gLite training at Sinaia '06

Victor Penso
Kilian Schwarz
GSI Darmstadt
Germany

Preliminaries
Login information

login to the Sinaia PC as user t01-t30

with password "abc123"

via ssh login to one of the 4 gLite UI provided at FZK.

- *iwrgks-4-5.fzk.de*
- *iwrgks-5-5.fzk.de*
- *iwrgks-6-5.fzk.de*
- *iwrgks-7-5.fzk.de*

user and PW to be used by each group can be found in the handout

Preliminaries - Certificates

- **Each user has 2 important files:**
  - `usercert.pem`
    - Could be read by everyone - file permission: 644
  - `userkey.pem`
    - Only the owner is allowed to read this - file permission: 600
- **They are located in `$HOME/.globus/`**
- **The certificate is needed for all grid activities like submitting jobs, copying files, etc.**
- **Browser certificates allow authentication on different websites, e.g. grid user support sites.**

Authentication

** To submit grid jobs, copy files to grid sites and all other activities in the grid requiring user authentication you need a 'grid proxy':*

- Make a new grid-proxy:

- voms-proxy-init –voms dgtest [-valid hh:mm]

- enter your Grid pass phrase (see handout)

- For long jobs, maybe you need to change the proxy lifetime using e.g. the '-valid 24:00' option for a lifetime of one day.

- Check the status of your proxy

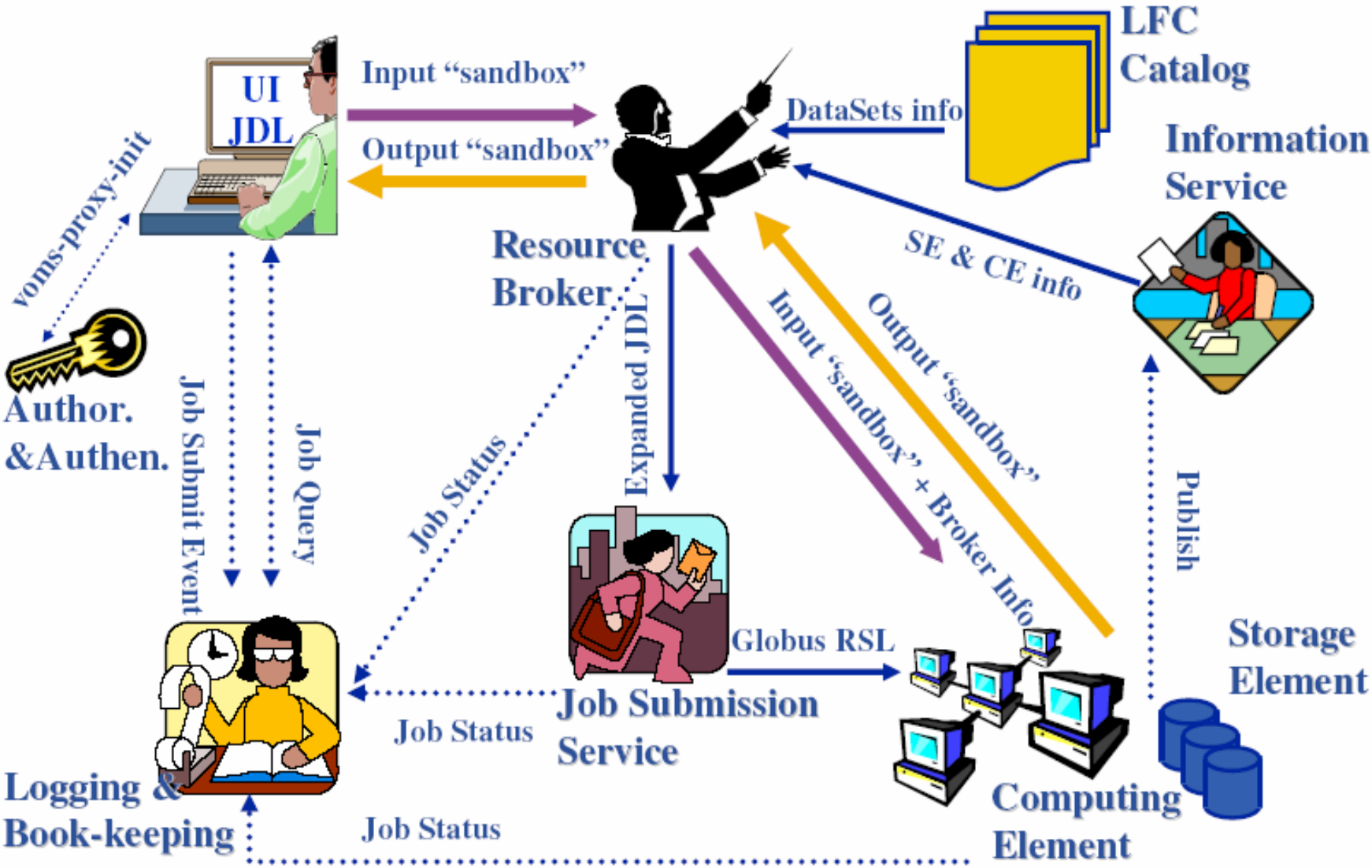
- voms-proxy-info

- Delete your proxy:

