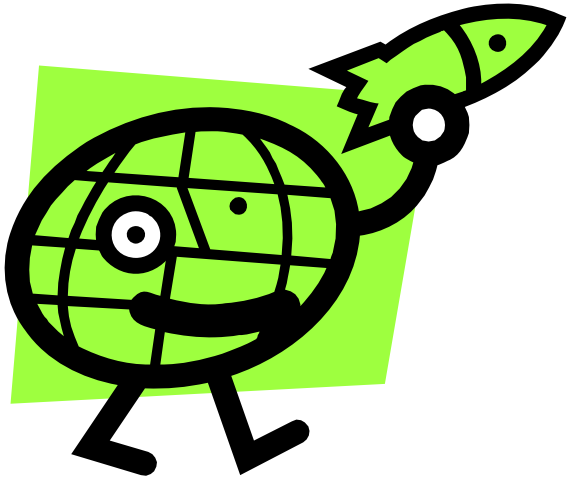
GILDA Praticals

GILDA Tutors
INFN Catania

EGEE Tutorial

Rome 02-04 November 2005



Workload Management System

Valeria Ardizzone

Giuseppe La Rocca

Workload Management System

- In the glite middleware a user can submit and cancel jobs, query their status, and retrieve their output. These tasks go under the name of ***Workload Management***.

- There are two different User Interfaces to accomplish these tasks. One is the Command Line Interface and the other is the Graphical User Interface.

Command Line Interface

- **Job Submission**
  - Perform the job submission to the Grid.

```
$ glite-job-submit  [options] <jdl_file>
```

  - where <jdl file> is a file containing the job description, usually with extension .jdl.

--vo <vo name> : perform submission with a different VO than the UI default one.

--output, -o <output file> save jobId on a file.

--resource, -r <resource value> specify the resource for execution.

--nomsgi neither message nor errors on the stdout will be displayed.

Command Line Interface (cont.)

If the submission is successful, the output is similar to:

```
glite-job-submit test.jdl

==========glite-job-submit Success ==========
The job has been successfully submitted to the Network Server.
Use glite-job-status command to check job current status.
Your job identifier (edg_jobId) is:
- https://lxshare0234.cern.ch:9000/rIBubkFFKhnSQ6CjiLUY8Q
==========
```

In case of failure, an error message will be displayed instead, and an exit status different form zero will be retured.

Command Line Interface (cont.)

If the command returns the following error message:

****** Error: API_NATIVE_ERROR ******
Error while calling the "NSClient::multi" native api
AuthenticationException: Failed to establish security context...
****** Error: UI_NO_NS_CONTACT ******
Unable to contact any Network Server

it means that there are authentication problems between the UI and the *Network Server* (check your proxy or have the site administrator check the certificate of the server).

Command Line Interface (cont.)

It is possible to see which CEs are eligible to run a job specified by a given JDL file using the command

glite-job-list-match test.jdl

```
Connecting to host lxshare0380.cern.ch, port 7772
Selected Virtual Organisation name (from UI conf file): dteam
**********************************************************************
COMPUTING ELEMENT IDs LIST
The following CE(s) matching your job requirements have been found:
adc0015.cern.ch:2119/jobmanager-lcgpbs-infinite
adc0015.cern.ch:2119/jobmanager-lcgpbs-long
adc0015.cern.ch:2119/jobmanager-lcgpbs-short
**********************************************************************
```

Command Line Interface (cont.)

After a job is submitted, it is possible to see its status using the glite-job-status command.

glite-job-status https://lxshare0234.cern.ch:9000/X-ehTxfdlXxSoIdVLS0L0w

```
***********************************************************
BOOKKEEPING INFORMATION:
Printing status info for the Job:
https://lxshare0234.cern.ch:9000/X-ehTxfdlXxSoIdVLS0L0w
Current Status: Ready
Status Reason: unavailable
Destination: lxshare0277.cern.ch:2119/jobmanager-pbs-infinite
reached on: Fri Aug 1 12:21:35 2003
***********************************************************
```

Command Line Interface (cont.)

The option -i <file path> can be used to specify a file with a list of job identifiers (saved previously with the -o option of glite-job-submit).

```
glite-job-status -i jobs.list
--------------------------------------------------------------------------------
1 : https://lxshare0234.cern.ch:9000/UPBqN2s2ycxt1TnuU3kzEw
2 : https://lxshare0234.cern.ch:9000/8S6IwPW33AhyxhkSv8Nt9A
3 : https://lxshare0234.cern.ch:9000/E9R0Yl4J7qgsq7FYTnhmsA
4 : https://lxshare0234.cern.ch:9000/Tt80pBn17AFPJyUSN9Qb7Q
a : all
q : quit
--------------------------------------------------------------------------------
Choose one or more edg_jobId(s) in the list - [1-4]all:
```

If the - -all option is used instead, the status of all the jobs owned by the user submitting the command is retrieved.

