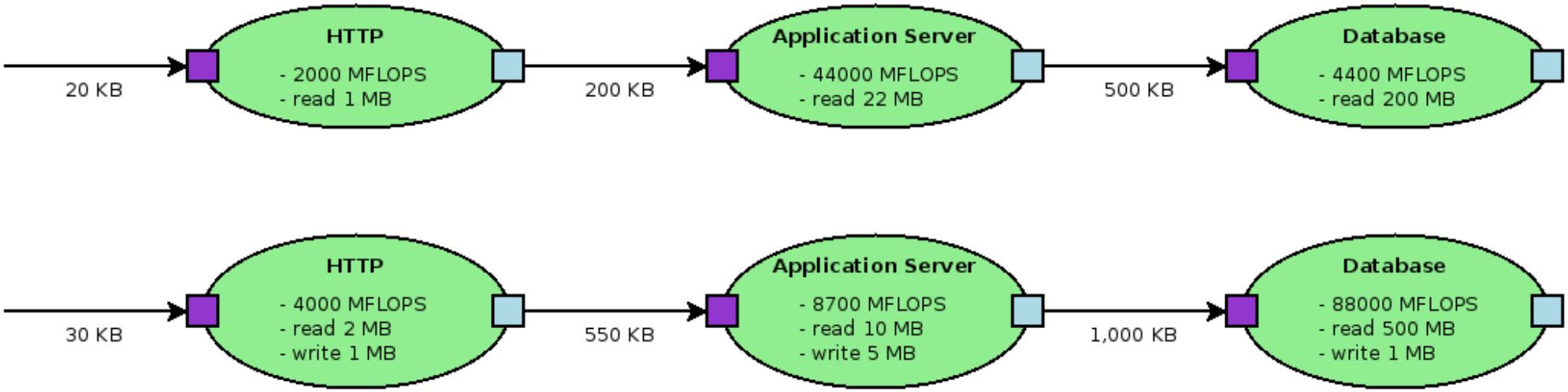
Well known and used benchmark for n-tier architecture

- Based on a classical 3-tier architecture
  - HTTP Server, Application Server and Database
- 2 application server instantiation exists
  - PHP and Java Servlets
- Client workload generator
  - Based on the analysis of the behavior of real users on real web site.

RUBBOS: Architecture Model



RUBBOS: Data Flow Model



Application Architecture is not Enough!

Given an application model, how to select the right value for each parameter ?

- Which is the request distribution over time ?
- Do the request characteristics are heterogeneous or homogenous ?
- How the requests flow between tiers ?

We need a performance model, including a client model

- What QoS a client is expecting?

Modelling Application Performance Model

Two approaches to model application performance

- White box approaches
  - Queue theory + a few experiments to calibrate
  - Require a very fine knowledge of the application and how it works
- Black box approaches
  - Run the application with all the different HW/SW combinations
  - Use statistical analysis to derive an analytical model of its behavior
  - Do not require fine grain knowledge
  - Require to run extensive (and costly) experiments

Let explore the black box approach!

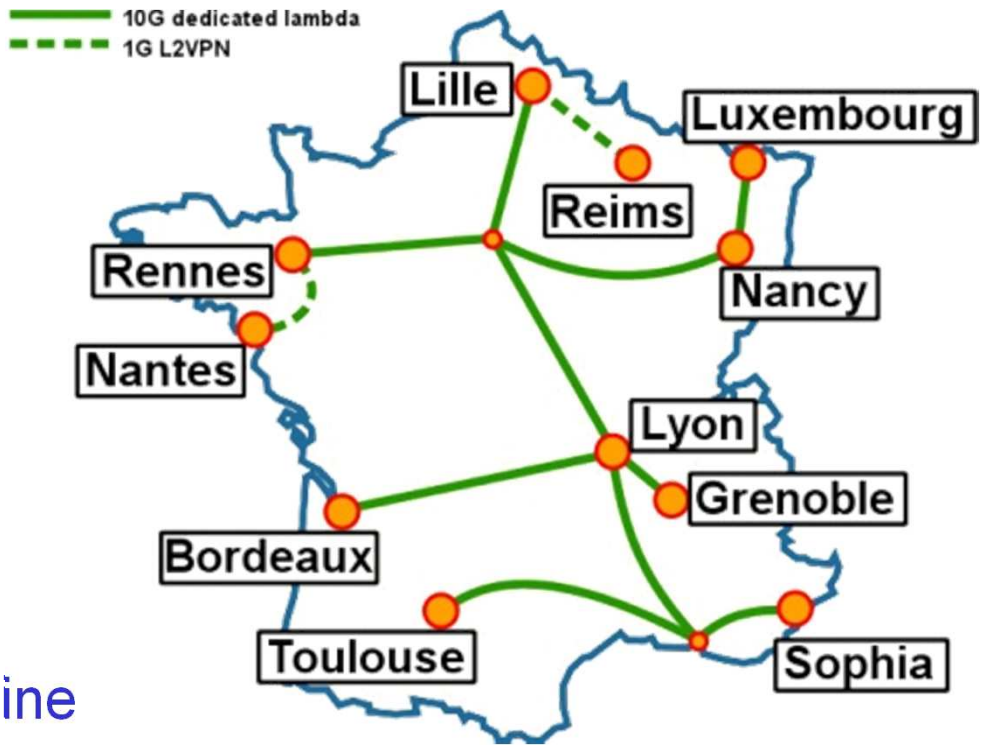
Grid5000:
A Hardware-as-a-Service Platform since 2003

Platform

- 10 sites
- ~ 26 clusters
- ~ 1260 nodes
- ~ 8000 cores

Dedicated 10G backbone

- Provided by Renater.



