IsolPharm

computing needs

Lisa Zangrando
INFN-PD

Padova, 05/12/2017

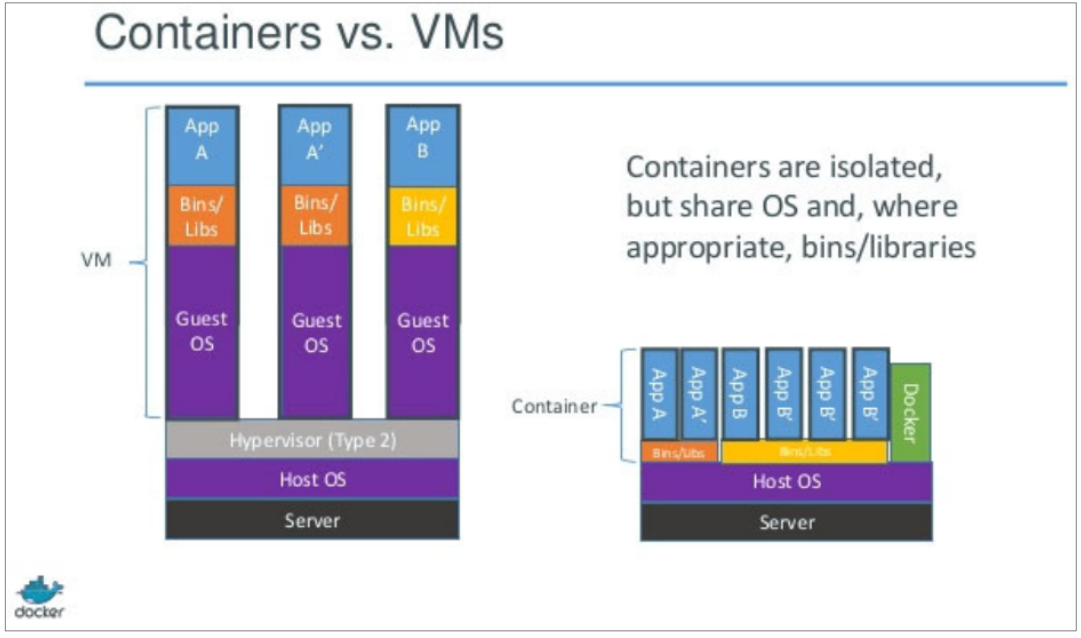
IsolPharm computing needs

- Infrastructure for executing:
  - long-running tasks (e.g. web server, databases)
  - batch jobs (e.g. docker geant4 app)

- IsolPharm relies on the Cloud Area Padovana for the provisioning of the required computing resources
  - created ISOLPHARM_Ag OpenStask project
  - assigned: 50 vcpus, 50GB ram, 700GB storage

A possible computing architecture

- Our computing architecture must be:
  - provide support for containerized applications (e.g. docker)
  - easily deployable (in cloud)
  - easily scalable
  - fault tolerant



