Grid-CSIC at IFISC

M. Antònia Tugores Pons

IFISC

GRID CSIC

CSIC Universitat de les Illes Balears

- *Computers*
  - 50 desktops (diskless)
  - 30 laptops
- *Transparent user access*
  - Single account
  - Unified account (login, mail, intranet,…)
  - SMB and NFS
  - LDAP
- *Easy administration*
  - Network booting (Ubuntu 9.04, 10.04)

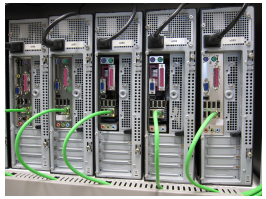
- *Applications*
  - Each user creates its own programs
  - The applications change every week
- *Applications examples*
  - Parametric simulations (several hours – few days)
  - Fluctuations and statistics
  - EDOs integration
  - PDEs integration
  - Social agents evolution in nets
  - Montecarlo simulations

Nuredduna

Mosix Cluster

Designed and assembled at IFISC
Diskless
40 quadcores + 20 dualcores
aprox 200 cores
Gigabit ethernet
max consumption: 11KW

Migration of processes between nodes



Grid Cluster

IBM iDataPlex
62 worker nodes (8 cores, 16GB RAM)
4 storage nodes (8 cores, 16GB RAM)
2 Gigabit ethernet + Fibre (data)
512 cores (496 for the grid)
max consumption: 18KW

Usage
- 25% → EGI
- 25% → NGI
- 25% → Grid-CSIC
- 25% → local users

IFISC environment

Installation started **from scratch**

Basic Software: Scientific Linux 5.4 (64bits) gLite **3.2** SGE 6.2 GPFS 3.3

Structure:

2 **Xen Servers** (hiperthreading: 16 effective cores per server, 16GB RAM)

- Site-BDII (1 core, 2GB RAM)
- Batch System (4 cores, 2GB RAM) → SGE master node
- Computing Element (4 cores, 2GB RAM) → CREAM
- Storage Element (2 cores, 2GB RAM)
- User Interface (2 cores, 2GB RAM) → compilers
- Monitoring (1 core, 2GB RAM) → gLite 3.1, SL4.8
  (gLite 3.2 not integrated in Ibergrid)

4 **Storage Servers** (8 cores, 16GB RAM) → 54TB

62 **Worker Nodes** (8 cores, 16GB RAM) → SGE execution nodes

Grid and IFISC environment

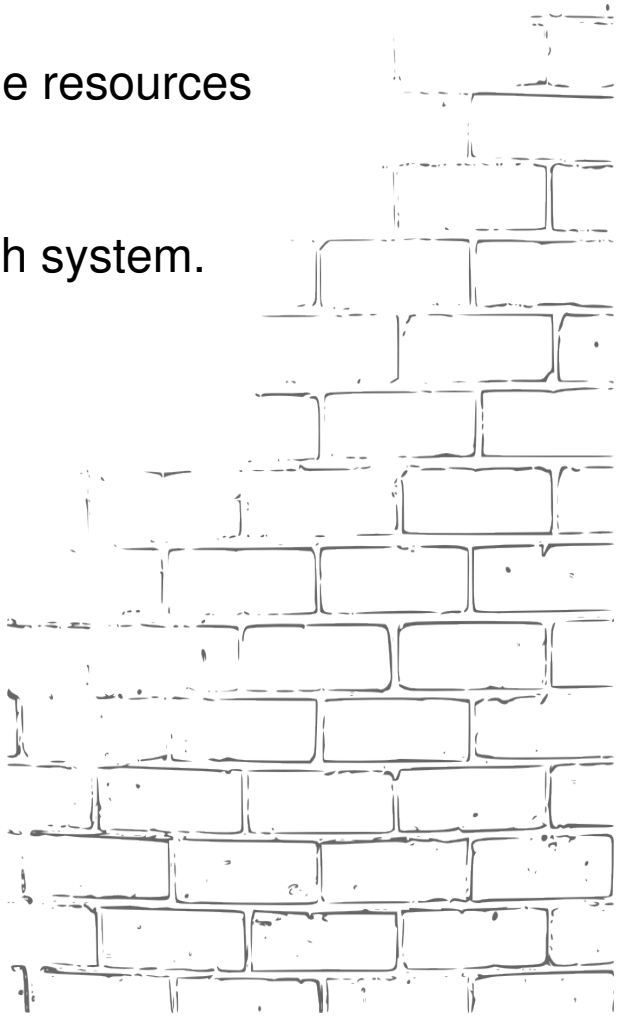
In grid sites local users can access the local part of the resources either using the batch system or the grid.