Tools

- Versioned, extensive and machine readable format description
  - cluster, network, PDU
- Oar to reserve
- Kadeploy to deploy an image on the bare hardware
- Kavlan to manage multi-site virtual network

Grid’5000: Examples of Usage

Deploying and studying gLite on Grid’5000

Automatic deployment of OpenStack on Grid’5000

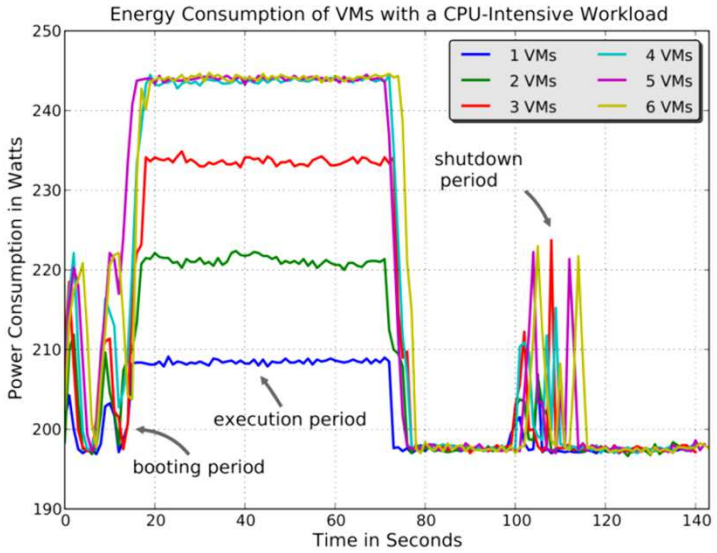
Large scale VM management (10.000+ VM on 512 nodes)

Energy consumption analysis

Large scale B&B algorithms (150.000+ peers)

MapReduce experiments

….



Towards a Performance Model of RUBBOS

Goal

- Obtain a performance model of RUBBOS

How

- Running RUBBoS with different configuration parameters and understanding how they impact metrics (throughput, RTT, cost)

Methodology

- Define all the parameters ranges
  - Number of VMs per tier, number of resource for each tier VM (core, memory), VM to PM mapping (how to (not) consolidate your n-tier applications)
- (Instrument application)
- Generate all the different combination of parameters
- Reserve resources on a computing platform (G5K in our case)
  - Hard, boring and easy to make errors: At least few thousands combinations of parameters....
  - VM5K (and Execo) automates the whole process !
    - Code it once, then sit and wait that all the combinations have been run.

Experimental Methodology

Deploy the VM image

- One per tier + one for each tier load balancer

Configure all the VM parameters

- Disk size, memory, cores, VCPU to CPU mapping, distribution of VMs on multiple PMs

Boot all the VMs

Retrieve the IP adress of all the VMs

Configure all the tiers (Apache, Tomcat, MySQL)

- And also the load balancer
  - Apache mod_proxy for HTTP and Application server, HAProxy for DB

Restart the DB, then the LB DB, then Tomcat, Tomcat LB, HTTP, HTTP LB

RUBBoS is running now !

Launch the client workload generator

Wait it finish

Gather all the logs and store them for later

Experiments

Experimental Setup

Platform

- A cluster for Grid'5000
- Node = multi-core high end machine
- GigaEthernet Switch