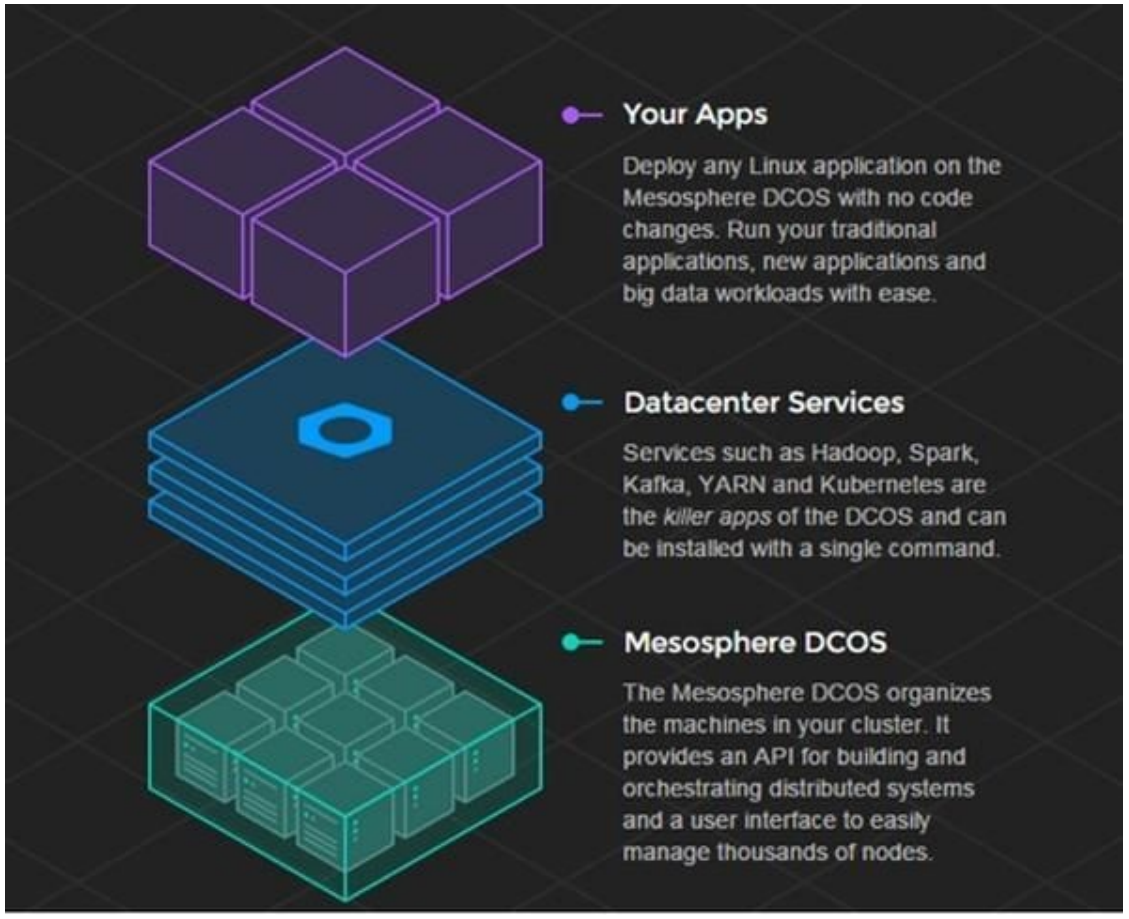
A possible computing architecture

- Several approaches:
  - Kubernetes: open-source system for automating deployment, scaling, and management of containerized applications.
    - https://kubernetes.io/
  - Docker swarm: cluster management and orchestration features embedded in the Docker Engine (open-source)
    - https://docs.docker.com/engine/swarm/
  - Apache Mesos: open-source platform the manage large computing cluster
    - https://mesosphere.com/

Apache Mesos

- Selected by the INDIGO-DataCloud European Project
- Advanced platform able to manage large cluster of resources (cpu, mem), providing isolation and sharing across distributed applications (frameworks)
- From a resource perspective, it’s a cluster manager:
  - pools of (cloud) resources to be centrally managed as a single unit
- From an application perspective, it’s a scheduler:
  - dispatches workloads to consume pooled resources
- Often described as a Data Center Operating System (DCOS)