Command Line Interface (cont.)

The --status <state> (-s) option makes the command retrieve only the jobs that are in the specified state, and the --exclude <state> (-e) option makes it retrieve jobs that are not in the specified state.

This two lasts options are mutually exclusive, although they can be used with --from and --to.

Example: All jobs of the user that are in the state DONE or RUNNING are retrieved.

```
glite-job-status --all -s Done -s Running
```

Example: All jobs that were submitted before the 17:35 of the current day, and that were not in the Cleared state are retrieved.

```
glite-job-status --all -e Cleared --to 17:00
```

Command Line Interface (cont.)

A job can be canceled before it ends using the command glite-job-cancel.

glite-job-cancel https://lxshare0234.cern.ch:9000/dAE162is6EStca0VqhVkog

Are you sure you want to remove specified job(s)? [y/n]n :y
========== glite-job-cancel Success==========
The cancellation request has been successfully submitted for the following job(s)
- https://lxshare0234.cern.ch:9000/dAE162is6EStca0VqhVkog
==========

Command Line Interface (cont.)

After the job has finished (it reaches the DONE status), its output can be copied to the UI

```
glite-job-output https://lxshare0234.cern.ch:9000/snPegp1YMJcnS22yF5pFlg

Retrieving files from host lxshare0234.cern.ch
************************************************************************
JOB GET OUTPUT OUTCOME
Output sandbox files for the job:
- https://lxshare0234.cern.ch:9000/snPegp1YMJcnS22yF5pFlg
have been successfully retrieved and stored in the directory:
/tmp/jobOutput/snPegp1YMJcnS22yF5pFlg
************************************************************************
```

By default, the output is stored under /tmp, but it is possible to specify in which directory to save the output using the - -dir <path name> option.

Exercises

Exercises

Create a bash script which displays hostname and current date

Save the script as yourscript.sh

Simple job submission

cp hostname.jdl exercise1.jdl

Modify exercise1.jdl file

Instead of running hostname command, run a bash script you have just created (yourscript.sh).

Submit the job, check its status and when done retrieve the output

JDL – Requirements

```
#Insert a requirement to parse only the short queues.
Requirements = (other.GlueCEPolicyMaxWallClockTime >
   720);

#Insert a requirement to parse only the long queues.
Requirements = (other.GlueCEPolicyMaxWallClockTime >
   1440);

#Insert a requirement to parse only the infinite
   queues.
Requirements = (other.GlueCEPolicyMaxWallClockTime >
   2880);

#Insert a requirement to stear the execution on a
   particular CE Queue.
Requirements = other.GlueCEUniqueID ==
   "grid010.ct.infn.it:2119/jobmanager-lcgpbs-long";
```

Exercises

- **Simple job using Requirements**
  - Modify exercise1.jdl file so that user with a even workstation number will submit their job on a “long” queue, and the other to an “infinite” one
  - Verify the list of CE suitable for this job execution
  - Submit the job, check its status and retrieve the output

JDL with Requirements

```
 Type = "Job";
JobType = "Normal";
Executable = "/bin/sh";
Arguments = "start_povray_cubo.sh";
StdOutput = "povray_cubo.out";
StdError = "povray_cubo.err";
InputSandbox = {"start_povray_cubo.sh","cubo.pov"};
OutputSandbox =
   {"povray_cubo.out","povray_cubo.err","cubo.png"};

Requirements = Member("POVRAY-
   3.5",other.GlueHostApplicationSoftwareRunTimeEnvir
   onment);
```

start_povray_cubo.sh

```
#!/bin/bash
mv cubo.pov OBJECT.POV #rename input file
/usr/bin/povray /usr/share/povray-3.5/ini/res800.ini  #run
    povray
mv OBJECT.png cubo.png #rename output file
```

Exercises

- Modify povray_cubo.jdl, specifying the resource for execution into the jdl file
- Check job status and when done retrieve the output
- Display .png file obtained as output, using ImageMagick

