EUBRA BIGSEA

EUROPE - BRAZIL COLLABORATION OF BIG DATA SCIENTIFIC RESEARCH THROUGH CLOUD-CENTRIC APPLICATIONS

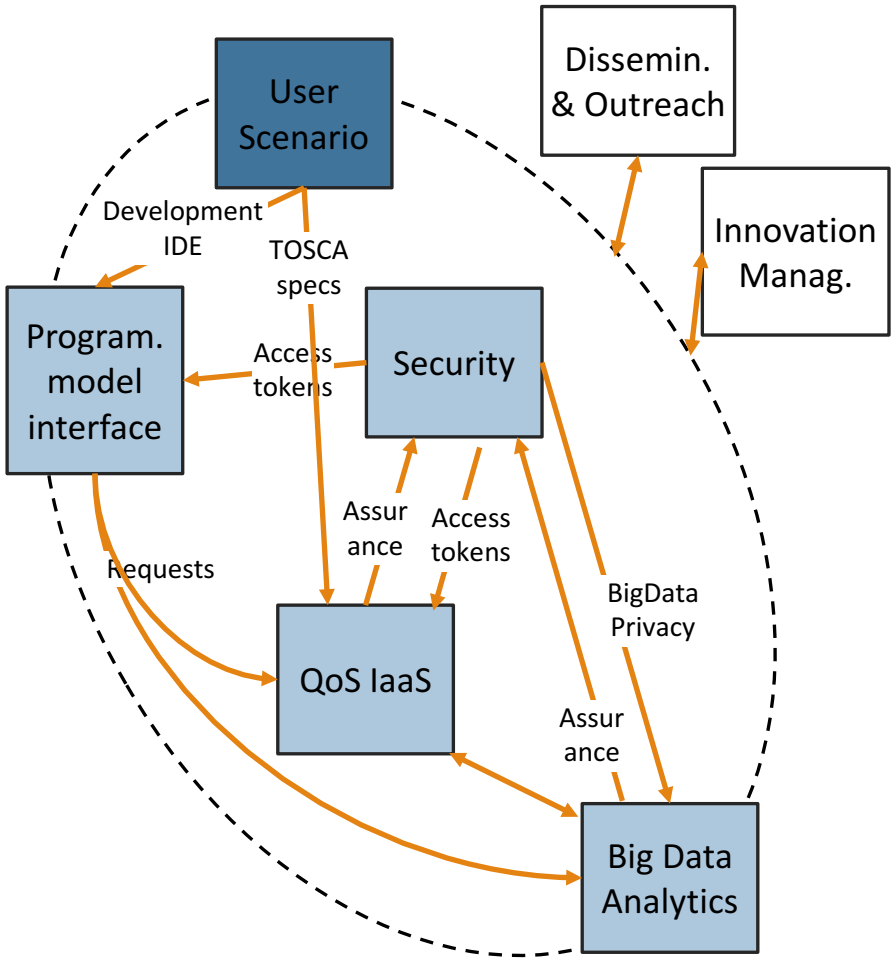
# EUBra-BIGSEA: cloud services with QoS guarantees for Big Data Analytics

*Ignacio Blanquer on Behalf of the EUBRA-BIGSEA Consortium*

# EUBra-BIGSEA

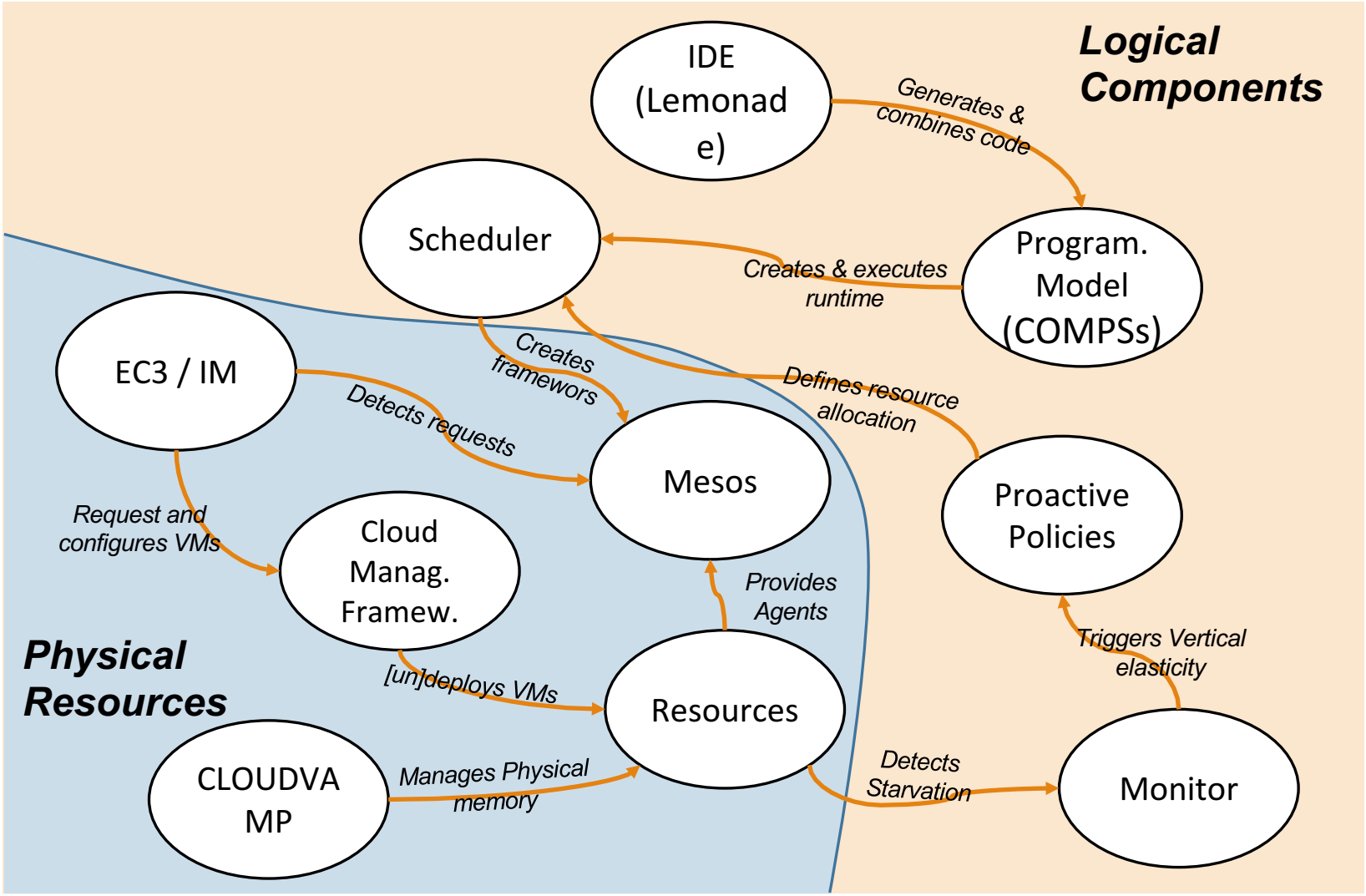
- A European-Brazilian Consortium aiming at
  - The development of QoS and secure cloud services to support Big Data.
  - The development of Big Data services for capturing, federating and annotating large volumes of data.
  - The use of efficient technologies for guaranteeing the fulfilment of the security and privacy policies.
  - The transfer of this technology to a real user scenario with high social and business impact, and of high interest for both EU and BR.