Apache Mesos architecture



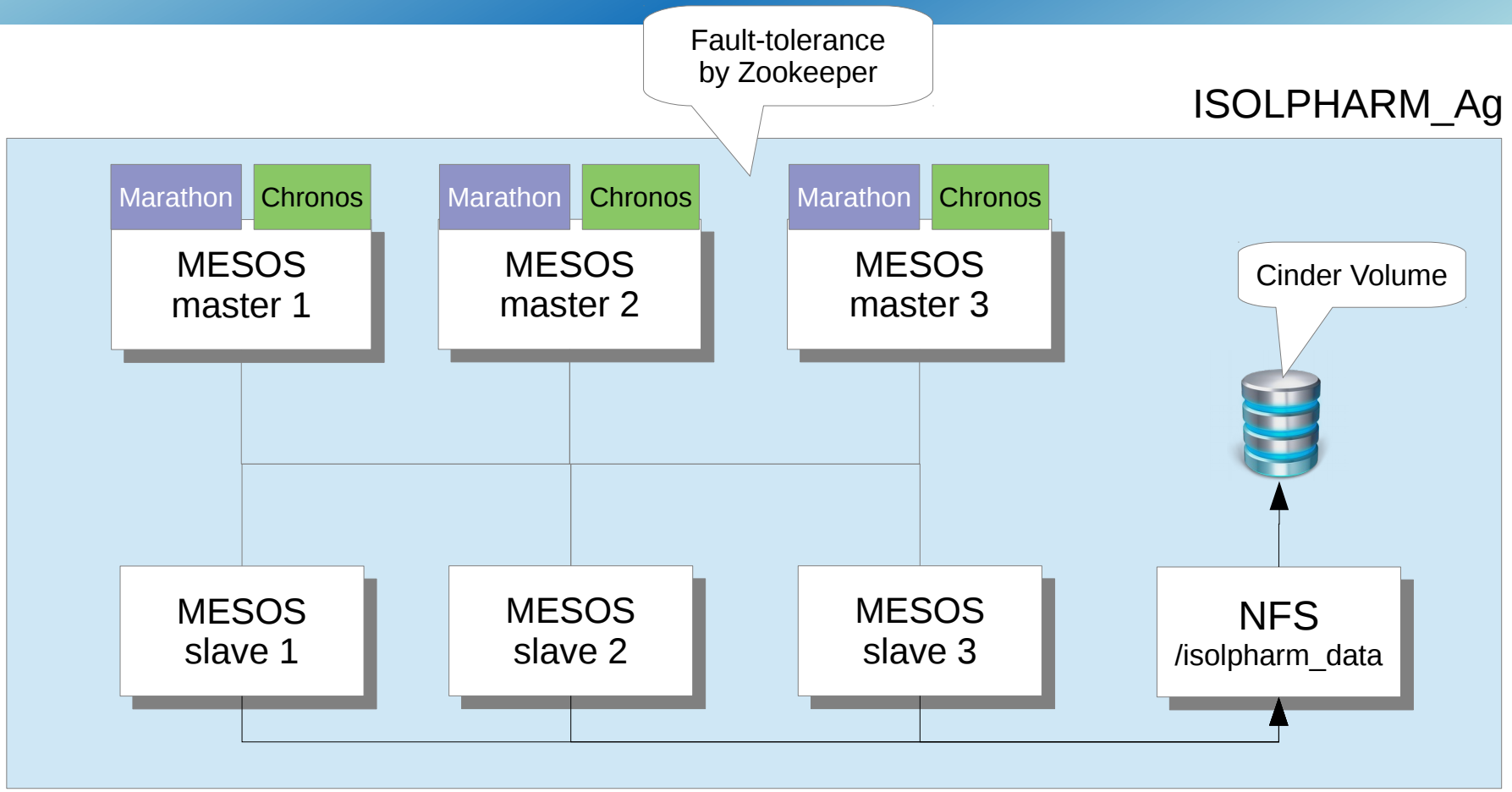
Apache Mesos

- Supports several frameworks for:
  - Big Data Processing
  - Machine Learning
  - Data Storage
  - Long Running Services
    - Aurora, **Marathon**, Singularity, SSSP
  - Batch Scheduling
    - **Chronos**, Cook (like Torque), Elastic-Job-Cloud, GoDocker (like SGE, Torque), Jenkins, JobServer
- http://mesos.apache.org/documentation/latest/frameworks/

Marathon and Chronos overview

- both frameworks deployed on top of a Mesos Cluster
- Marathon: automatically handles hardware or software failures and ensures that long-running services (e.g. web server, databases, etc) are always up and running
- Chronos: distributed and fault-tolerant cron-like system, that can express dependencies between jobs
- Both frameworks provide:
  - support for docker apps
  - REST API
  - simple dashboard

MESOS testbed



Marathon and Chronos tests

- Preliminary tests done on Marathon and Chronos
- Marathon:
  - deployment based on bash script invoking REST API via curl
  - executed simple docker based web-server
- Chronos:
  - deployment based on bash script invoking REST API via curl
  - executed successfully our geant4 app (EF10) in parallel
    - 96 jobs
  - input files (macros) and output files (.tgz) stored into the shared nfs partition /isolpharm_data (Cinder volume)