RUBBOS

- #VM & #core for HTTP
- #VM & #core for Tomcat
- VM=1 & #core for DB

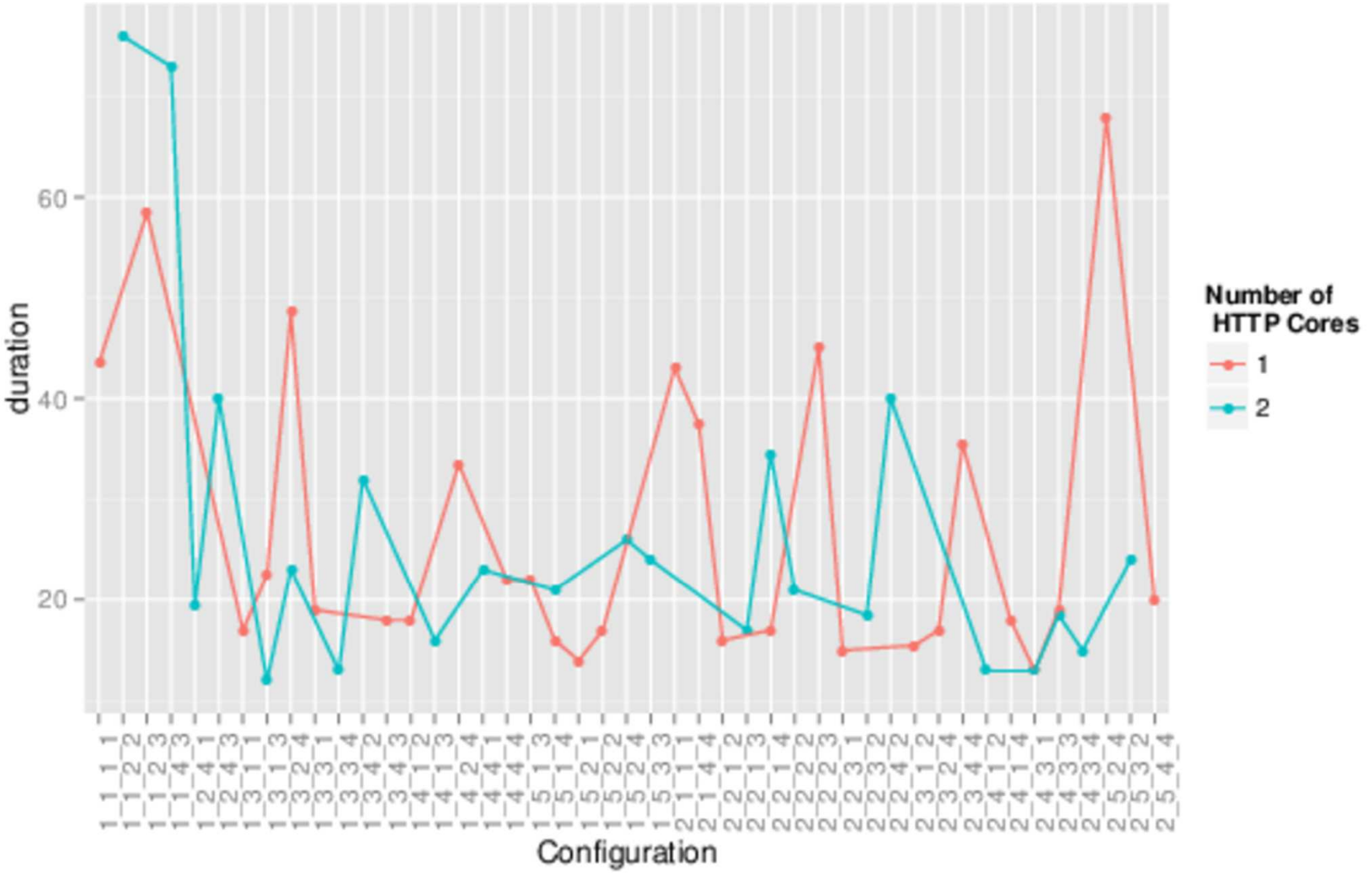
Mapping parameter

- All VM on the same node
- A VM per node

5,000 different combinations of parameters' values