# BIGSEA in a NutShell




- BIGSEA is structured along 7 (+ Project coordination) activities defined as WPs.
- User Scenario will make use of the Data Analytics API and the Programming model interface.
- The Programming model interface will use the dervices deployed using TOSCA standard
  - Service specifications will be instantiated on a cloud IaaS, automatically managing Quality of Service.
  - The BigData Analytics services will run on the cloud infrastructure.
- A security framework will be defined to provide security and privacy.
- Globally, dissemination and exploitation activities will pursue the outreach and transfer of technology.

# BIGSEA Architecture



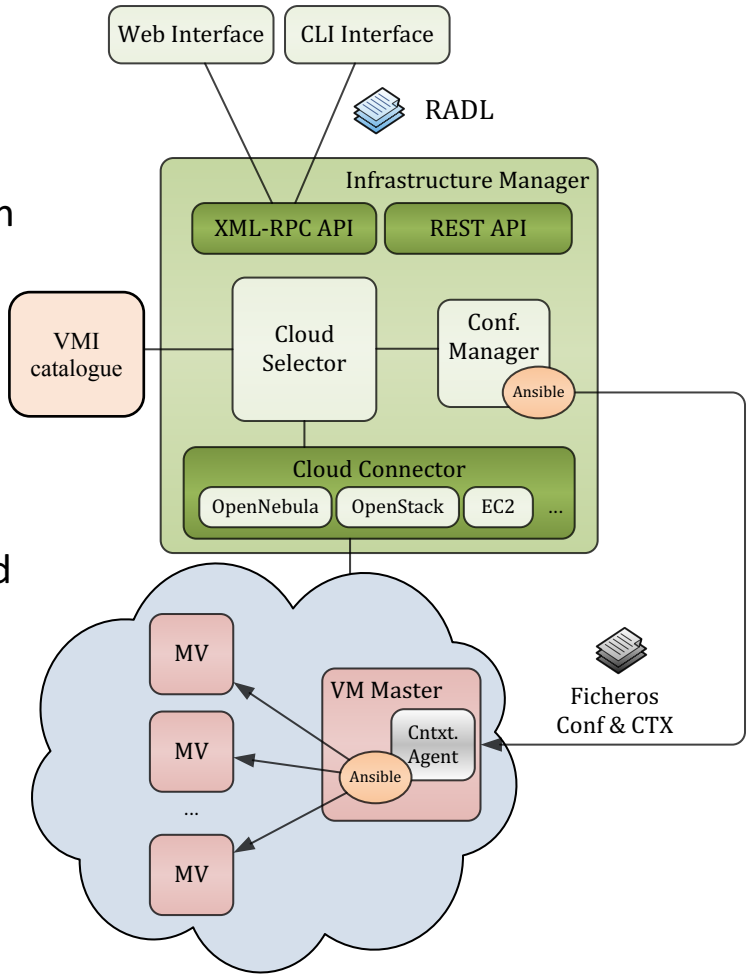
# Management of Physical Resources

# Physical Resources

- Resources are provisioned by a Mesos Cluster to the execution frameworks
  - COMPSs, Marathon, Spark, Chronos.
- Mesos Cluster is composed of a set of VMs provisioned from a CMF
  - Automated Deployment
  - Platform-agnosticism
  - Vanilla VMIs
- Horizontal Elasticity at the level of the resources
  - The registration of new frameworks will check for the availability of the needed resources.
  - New requests will trigger booting up new resources if needed.
  - Idle pure computing resources will be powered off.
- Vertical Elasticity at the level of the VMs
  - Higher flexibility in the allocation of physical resources.

# Infrastructure Manager (IM) www.grycap.upv.es/im

- Two approaches to VMI management
  - Deploy existing vanilla images (plain OS) and configure instances using tools to re-create the desired conf.
    - Example: Instantiate a plain Ubuntu 12.04 AMI on Amazon EC2 and use Ansible to automatically install a full LAMP (Apache, MySQL, PHP) stack.
  - Create specific VMIs for different Clouds from templates.
    - Deployments based on recipes, configuration and contextualization services.
  - Enables platform-agnostic or hybrid deployments.