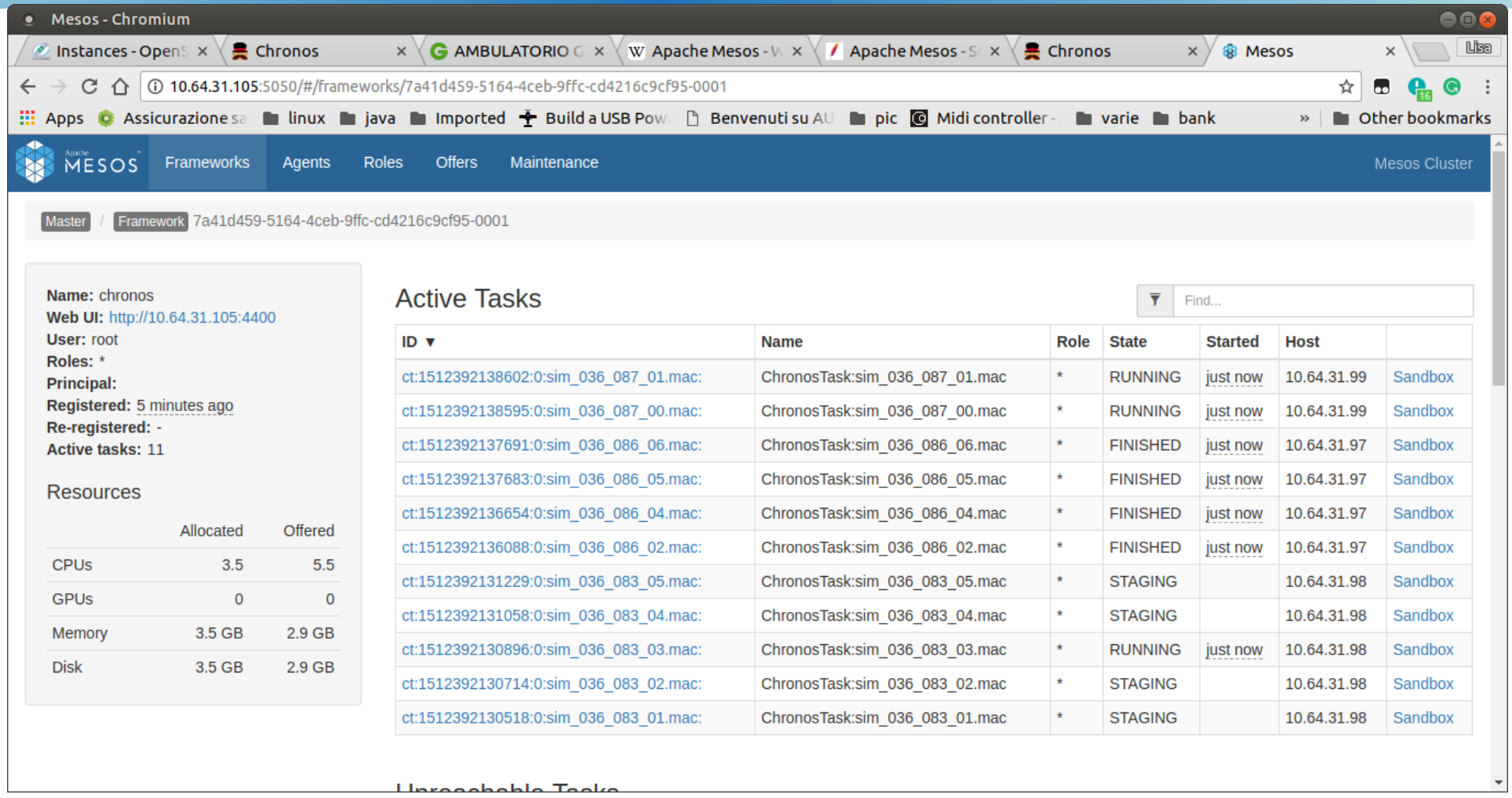
Chronos tests: geant_ef10.json

```
{
  "schedule": "R1/_TIME_/P10M",
  "name": "_MACRO_",
  "container": {
      "type": "DOCKER",
      "image": "ebagli/isolpharm",
      "network": "BRIDGE",
      "volumes": [
          { "containerPath": "/macros/", "hostPath": "/isolpharm_data/EF10/mac/", mode": "RO" },
          { "containerPath": "/output/", "hostPath": "/isolpharm_data/sim_lisa", "mode": "RW" },
          { "containerPath": "/runme.sh", "hostPath": "/isolpharm_data/runme.sh", "mode": "RO" }
      ]
  },
  "cpus": "0.5",
  "mem": "512",
  "command": "/entry-point.sh /runme.sh _MACRO_",
  "shell": true
}
```