Exercises

Run an ls command on the selected resource.

```
[
  Executable = "ls.sh";
  Arguments = "-alt";
  StdError = "stderr.log";
  StdOutput = "stdout.log";
  InputSandbox = "ls.sh";
  OutputSandbox = {"stderr.log", "stdout.log"};
]
```

ls.sh

```
#!/bin/sh
/bin/ls
```

mpi.jdl

```
Type = "Job";
JobType = "MPICH";
Executable = "MPItest.sh";
Arguments = "cpi 2";
NodeNumber = 2;
StdOutput = "test.out";
StdError = "test.err";
InputSandbox = {"MPItest.sh","cpi"};
OutputSandbox = {"test.err","test.out","executable.out"};
Requirements = other.GlueCEInfoLRMSType == "PBS" ||
    other.GlueCEInfoLRMSType == "LSF";
```

MPItest.sh

```
#!/bin/sh
#

# this parameter is the binary to be executed
EXE=$1
# this parameter is the number of CPU's to be reserved for parallel
    execution
CPU_NEEDED=$2

# prints the name of the master node
echo "Running on: $HOSTNAME"

if [ -f "$PWD/.BrokerInfo" ] ; then
      TEST_LSF=`edg-brokerinfo getCE | cut -d/ -f2 | grep lsf`
else
      TEST_LSF=`ps -ef | grep sbatchd | grep -v grep`
fi
```

MPItest.sh (cont.)

```
if [ "x$TEST_LSF" = "x" ] ; then
      # prints the name of the file containing the nodes allocated for
    parallel execution
      echo "PBS Nodefile: $PBS_NODEFILE"
      # print the names of the nodes allocated for parallel execution
      cat $PBS_NODEFILE
      HOST_NODEFILE=$PBS_NODEFILE
else
      # print the names of the nodes allocated for parallel execution
      echo "LSF Hosts: $LSB_HOSTS"
      # loops over the nodes allocated for parallel execution
      HOST_NODEFILE=`pwd`/lsf_nodefile.$$

      for host in ${LSB_HOSTS}  do
            host=`host $host | awk '{ print $1 } '`
            echo $host >> ${HOST_NODEFILE}
      done
fi
```

MPItest.sh (cont.)

```
# prints the working directory on the master node
echo "Current dir: $PWD"

for i in `cat $HOST_NODEFILE` ; do
      echo "Mirroring via SSH to $i"
      # creates the working directories on all the nodes allocated for
    parallel execution
      ssh $i mkdir -p `pwd`
      # copies the needed files on all the nodes allocated for parallel
    execution
      /usr/bin/scp -rp ./* $i:`pwd`
      # checks that all files are present on all the nodes allocated for
    parallel execution
      echo `pwd`
      ssh $i ls `pwd`
      # sets the permissions of the files
      ssh $i chmod 755 `pwd`/$EXE
      ssh $i ls -alR `pwd`
done
```

MPItest.sh (cont.)

```
# execute the parallel job with mpirun
echo "************************************"
echo "Executing $EXE"
chmod 755 $EXE
mpirun -np $CPU_NEEDED -machinefile $HOST_NODEFILE
    `pwd`/$EXE > executable.out
echo "************************************"
```

.BrokerInfo file

- It is a mechanism by which a job can access at some information about itself…at execution time!
- The Resource Broker creates and attaches this file to the job when it is ready to be transfered to the resource that best matches the request.
- Two ways for parsing elements from .BrokerInfo file:

1)Directly from the Worker Node at execution time;

2)From User Interface, but only if you have inserted the name of ".BrokerInfo" file in the JDL's OutputSandbox, and you have just retrieved job output, once that job has been Done;

edg/glite-brokerinfo [options] function param

Example of .BrokerInfo file

```
[       ComputingElement =
         [
            CloseStorageElements =
              {
                [
                        GlueSAStateAvailableSpace =
    14029724;
                        GlueCESEBindCEAccesspoint
    = "/flatfiles/SE00";
                        mount =
    GlueCESEBindCEAccessPoint;
                        name = "grid003.cecalc.ula.ve";
                        freespace =
    GlueSAStateAvailableSpace
                    ]
              };
            name =
    "grid006.cecalc.ula.ve:2119/jobmanager-
    lcgpbs-infinite"
         ];
       InputFNs =
         {
         };
       StorageElements =
         {
         };
       VirtualOrganisation = "gilda"    ]
```

edg-brokerinfo getCE

edg-brokerinfo getDataAccessProtocol

edg-brokerinfo getInputData

edg-brokerinfo getSEs

edg-brokerinfo getCloseSEs

edg-brokerinfo getSEMountPoint <SE>

edg-brokerinfo getSEFreeSpace <SE>

edg-brokerinfo getSEProtocols <SE>

edg-brokerinfo getSEPort <SE> <Protocol>

edg-brokerinfo getVirtualOrganization

edg-brokerinfo getAccessCost

Exercises

Exercise 1

- Create a file called **startScriptBrokerInfo.sh** with this content:

```
#!/bin/sh
MY_NAME="Your name"
WORKER_NODE_NAME=`hostname`

echo "Hello $MY_NAME, from $WORKER_NODE_NAME"
ls -a
echo "This job is running on this CE: "
/opt/edg/bin/edg-brokerinfo getCE
```