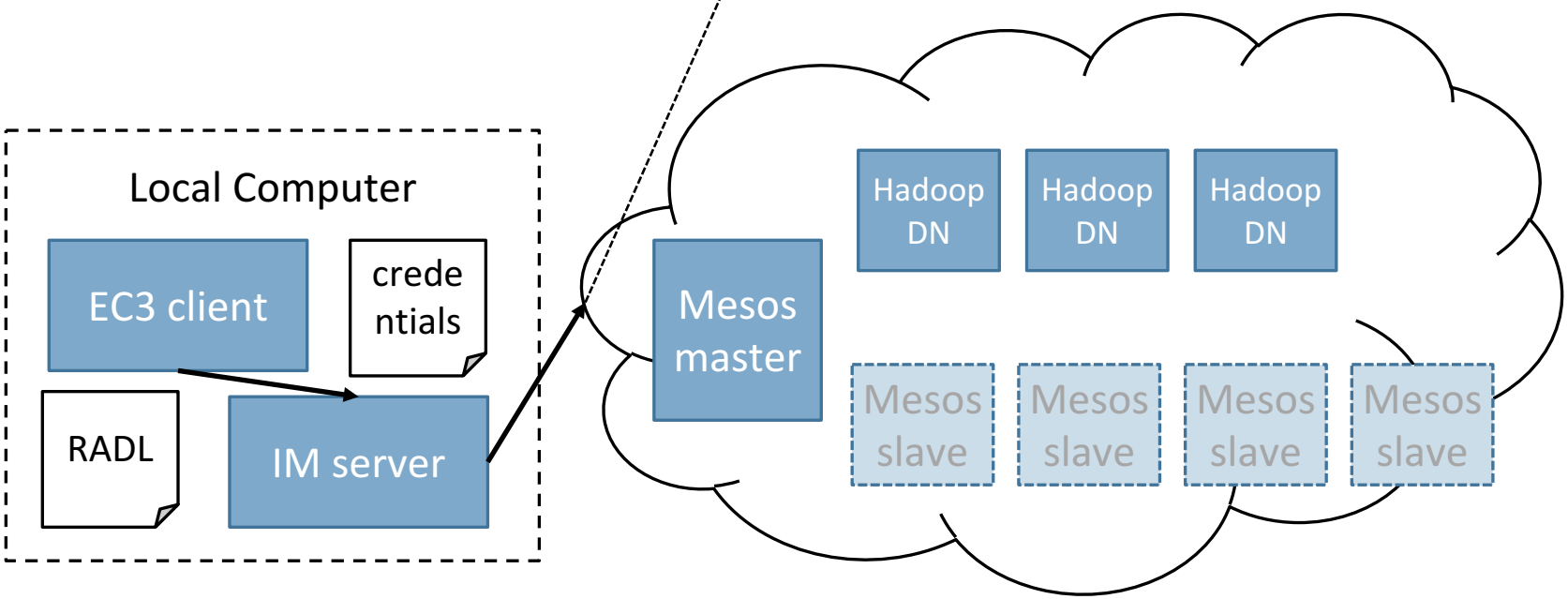


# Platform-agonstic Deployment

```
$ docker pull eubrabigsea/ec3client
$ docker pull grycap/im
```

Bound to ONE, but easy to extend by:

- Changing base image
- Providing credentials




```
$ ./ec3 launch myMesosHadClus ubuntu-one mesos docker spark nfs hadoop
-a /root/auth.dat -u http://localhost:8899
```