- voms-proxy-destroy

Remark: in this course we use the VO "dgtest" of the German D-Grid project

gLite - Job Workflow



- **Job Description Language (JDL)**
  → "Hello World" & more
  - The JDL is used to specify special needs of your job like necessary input-files/variables, generated output- and logging-information or requirements concerning the computing resources on remote sites.

create the file "simple.jdl":

```
Executable = "simple.sh";
Arguments = "Hello world!";

StdOutput = "std.out";
StdError = "std.err";

InputSandbox = {"simple.sh"};
OutputSandbox = {"std.out", "std.err" };

VirtualOrganisation = "dgtest";
RetryCount = 2;
```

The executable

**** The executable:***

- in our example, the executable is "simple.sh"

- it is a shell script gathering useful information about the worker node

- please create it yourself: it contains, e.g.:

```
#!/bin/bash
echo "I am \"`whoami`\" on host \"`hostname`\"."
echo `pwd`
echo `cat /proc/cpuinfo`
echo `uname -a`
```

The output information is written to the two log-files specified in the JDL: `std.out` and `std.err`

Job Submission

- **Overview of available computing element resources:**
  - `lcg-infosites --vo dgtest ce`
- **Check the sites matching your job requirements:**
  - `glite-job-list-match simple.jdl`
- **Job Submission:**
  - `glite-job-submit -o job.id [-r site] simple.jdl`
  - Parameters:
    - `-o <job.id>` Specify the output file for gLite job IDs
      - *not necessary but comfortable*
    - `-r <site>` Specify a CE to directly send your job there

job status

- **Get the job status:**
  - `glite-job-status [-v 1] -i job.id`
  - Parameters:
    - `-i <job.id>` Specifies the input file for gLite job IDs
    - `-v <0|1|2|3>` The higher the verbosity level, the more information you get (e.g. about rescheduled jobs)
- **In case something goes wrong, your job can be cancelled via:**
  - `glite-job-cancel -i job.id`

get job output

Get job output:

- glite-job-output –i job.id [--dir <outputdir>]

**** parameters: --dir <outputdir>specifies the job output directory (e.g. –dir .)***

A Simple Job - Playing Around

- Run `./simple.sh` locally and inspect the output
- Submit the job (2-3 times)
- You will see that the job will run at different sites
- To force the job to a specific site, requirements can be added to the jdl file see later in this tutorial

Data Management - Overview

- **A file on the grid:**
  - has a global unique identifier (GUID)
    - e.g. guid:c73cf5db-0f80-46c1-9571-111d281c70a7
  - can have several replicas at different sites, each having a different physical file name (PFN/SFN)
    - e.g. sfn://iwrgks-2-2.fzk.de/storage/dgtest/generated/2006-09-11/filedeab4811-0e1f-4439-bffa-4efc46f40b82
  - can be given several logical file names (LFN) by the user
    - e.g. lfn:/grid/dgtest/tarballs/dag.tar
  - This information is stored in the LCG file catalog (LFC).

Data Management - LFC

- **Were is the LCG File Catalog located?**
  - `lcg-infosites --vo dgtest lfc`
    - In our case: `iwrlfc.fzk.de`
  - `export LFC_HOST=`lcg-infosites --vo dgtest lfc``
    - `$LFC_HOST` is already set on our UIs (not on WNs)
- **List the existing files and directories, e.g.:**
  - `lfc-ls -l /grid/dgtest`
    - `-l` gives more information like the file size in the catalog
- **Create your own directory in the LFC:**
  - `lfc-mkdir /grid/dgtest/$USER`

Data Management - Storage

- **"How to store a file on the grid?" rises two questions:**
  - Where should the file be stored physically?
  - Which name should be assigned to the file?

- **Get a list of available storage elements:**
  - `lcg-infosites --vo dgtest se`

- **Copy a file to a SE:**
  - `echo "my test file: $USER" > test_$USER.txt`
  - Choose an SE, e.g. `iwrgks-3-2.fzk.de`
  - `lcg-cr --vo dgtest -v -d iwrgks-3-2.fzk.de -l \`
    `lfn:/grid/dgtest/$USER/test.txt \`
    `file://$PWD/test_$USER.txt`
  - → The `-v` option displays verbose output like used LFC, GUID, etc.