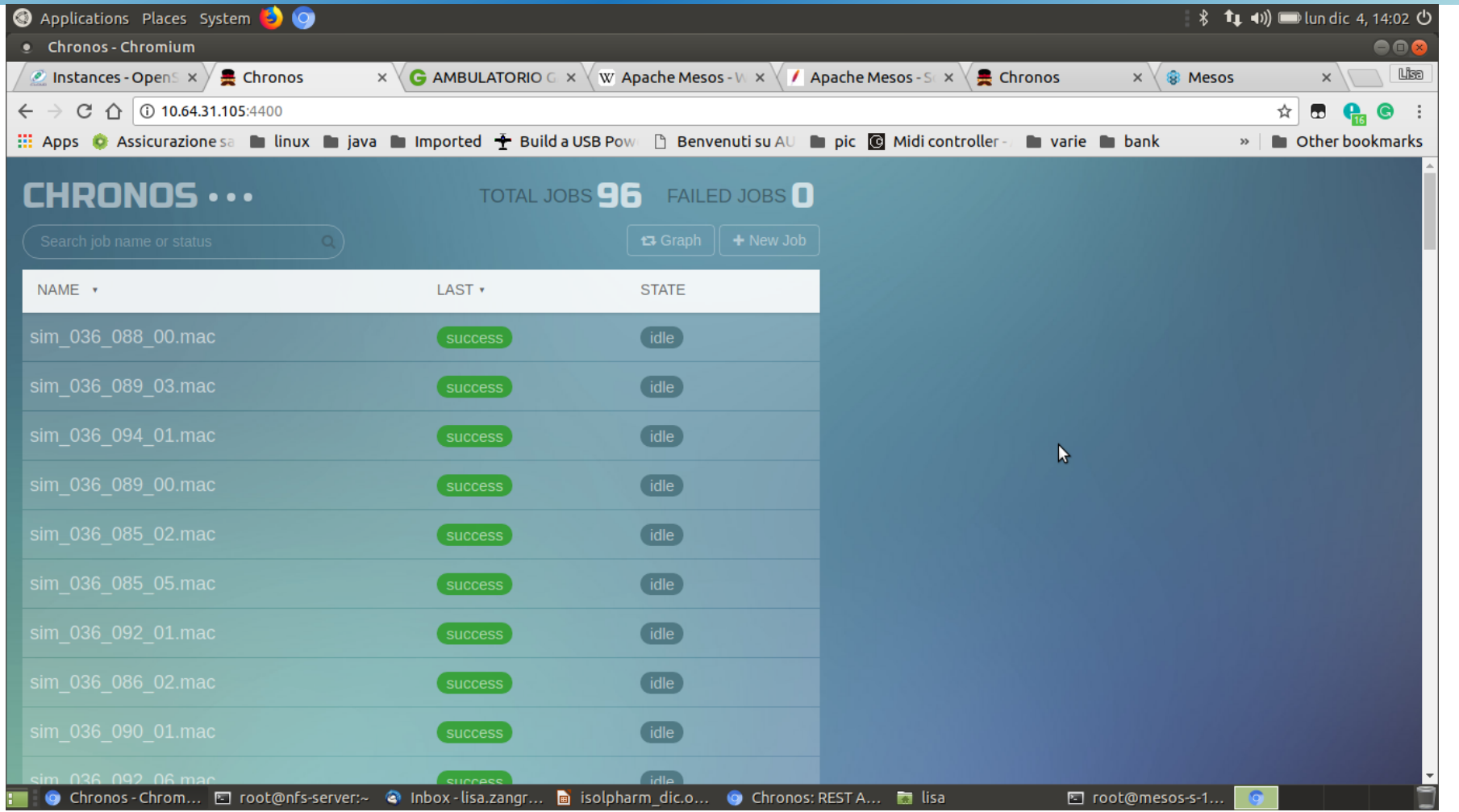
EQUIVALENT TO

```
docker run --rm -v /isolpharm_data/sim2/:/output:rw -v /isolpharm_data/EF10/mac/:/macros/:ro -v
/isolpharm_data/runme.sh/:/runme.sh:ro --name _MACRO_ ebagli/isolpharm /bin/exec /runme.sh _MACRO_
```