Credentials

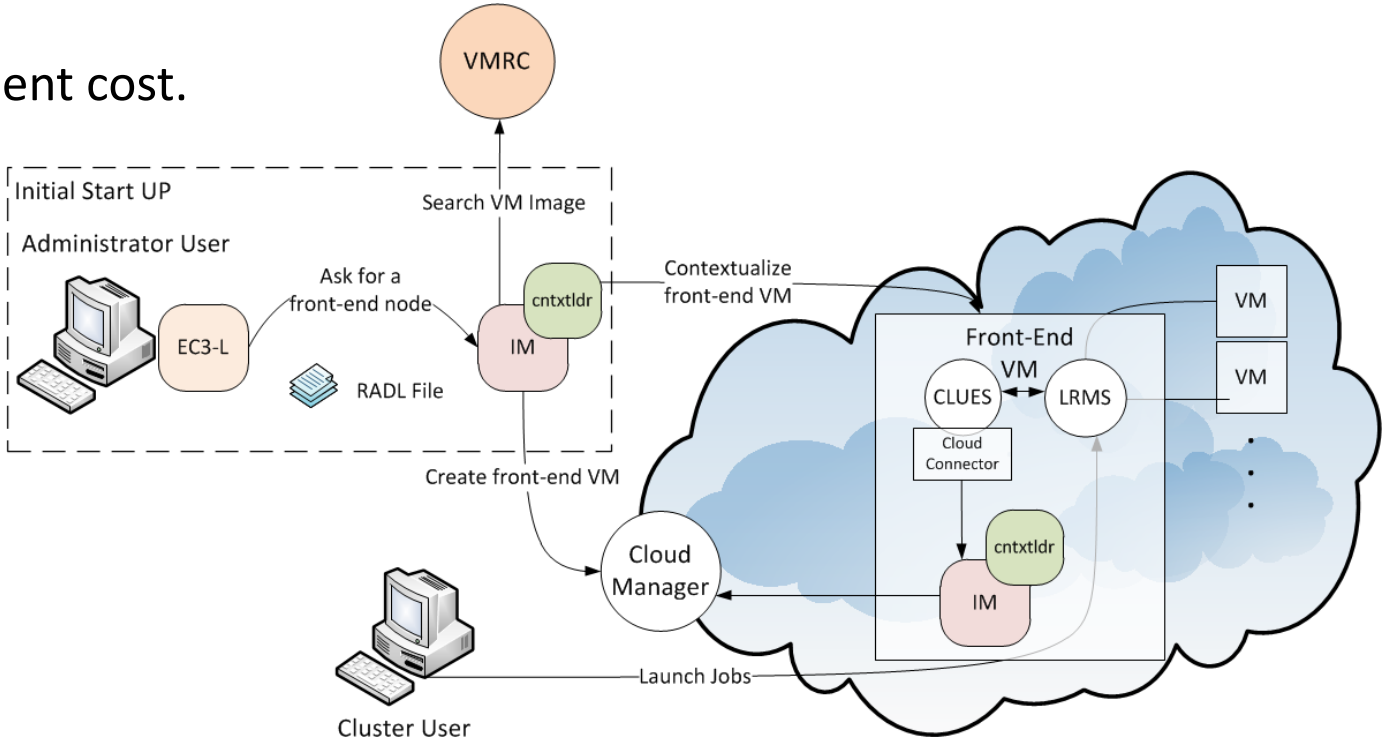
Address of the IM server

# Platform-agnostic Deployment

- Three parts:
  - Cloud-backend (**`ubuntu-one.radl`**)
    - Three objects: front, wn & wnmesos
    - Four attributes per object: name, image url, username and password.
  - Resource Management Framework (**`mesos.radl`**)
    - Defines network: ports, DNS names, interfaces.
    - Defines system virtual hardware: Memory, CPUs.
    - Configuration: Master and Slaves.
  - Other dependencies
    - **`docker.radl`**: On every node.
    - **`spark.radl`**: On every node.
    - **`nfs.radl`**: On every node.
    - **`hadoop.radl`**: on master & wn instances.

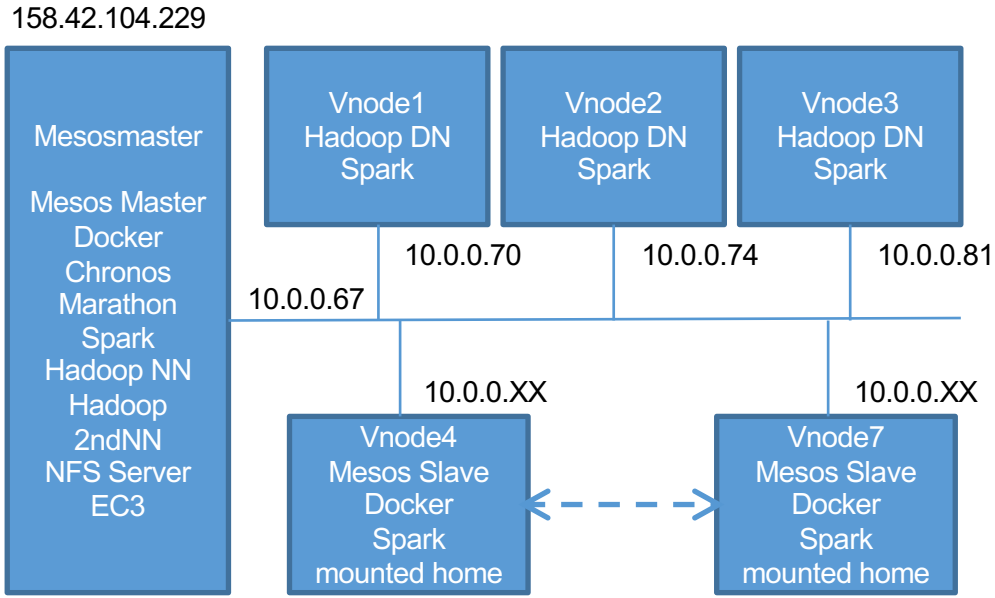
# Elastic Compute Clusters on the Cloud (EC3)

- EC3 (Elastic Cloud Computing Cluster) leverages CLUES to create elastic virtual clusters in the Cloud.
  - No upfront investment.
  - Customizable.
  - Usage-dependent cost.
  - Automatic elasticity.



# Resource-level elasticity

- When a new Mesos framework is requested, EC3 will check if there are enough resources to fulfill the request
  - Additionally, Marathon and Chronos plugins will capture the details of the request.
  - EC3 will request the Cloud Management Framework to deploy new VMs to allocate the necessary resources for the new framework
    - EC3 uses Infrastructure Manager to install and re-configure the software.
  - When resources become idle for a while, they are undeployed.
  - Data resources are not undeployed.



# Vertical elasticity at the resource level

- VMs are allocated from a CMF that manages a general-purpose on-premise cloud
  - Physical limitations are normally at the memory size rather than at the CPU share.
  - A physical machine can be shared among different deployments
    - Both BIGSEA stack or other one.
    - Or due to increased isolation.
  - Oversubscribing memory can enable reaching higher number of VMs per physical node, not bounding to a specific partitioning.
    - A VM from one deployment can ”release” the free memory to other VMs in the node.

# Infrastructure management

- Interaction with the CLUES client to get information about the resources powered on / off and to force them to be powered on / off.
- Mainly intended for administration, it can be used to anticipate powering on a resource .