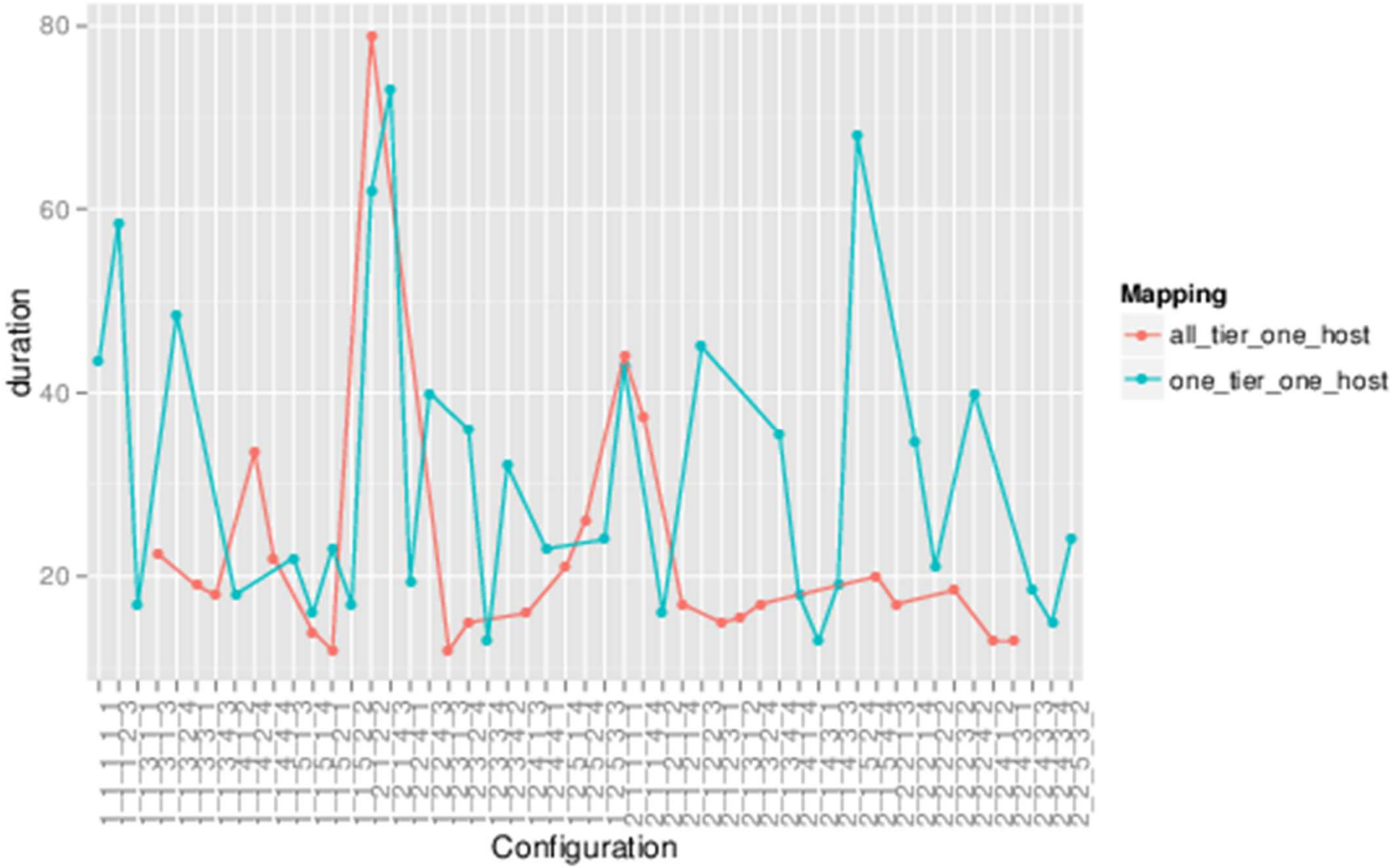
Each experiment runs for 20min on up to 4 nodes

HTTP Core matters



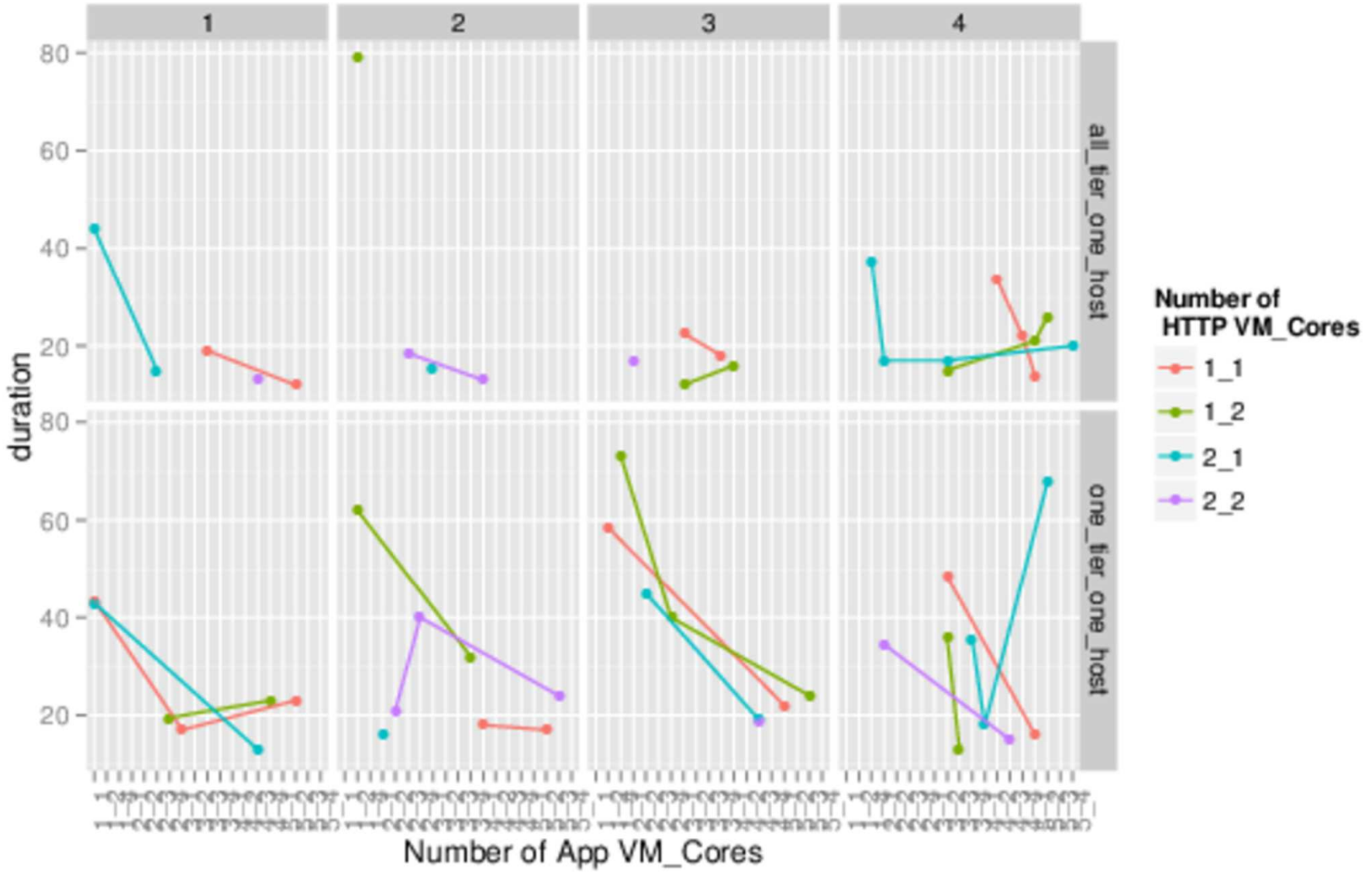
Legend: #VMHTTP_#VMHTTPCore_#VMApp_#VMAppCore_#VMDBCore