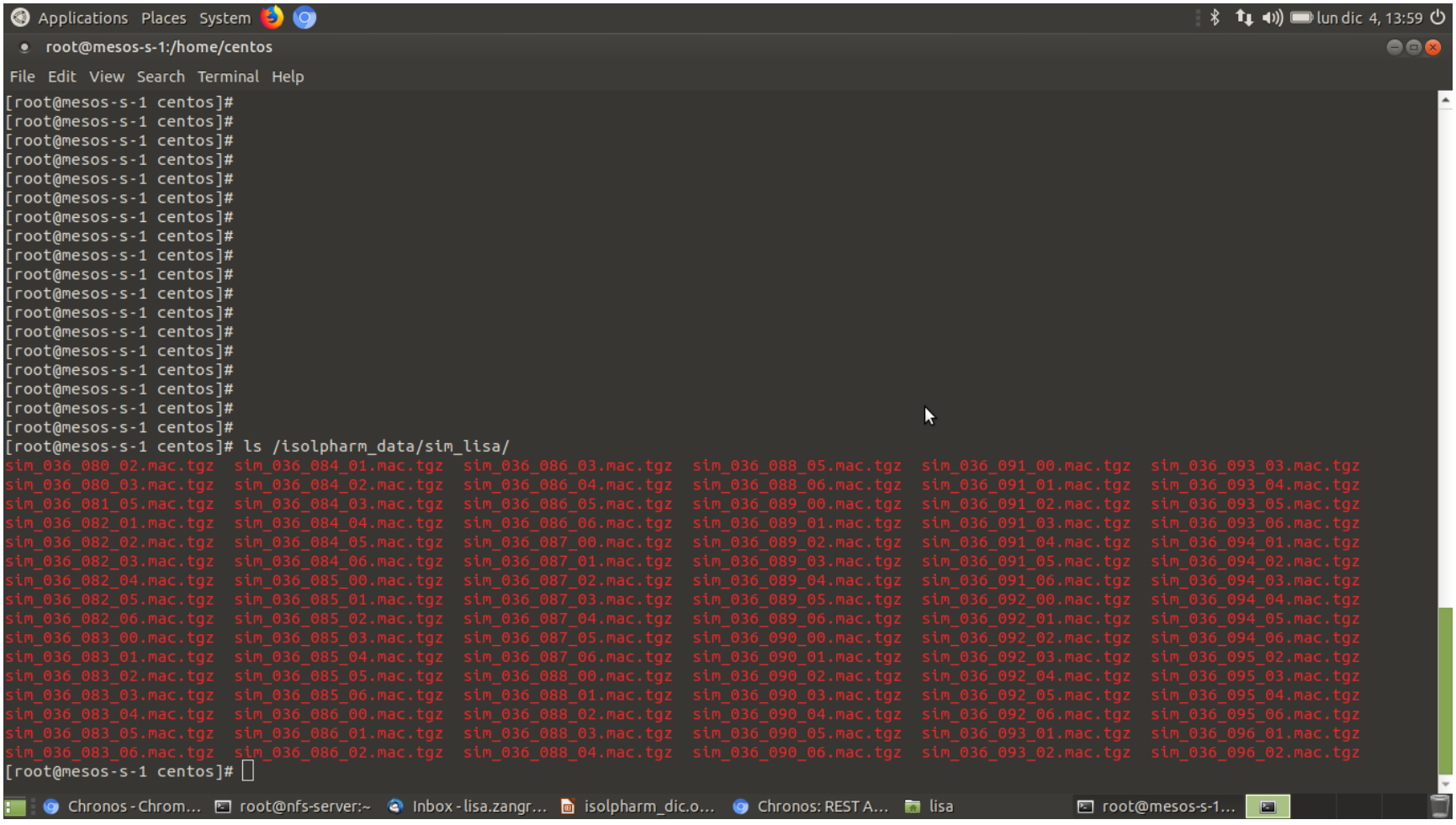
Chronos tests



Chronos tests



Chronos tests



Marathon and Chronos tests

- Preliminary tests done on Marathon and Chronos
- Marathon considerations:
  - it works fine (useful for our web portal)
- Chronos considerations:
  - deployment of docker apps by using REST API needs some tricks
  - designed to perform periodic jobs (although a job can run once)
  - not multi-user (MESOS roles to be investigated)
  - experienced some issues (found workaround):
    - some job attributes produce a wrong docker command
    - failure: java.lang.ArithmeticException: / by zero

Marathon and Chronos tests

- We are still evaluating Chronos
- in the case we need a more sophisticated batch scheduler (multi user, fair-share, etc) we can investigate different solutions
  - Mesos Cook, Mesos GoDocker, Kubernetes, Docker Swarm