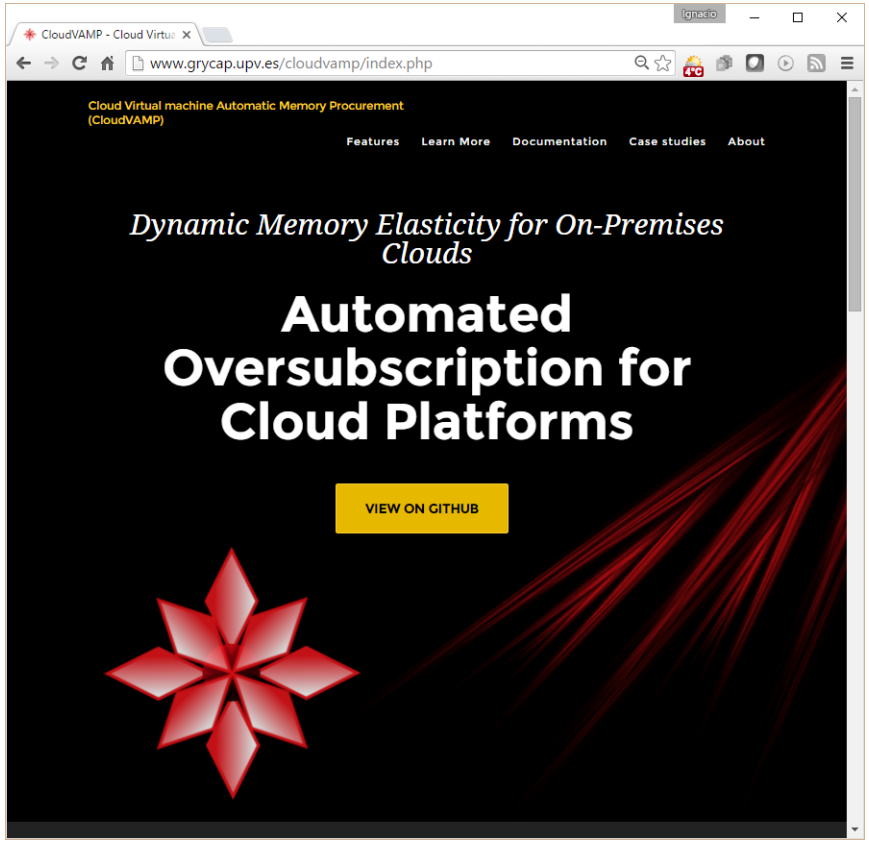
| | |
|---|---|
| `/v1/resource/slaves` | GET method, lists all the resources registered in the master or cluster of masters. |
| `/v1/resource/slaves/slaveid` | GET method, provides the status information of a specific slave. |
| `/v1/resource/up` | PUT method, boots up a new resource. |
| `/v1/resource/slaves/slaveid` | DELETE method, powers down a specific resource. |
| `/v1/resource/slaves/slaveid/enable` | POST method, enables a disabled resource. |
| `/v1/resource/slaves/slaveid/disable` | POST method, disables an active resource – it is not considered any longer to be powered off by the system. |