Mapping Matters

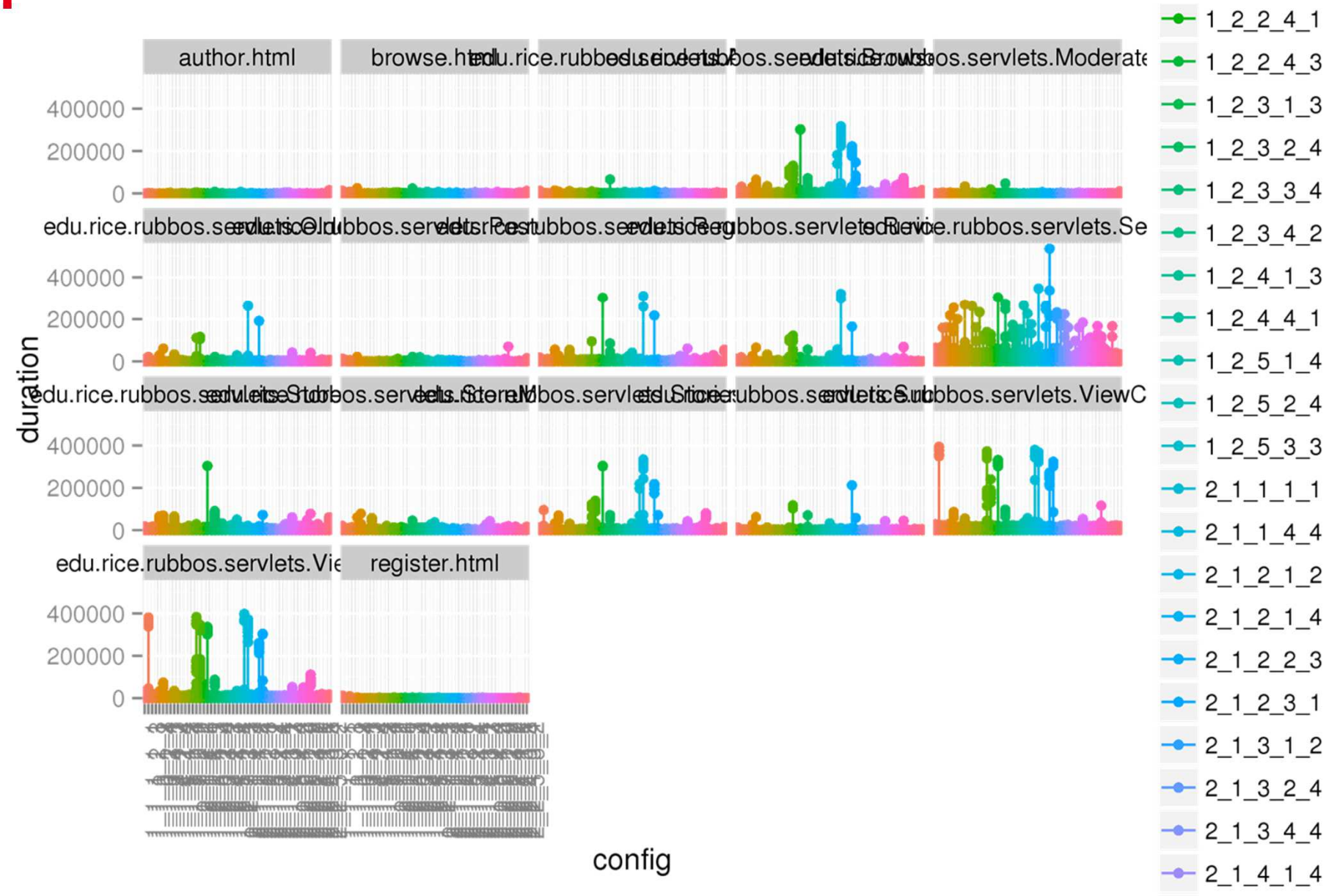


Legend: #VMHTTP_#VMHTTPCore_#VMApp_#VMAppCore_#VMDBCore

All Parameters Matter

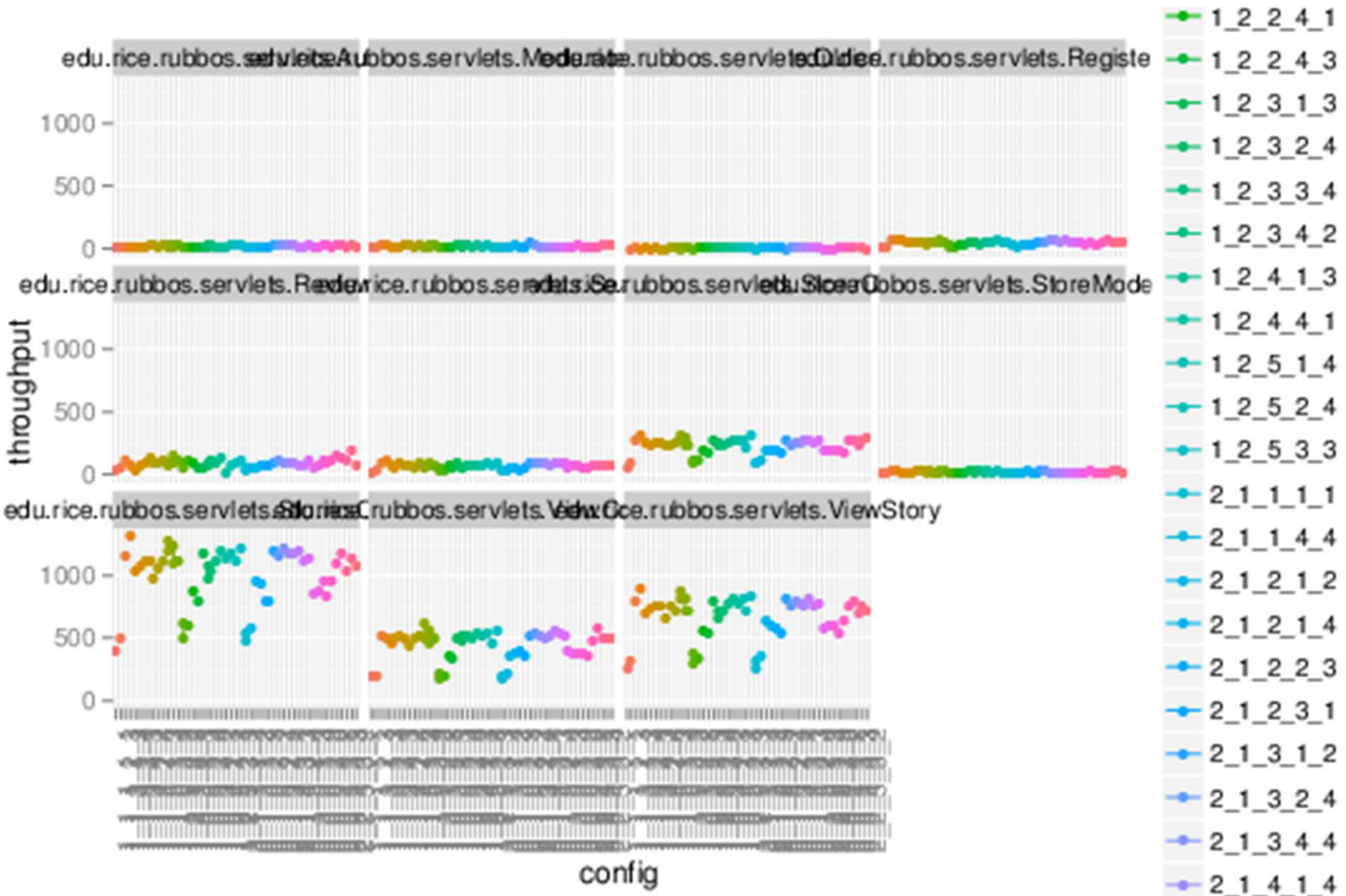


Experiment Results: RTT for all Services



Legend: #VMHTTP_#VMHTTPCore_#VMApp_#VMAppCore_#VMDBCore

Experiment Results: Throughput



Legend: #VMHTTP_#VMHTTPCore_#VMApp_#VMAppCore_#VMDBCore

Simulating a Cloud Application with SGCB

Towards Simulation

Let assume we have a (RUBBOS) performance model

Next step: design algorithms

- ie how to manage the elasticity of the application?

How to evaluate and compare such algorithms?