We decided they **should use grid** instead of the batch system.

As a **trade-off** we made the use of the grid easier.

- users were helped to get the certificates
- UI integrated with the rest of resources
- unified home directory
- usual compilers and libraries in UI
- IFISC VO
- creation of scripts simplifying job submission

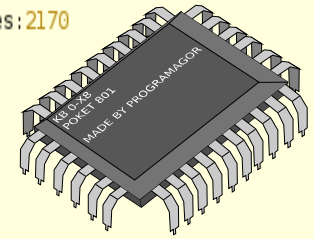
  gridLoad. status, cancelJobs, clearSE, …
- proactive support



Usage: runGrid -a userapplication -p “arg1 arg2” -i input1,dir/*,*.ini

```
# update environment
updateEnvironmentVars(parameters)
createProxy(parameters)

#upload files
remoteInputFile = uploadInputFiles(parameters)

# create files
generateScript(remoteInputFile)
jdlFile = generateJDL(parameters)

# submit
submitJob(jdlFile)

while not finished and not exceededTime:
  time.sleep(sleepTime)
  finished,status = hasFinished()

# check finalization and retrieve results/information if necessary
if exceededTime and not finished:
  cancelJob()
else:
  if status.find(STATUS_FINISHED) != -1:
    retrieveResults()
    if status.find(STATUS_FINISHED_EXITCODENONZERO) != -1:
      FINISHERROR = True
  elif status.find(STATUS_ABORTED) != -1:
    retrieveInfo()
  else:
    retrieveInfo()
    FINISHERROR = True

# clear SE
if not ERROR:
  removeInputOutputFiles(parameters, remoteInputFile)
  removeRemoteDir(parameters)

if not ERROR and not FINISHERROR:
  cleanUserInterface()
else:
  moveDataToOutputs()
```

```
export export LCG_CATALOG_TYPE=lfc
export LCG_GFAL_INFOSYS=gridii01.ifca.es:2170
export LFC_HOST=lfc02.ific.uv.es

# get current dir
currentDir=`pwd`

# download files
echo downloading file
lcg-cp --vo $vo lfn:$remoteInputFile file://$currentDir/inputs.tgz

# untar inputs
echo untar input files
tar zxvf inputs.tgz -p
rm inputs.tgz

# run application redirecting stdout and stderr
echo run application
./$application $parameters  > std.out 2> std.err

rc=$?

# create outputs tgz
echo create outputs file
tar zcvf $outputFile * --exclude-from $exclusions

# upload files
echo upload outputs
lcg-cr --vo $vo -l lfn:$remoteOutputFile -d nureddunase.uib-csic.es
file://$currentDir/$outputFile

exit $rc
```

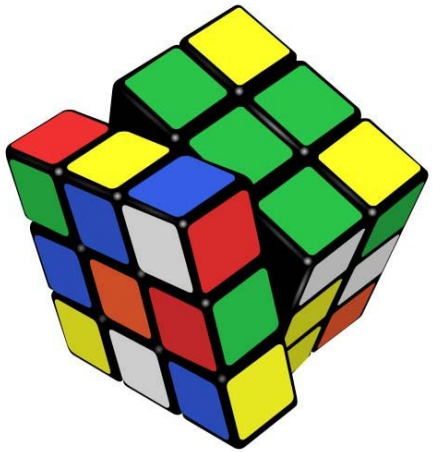


Challenges

gLite installation is far from trivial (packages, incompatibilities, ports, ...)

gLite commands are too complex to be used by average scientists

- Different parameters for similar commands
- Too many parameters
- Examples: lcg-cr, lcg-cp, lfc-ls



Communication between CE and WMS

- A simple job (that lasts some seconds of CPU time) will last about 10min when submitted through the WMS

Proxies

- The default time is 12h (most of the people run longer programs)
- Long term proxies:
  - users depend on another service
  - independent UI proxy renewal when using long term proxies

Requirements JDL attribute

Powerful, but too complex

```
other.GlueCEUniqueID == "nureddunace.uib-csic.es:8443/cream-sge-esngi"
other.GlueCEInfoTotalCPUs > 1
```

Thank you!