EUBRA BIGSEA
EUROPE - BRAZIL COLLABORATION OF BIG DATA SCIENTIFIC RESEARCH THROUGH CLOUD-CENTRIC APPLICATIONS

# Automated Oversubscription for Cloud Platforms (CLOUDVAMP)

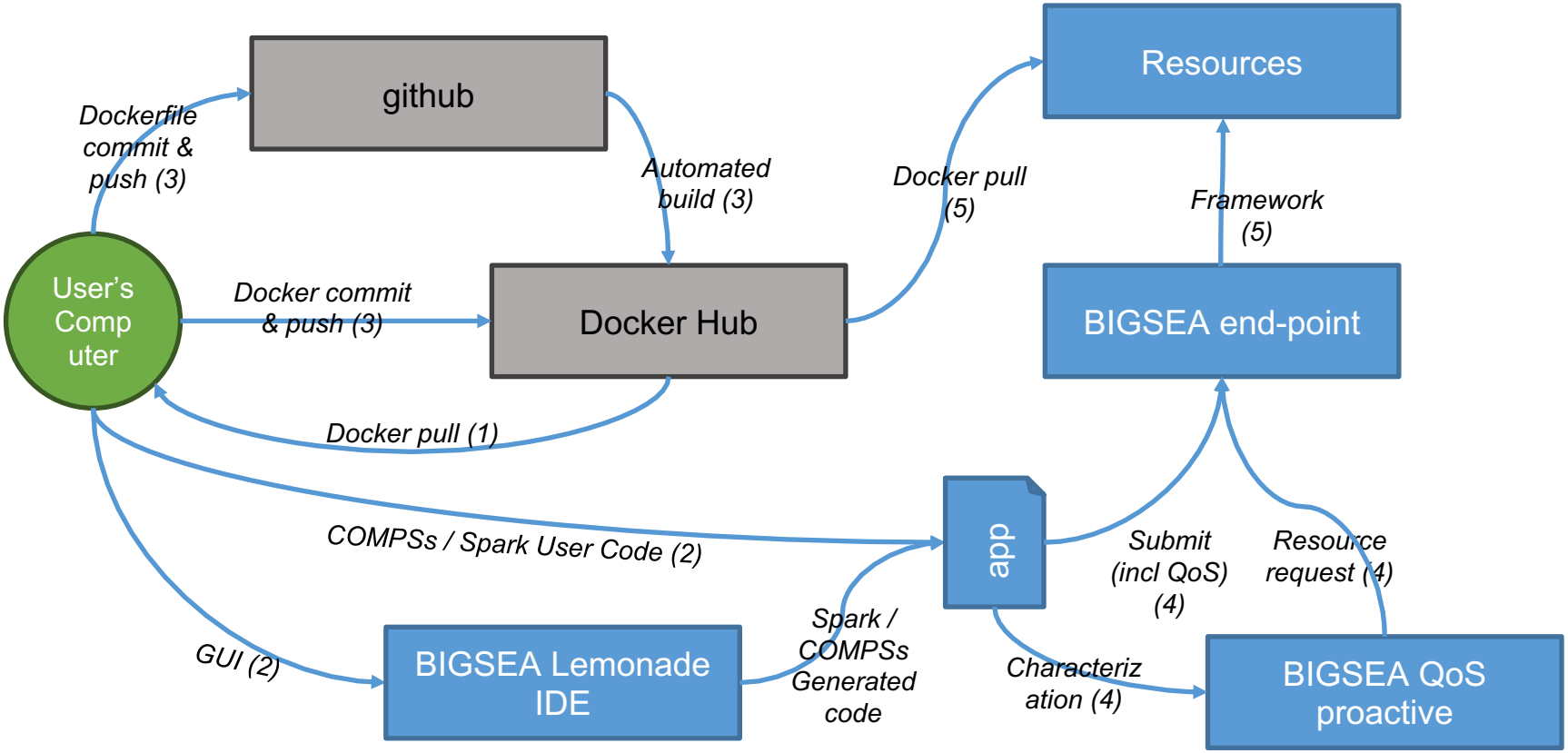
http://www.grycap.upv.es/cloudvamp

- A system to dynamically adjust the physical memory allocated to VMs to the real usage.
- It can be used to allocate more VMs than in a fixed memory allocation model.
- It resizes memory in the VM as need and even migrates VMs to new hosts.

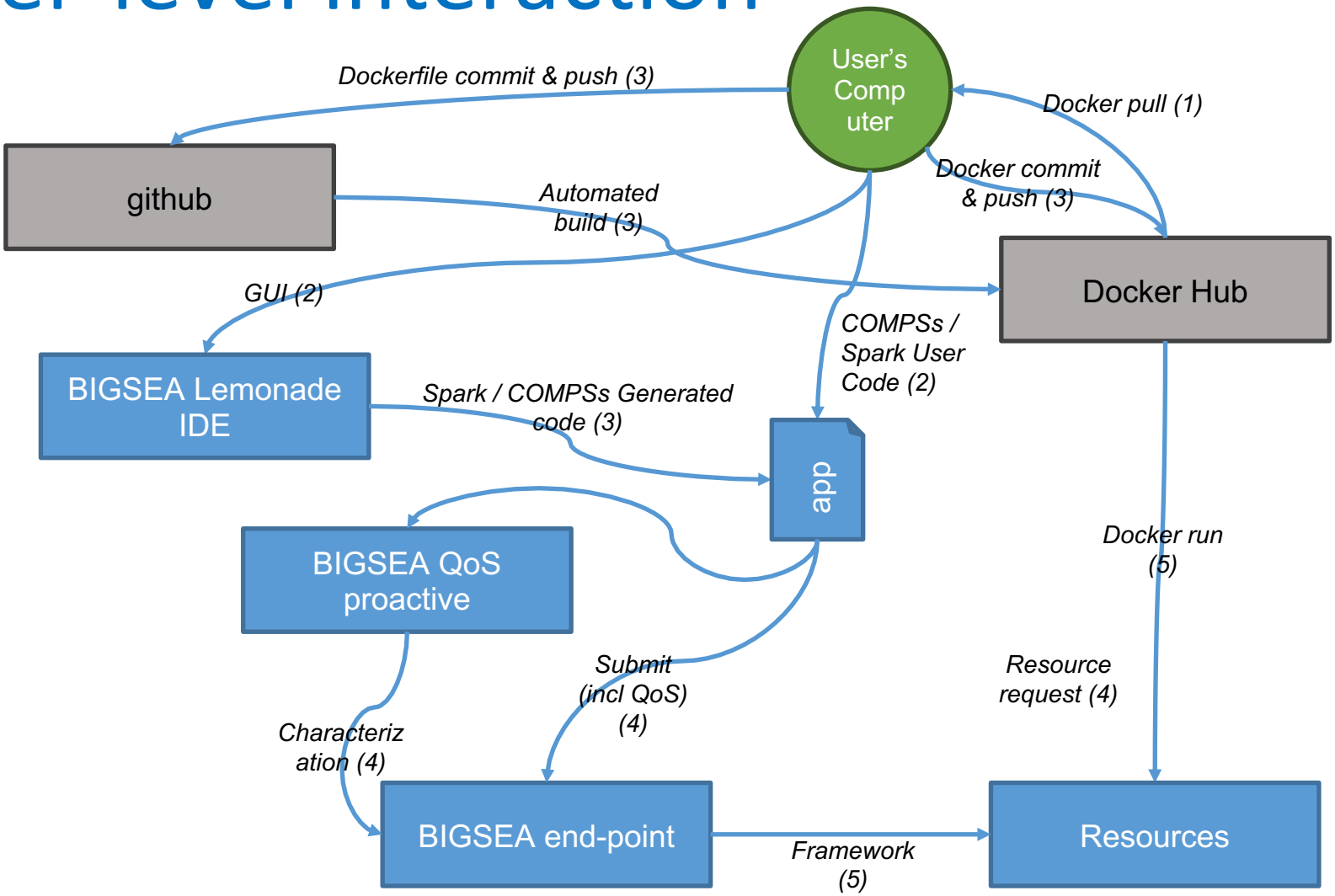


# Logical Components

# User-level interaction



# User-level interaction



# QoS characterization

**Programm Writing**

- Combining Spark and general code in an IDE.
- Using COMPSs for DAG parallelilsation.
- Use of pre-characterized building-blocks and algorithms.

**Customizing Depend.**

- Container images with BIGSEA dependencies (preferred), also supporting direct execution (e.g. Spark).
- Automated build or container push.