Exercises

Exercise 2

- Create a file called **scriptBrokerInfo.jdl** with this content:

```
[
   Executable = "startScriptBrokerInfo.sh";
   StdOutput = "std.out";
   StdError = "std.err";
   VirtualOrganisation = "gilda";
   InputSandbox = {"startScriptBrokerInfo.sh"};
   OutputSandbox = {"std.out","std.err",".BrokerInfo"};
   RetryCount = 7;
]
```

Exercises

1. Replace your name in the file script startScriptBrokerInfo.sh;
2. Submit / Query the status / Retrieve Output the JDL file scriptBrokerInfo.jdl;
3. In JobOutput folder, go into directory of the job that you have just retrieved and inspect the .BrokerInfo file.
4. Take practice with the edg-brokerinfo command and its functions.

Exercises

This exercise allows user to submit a C program.

Modify c_sample.c file as follow:

```
#include <stdio.h>
int main(int argc, char **argv)
{
    printf("\n\n\n");
    printf("Hello !\n");
    printf("Welcome to  EGEE Tutorial, Rome 02th-04th
    Nov - 2005 \n\n\n");
    exit(0);
}
```

Exercises (cont.)

Compile your script using make.

Submit the **c_sample.jdl** job to the grid using the *glite-job-submit c_sample.jdl* command.

Inspect the status and retrieve its output when the job is finished.

Exercises

Modify c_sample.c file as follow:

```
#include <stdio.h>
int main(int argc, char **argv)
{
  char *name = argv[1];
  printf("\n\n\n");
  printf("Hello !\n");
  printf("Welcome to EGEE Tutorial, Rome 02th-04th Nov
     - 2005 \n\n\n");
  exit(0);
}
```

Exercises (cont.)

Compile your script with make.

Modify the start_c_sample.sh script as follow:

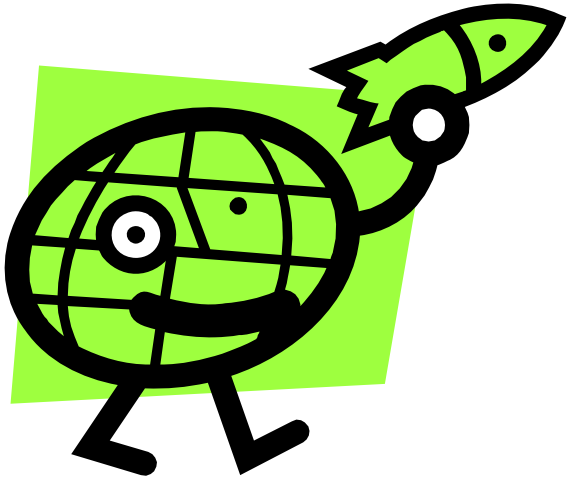
```
#!/bin/sh
chmod 777 c_sample
./c_sample $1
```

Modify c_sample.jdl's Arguments as follow:

Arguments = "start_c_sample.sh <Your Name>";

Submit, inspect the status and retrieve its output when the job is finished.

Questions…



DataGrid Accounting System

Giuseppe La Rocca

Exercise 11

View user Credits

$ dgas-check-balance

User: Giuseppe La Rocca

E-mail: giuseppe.larocca@ct.infn.it

Subject: /C=IT/O=GILDA/OU=Personal Certificate/L=INFN Catania/CN=Giuseppe La Rocca/Email=giuseppe.larocca@ct.infn.it

Assigned credits (0=infinite): 0

Booked credits: 0

Used credits: 451

Used wall clock time (sec): 1187

Used CPU time (sec): 264

Accounted jobs: 22

Exercise 12

View CE Price

Usage: dgas-check-ce-price <CE name>:2119/jobmanager-lcgpbs-<queue>

Example: dgas-check-ce-price grid010.ct.infn.it:2119/jobmanager-lcgpbs-short

Price Authority queried at: Thu Oct 20 18:43:39 CEST 2005

Computing Element: grid010.ct.infn.it:2119/jobmanager-lcgpbs-short

Price (credits for 100 CPU secs): 170

Questions…