Data Management - Storage

- **Where is this file now?**
  - `lcg-lr --vo dgtest lfn:/grid/dgtest/$USER/test.txt`
    - You get the storage file name of the file: `sfn://<host>/<path>`
- **Have a look at the file on the SE:**
  - `edg-gridftp-ls --verbose gsiftp://<host>/<path>`
- **Compare the physical size with the one from the catalog**
  - use `lfc-ls` and `edg-gridftp-ls`

- **Copy the file back in your working directory:**

```
lcg-cp -v --vo dgtest \
        lfn:/grid/dgtest/$USER/test.txt \
            file://$PWD/junk
```

Replicas: various copies of the same file on several SEs –

see next tutorial ☺

- **Cleaning up:**
  - Remove the file from one specific SE:
    - `lcg-del --vo dgtest -s iwrgks-3-2.fzk.de \`
      `lfn:/grid/dgtest/$USER/test.txt`
  - Remove your directory in the LFC
    - `lfc-rm -r /grid/dgtest/$USER`

An (almost) real life example from the Physics world

"data simulation" using ROOT (http://root.cern.ch)

- If you require special software packages, there is no need to send them together with every job.
- There is a special user-role in each VO, the **softwaremanager**:
  - In case of the VO "dgtest" this is "dgtestsgm"
  - The softwaremanager can install software in the directory `$VO_DGTEST_SW_DIR` on each site.
- Installed software is published via "**glue-tags**". They can be put into the requirements of the jdl-file, e.g.:

Requirements = Member("VO-dgtest-ROOTv5.13.04", other.GlueHostApplicationSoftwareRunTimeEnvironment)

producing "data"

our "data" shall be: a histogramm with random numbers following a Gauss distribution

for that please copy to your local work dir:

- /opt/root/jdl/gauss.jdl

- /opt/root/bin/gauss.sh

- /opt/root/macro/gauss.C

gauss.jdl

```
Executable = "gauss.sh";
Arguments = "";
StdOutput = "stdout";
StdError = "stderr";
InputSandbox = {"gauss.sh","gauss.C"};
OutputSandbox = {"stdout","stderr","gauss.root"};
VirtualOrganisation = "dgtest";
Requirements = Member("VO-dgtest-ROOTv5.13.04",
other.GlueHostApplicationSoftwareRunTimeEnvironment);
RetryCount = 3;
```

gauss.sh

```
#!/bin/bash
source /opt/root/.rootlogin
root -b -q gauss.C
export LFC_HOST=iwrlfc.fzk.de
# please do locally beforehand:
# lfc-mkdir /grid/dgtest/$USER
lcg-cr --vo dgtest -v -d iwrgks-3-2.fzk.de
-l lfn:/grid/dgtest/YOURUSERNAME/gauss.root
file://$PWD/gauss.root
```

please type indeed your user name, and not $USER, since on the WNs only Pool accounts are existing !!!

gauss.C

```
{
TObjArray Hlist(0);
TH1F* h1;
h1 = new TH1F("h1","Histo from a Gaussian",100,-3,3);
Hlist->Add(h1);
h1->FillRandom("gaus",10000);
TFile f("gauss.root","recreate");
Hlist->Write();
f.Close();
}
```

submit the job

as usual via

glite-job-submit –vo dgtest –o job.id gauss.jdl

it will only run on the FZK WNs since only there the proper ROOT version has been installed.

this can be checked also via

lcg-infosites –vo dgtest tag

watch status, get output as usual

look at gauss.root via ROOT

. /opt/root/.rootlogin

root

root>TFile f("gauss.root")

root>f->ls()

root>h1->Draw()

Data Analysis

in this exercise we try to find the "data" of the last exercise in the Grid to analyse them.

The analysis shall be: we do a gaussfit through the distribution we created in the last exercise

Please download to your work dir:

- /opt/root/jdl/gaussfit.jdl

- /opt/root/bin/gaussfit.sh

- /opt/root/macro/gaussfit.C

gaussfit.jdl

```
Executable = "gaussfit.sh";
Arguments = "";
StdOutput = "stdout";
StdError = "stderr";
InputSandbox = {"gaussfit.sh","gaussfit.C"};
OutputSandbox = {"stdout","stderr","gaussfit.root"};
VirtualOrganisation = "dgtest";
Requirements = Member("VO-dgtest-ROOTv5.13.04",
other.GlueHostApplicationSoftwareRunTimeEnvironment);
RetryCount = 3;
```

gaussfit.sh

```
#!/bin/bash
echo "LFC HOST = " $LFC_HOST
lcg-cp --vo dgtest -v
lfn:/grid/dgtest/YOURUSERNAME/gauss.root
file://$PWD/gauss.root

source /opt/root/.rootlogin
root -b -q gaussfit.C
```

please type indeed your user name, and not $USER, since on the WNs only Pool accounts are existing !!!

gaussfit.C

```
{
TObjArray Hlist(0);
TFile f("gauss.root");
Hlist->Add(h1);
h1->Fit("gaus");
TFile f2("gaussfit.root","recreate");
Hlist->Write();
f.Close();
f2.Close();
}
```

submit, watch status, get ouput as learned before

start ROOT and look at ROOT output the way we have done it before.

Job Submission via the LHCb tool Ganga

see tutorial of
Ulrik Egede

README at

/home/icfa26/README