**Proactive Resource Character.**

- Run-time policies optimize resource allocation at initial deployment.
- Performance models recalculate QoS metrics (execution time) and trigger optimization module (if necessary).
- API provides system parameter reconfiguration.

**Submission**

- Submission from a single end-point.
- Frameworks interacting with the Mesos cluster.
- Adding QoS constraints (deadlines, execution time,…) and previous proactive resources demand characterization.

# Single Submission end-point

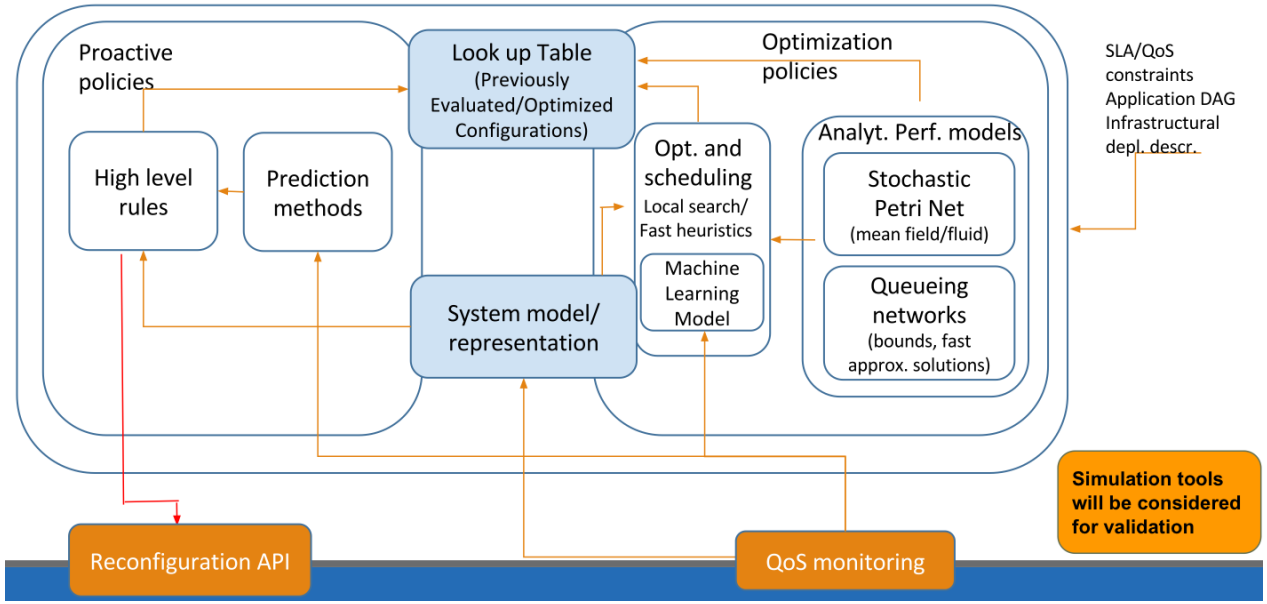
- JSON document with the information on the execution container, framework resources and QoS.

```
{ "type": "CMD",
  "name": "my_job_name",
  "deadline": "2016-06-10T17:22:00Z+2",
  "periodic": "R24P60M",
  "expectedduration": "10M"
  "container" : [
            "type": "DOCKER",
            "image": "eubrabigsea/ubuntu",
            "forcePullImage": true
            "volumes": [
            { "containerPath": "/var/log/",
              "hostPath": "/logs/",
              "mode": "RW"}],
   "portMappings": [
            { "containerPort": 8080,
              "hostPort": 0,
              "protocol": "tcp" }]],
  "environmentVariables": [
  { "name": "value" } ],
  "cpu" : "1.5", "mem" : "512M", "disk" : "1G",
  "command" : "python -m SimpleHTTPServer 8000"
}
```

| | |
|---|---|
| /v1/scheduler | POST method, submits a job request in JSON. |
| /v1/scheduler/jobs | GET method, lists all the jobs in the scheduler (a JSON with all the job). It can accept a JSON with a deadline that defines the date limit. |
| /v1/scheduler/job/jobid | GET method, gets all the information from a specific job; DELETE method, kills the specific job; POST method, reallocates resources. |
| /v1/scheduler/job/jobid/scale | PUT method, changes the resource allocation of a framework. |