- Real experiments
  - Accurate
  - Very complex and very time-consuming
  - What about reproducibility?
- Simulations
  - Quite accurate (provided the performance is ok)
  - Less complex and less time-consuming
  - Reproducible

SimGrid and SimGrid Cloud Broker (SGCB)

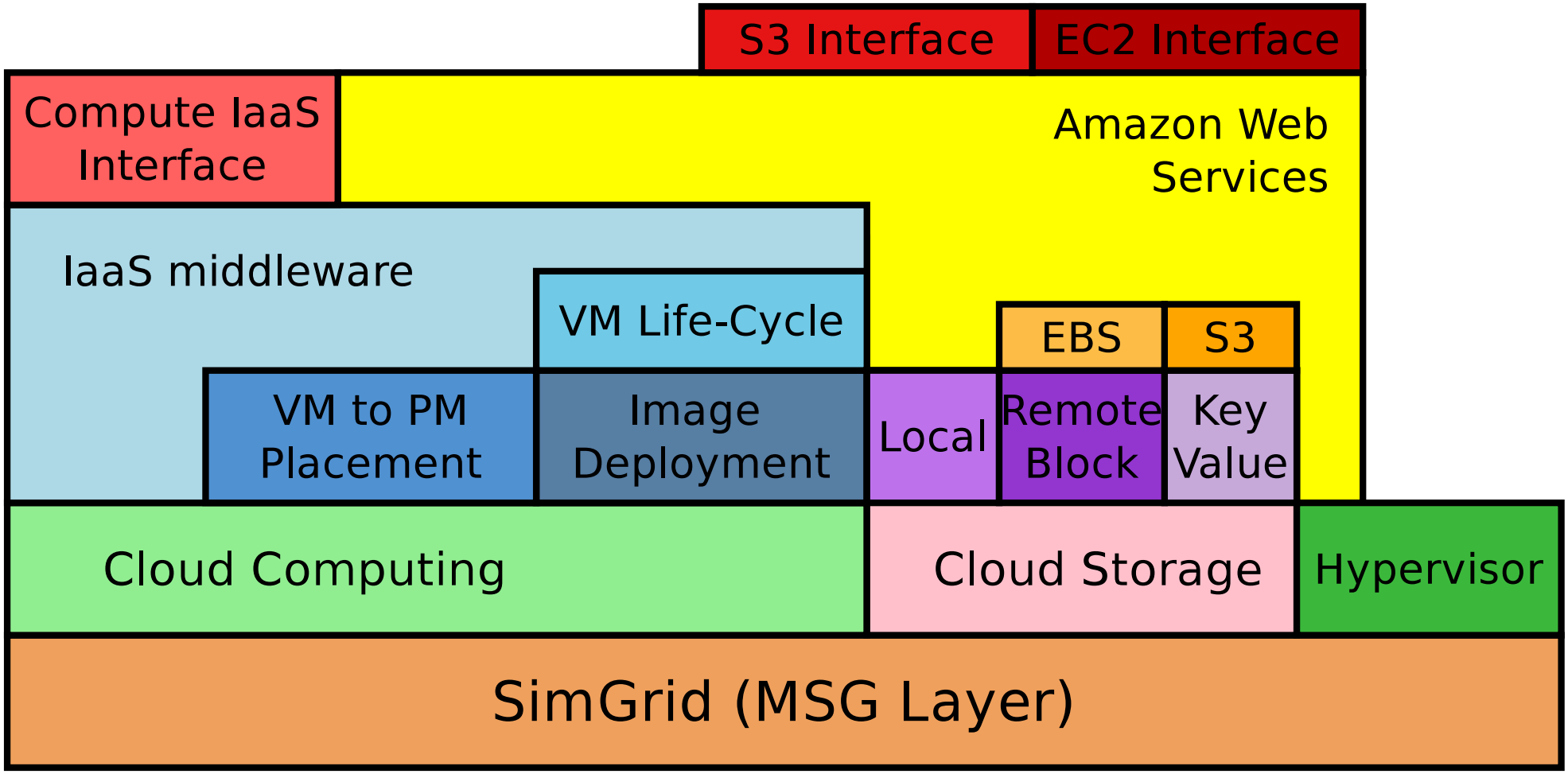
SimGrid

- Developed by 3 French research teams (Algorille, Avalon, Mescal) and the University of Hawai'i at Manoa (USA)
  - http://simgrid.gforge.inria.fr/
- simulates many different distributed systems
  - clusters, wide-area and local-area networks, peers over DSL connections, data centers, etc
- has models are theoretically and experimentally assessed
- is scalable
- exists for 13 years

SimGrid Cloud Broker: a SimGrid extension

- From an user point of view
- From a cloud provider point of view
- Evaluates application(s) running on Clouds
- Evaluates different policies in Cloud middleware
- Multi-Clouds (private and public)

SGCB: Architecture Overview



SGCB: Amazon Web Services

S3 / EC2 API

All the instance types

All the regions of AWS

On-demand and Spot Instances

3 types of storage: local, EBS and S3

Accounting of network, computing and storage resources

Models

- Spot instance prices: smart random, file and prediction model
- Life cycle of VMs
- Storage (3 models)
- Compute
- Network: VM inter/intra-regions and S3