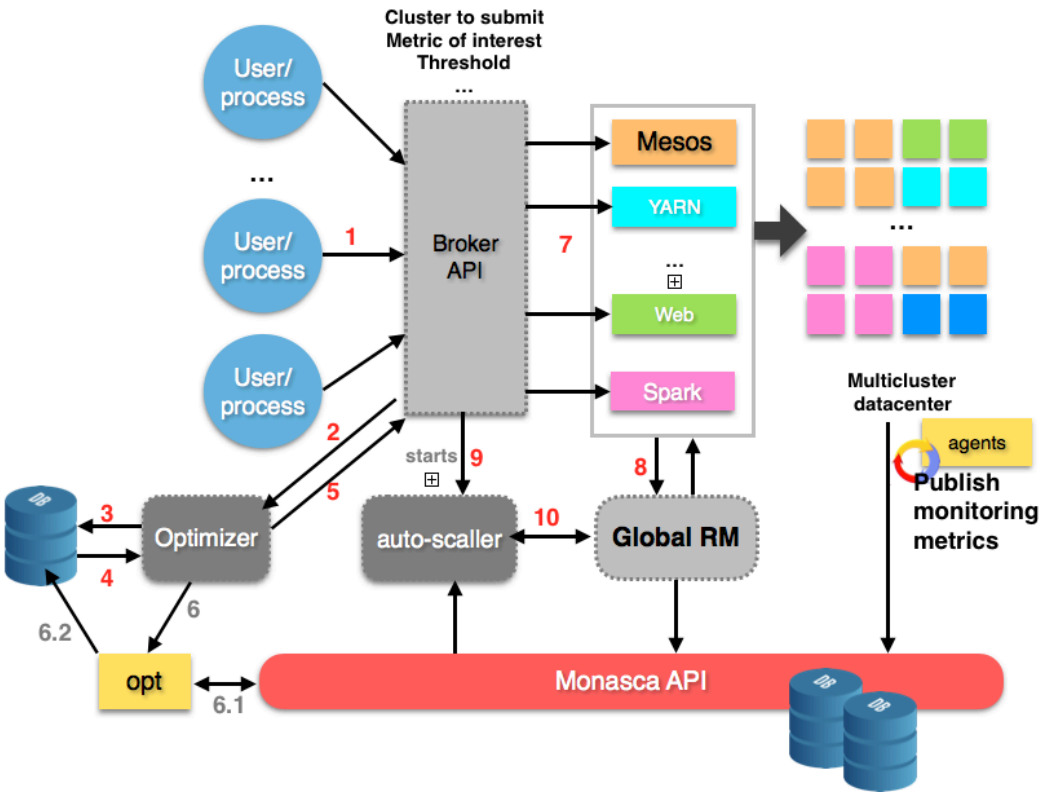
# QoS Proactive Policies

- Policies are based on historical data (look up table) and current system status (QoS monitoring)
- Analytical performance models provide new estimation of quality metrics to the optimization module (at initial deployment and execution times)
- Optimization module recalculates resource allocation, if necessary (new application arrival, changes on the system configuration).



# Monitoring

- Use OpenStack MONASCA as monitoring framework
- Define metrics at the container level.
  - Triggers new resource allocation request if necessary.



# Vertical Elasticity at the level of the framework

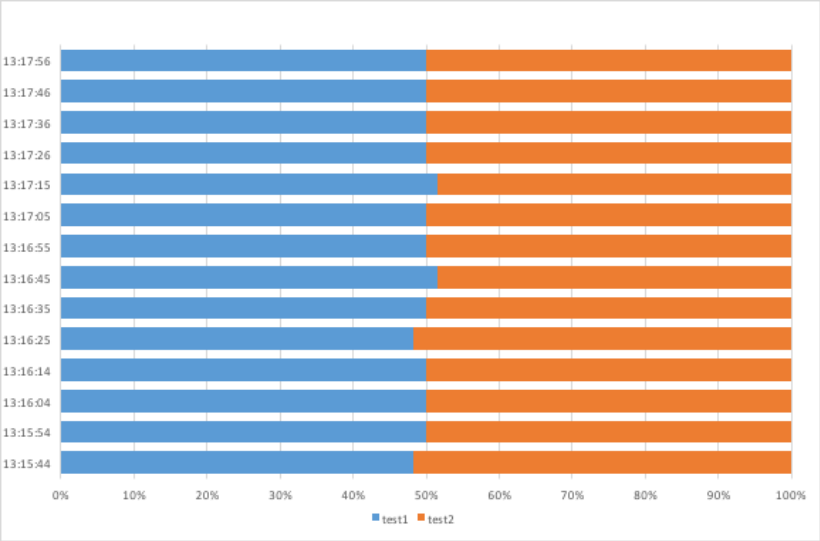
- Directly managed by the submission service
  - It hides the different framework features, interacting with the scheduler and/or Mesos.
  - As example
    - In Chronos it gets the current configuration of the job and resubmits the job with the new resource allocation and the same configuration.
    - In Marathon it updates the json and uses the specific API call

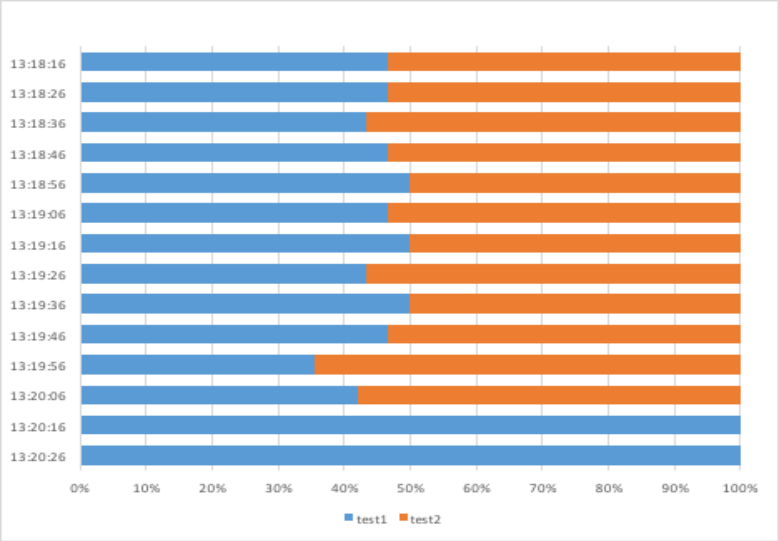
```
$ curl -i -L -H 'Content-Type: application/json' -X PUT -
d@"temporary.json" marathonserver:port/v2/apps/appname?force=true
```

    - In Spark and COMPSs directly interacting with the Mesos Framework.

# Elasticity scenario

- Two multithreaded processing benchmarks running concurrently in the same node
  - Change the allocation from 50%-50% of CPU to 25%-75% as the deadline was approaching.



# Conclusions

- EUBra-BIGSEA aims at providing a set of cloud services to
  - Facilitate the deployment of complex multi-framework Big Data infrastructures.
  - Provide elasticity at the level of the physical resources and frameworks.
  - Include QoS Constraints.
  - Facilitate the use of customized environments.
- Those cloud services are directly used by higher-level programming models hiding the platform particularities.