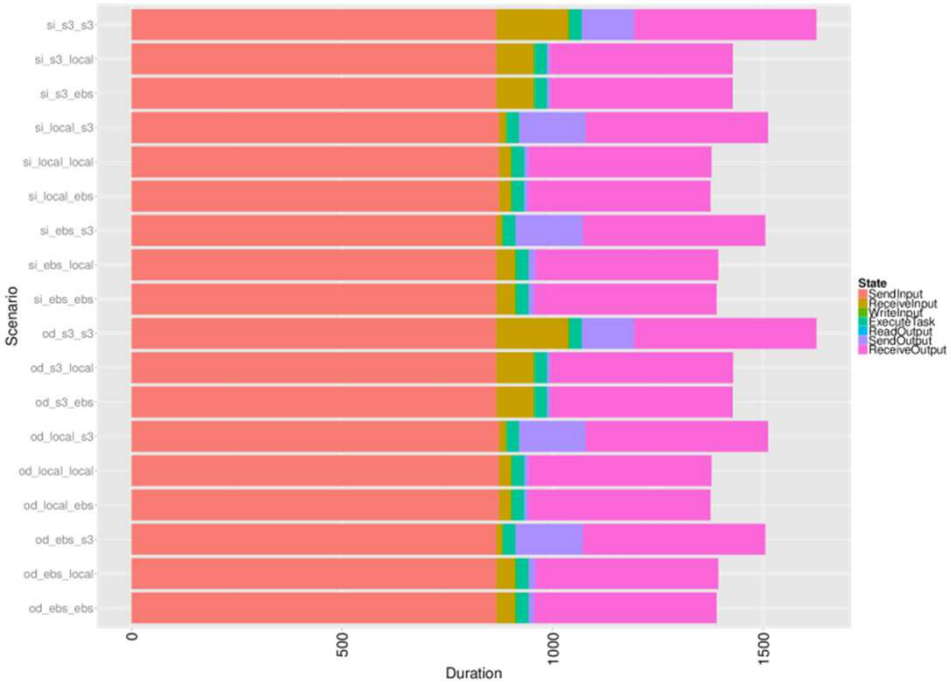
Model calibrated by running many experiments on AWS

SGCB Example

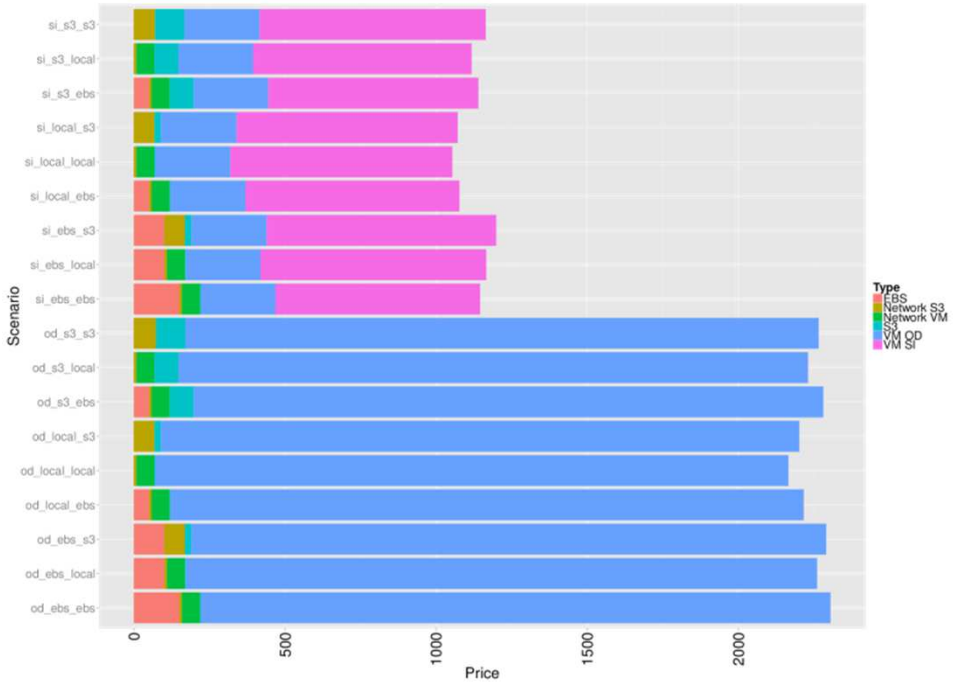
N* Tasks and *no* dependency between them but *a large number of parameters

Three parameters (I, O and FLOPS) for tasks in BoT (impact task allocations)

- Homogenous
- Stochastic (uniform/bimodal/heavytail)



Impact of Storage Policies on Completion Time



Impact of Storage Policies on Billing

Conclusion

Conclusion

Application architecture description

- is important, in particular for deployment
- but not enough: application performance is needed for deciding parameters!

Obtaining an application performance model

- Manually
- Statistically through many experiments (this talk)
- Automatically learned (FP7 PaaSage)

Platforms and tools help a lot

- Grid'5000 as an HaaS for research in computer science
- SGCB: an extensible cloud simulator

On going work to finish modeling a (simple!) cloud application