LCG-2 Induction Course

Rüdiger Berlich
(Rüdiger.Berlich@iwr.fzk.de)

Christopher Jung
(Christopher.Jung@iwr.fzk.de)

Slides contributed by Torsten Antoni, Christopher Jung and Rüdiger Berlich
Special thanks to Volker Büge

Information Society

www.eu-egee.org

Topics:

Preliminaries

- User Interface
- User accounts

Certificates

Job submission

Data management

Physics examples

Topics:

Preliminaries

Certificates

Job submission

Data management

Physics examples

Topics:

Preliminaries

Certificates

Job submission

- Hello World
- Submit your own shell script

Data management

Physics examples

Topics:

- **Preliminaries**
- **Certificates**
- **Job submission**
- **Data management**
  - Concept
  - Replica management
  - Job to the data
- **Physics examples**

Topics:

- Preliminaries
- Certificates
- Job submission
- Data management
- Physics examples
  - Geant3/Root
  - Pythia

Preliminaries

Preliminaries

Network

- the User Interface is “grid-tutor.ct.infn.it“
- you can log on it via ssh
- the account names are “desyXX“, the passwords are “GridDESXX“
- the grid passphrase is “DESY“
- you need an IP in the DESY network

Certificates

Rüdiger Berlich, Christopher Jung

Certificate files

Two files:

- `usercert.pem` (permissions: 644)
- `userkey.pem` (permissions: 200)

Stored in $HOME/.globus:

Transfer to p12-format (e.g. if you want to sign your e-mails using your certificate):

```
openssl pkcs12 -export -in usercert.pem\
  -inkey userkey.pem -out cert.p12\
  -name "name for certificate“
```

Grid Proxy

- "logging in"
  - **grid-proxy-init**

```
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Your proxy is valid until: Wed Sep  22 04:49:57 2004
```

    Parameter: `-valid hh:mm` (e.g. `–valid 24:00`, lasts a day)

  - **grid-proxy-info**
  - **grid-proxy-destroy**

Job submission

Rüdiger Berlich, Christopher Jung

Hello World

Job Description Language (JDL) is used for neccessary inputs, generated outputs and resource requirements

Create a file called HelloWorld.jdl with the this content:

```
Executable = "/bin/echo";
Arguments = "Hello World";
StdOutput = "stdout";
StdError = "stderr";
OutputSandbox = {"stdout","stderr"};
```

Hello World

Overview of available resources:

```
edg-job-list-match --vo gilda HelloWorld.jdl
```

Parameters: `--vo <vo_name>` specifies your VO to the RB

(specification of VO not needed on grid-tutor.ct.infn.it)

Hello World

Job submission

```
edg-job-submit --vo gilda -o job.id HelloWorld.jdl
```

Parameters: `-o <JobId>` specifies the output file for
edg job id(s)
not necessary, but comfortable

`-r <ce_id>` specifies a CE to submit
the job directly to

Rüdiger Berlich, Christopher Jung

Hello World

Job status

```
edg-job-status -v 1 -i job.id
```

Parameters: `-i <JobId>` specifies input file for edg job id(s)

`-v <verbosity level> the higher the verbosity level the more information you get (e.g. about a rescheduled job)`

Hello World

Cancel Job

```
edg-job-cancel –i job.id
```

Get output

```
edg-job-get-output -i job.id --dir .
```

Parameter: `--dir <OutputDir> specifies the job output directory`

Standard output directory:

- In general: `/tmp/jobOutput/<username>_<edg_job_id>`
- For gilda: `$HOME/jobOutput/<username>_<edg_job_id>`

Hello World

Run “Hello World”.
Improve it to get the name of the worker node it ran on, via “/bin/hostname”.
Submit the job (2-3 times).

You will see that the job run at different sites

Shell script

Submit a shell script that gathers useful information on the worker node it ran on, e.g: free disc space, time, date, …

.jdl file extensions:

```
InputSandbox = {"script.sh"};
```

Advanced: copy the /etc/passwd from the worker node and return it via the OutputSandbox

That's it for today!

Data management

Rüdiger Berlich, Christopher Jung

Data management

A file at a SE:

- has a Globally Unique Identifier (GUID)
- can have several replicas at different sites, each having a different Physical File Name (PFN)
- can be given one or several Logical File Names (LFN) by the user

These informations are stored in

the Replica Metadata Catalog (RMC)

and the Replica Location Service (RLS)

Data management

Replica manager (`edg-rm`)

User tool to manage replicas

```
edg-rm –v --vo <VO> <command> [<parameters>]
```

e.g.:

```
edg-rm --vo gilda printInfo
```

shows extensive information about the available SEs:

`hik-lcg-se.fzk.de, grid-se.desy.de, …`

There are now lcg commands for data management, which will be used in the following. The documentation on them is unfortunately short.

Data management

Uploading a file to the Grid (from a WN or a UI)

```
lcg-cr --vo gilda file://`pwd`/<name> \
     -l lfn:<name> -d <SE>
```

–d: specifies destination (if not given local SE is used)

-l: specifies logical file name

("cr" stands for "copy and register")

Replicate file

```
lcg-rep –v --vo gilda \
 -d <2nd SE> lfn:<name>
```

–v: verbose (always helpful in grid computing)

Of course you can also specify a file by its guid.

Data management

list replicas

```
lcg-lr --vo gilda lfn:<name>
```

(file can again also be specified by its guid)

List guid

```
lcg-lg --vo gilda lfn:<name>
```

(file can also be specified by its surl)

Copying files out of the grid (to a UI or to a WN)

```
lcg-cp --vo gilda lfn:<name>
  file://$HOME/myGridFile
```

(local file name is of course an example)

Data management

There is even one lcg command for getting the SE names:

lcg-infosites --vo gilda se

lcg-infosites --vo gilda closeSE (get the close SE(s))

Deleting files on the grid

Command **lcg-del**

- with a surl: **lcg-del --vo gilda sfn:<name>**
- with a lfn or guid: you also have to specify the storage element via the **"-s <SE>"** option
- deleting all(!) replicas of a file with a lfn or a guid: use the **"-a"** option; file will also be unregistered

There are still more data management commands, like adding (lcg-aa) and removing aliases (lcg-ra).

Data management

Exercise

Create files (different names, small)
Register
Replicate (plurally, also remove replicas)
Show replicas
Copy back replicas

Finally: remove all replicas !!!!

Data management

Problems in everyday use

Size of a job: about 10MB

Size of the data: about 10GB

=> Job to the Data!

Data management

If you want your job go to the data, just add the following lines to your JDL file

```
InputData = {"lfn:<LFN>"};
DataAccessProtocol = {"gsiftp"};
```

Data management

Exercise

write a job and a script that do the following:

Run at a site where a file has been registered
Copy that file to the WN, display it

Create an output in a job, register it in theReplica Catalog, return the LFN and the guid

Afterwards:
Copy file to User Interface (using the LFN)
Delete your files

Data management

Exercise

- Register two files at two different sites; require them both in the jdl file; does your job run?
- Try to copy and register two files with the same logical file name
- Work with somebody using a different account; can you delete his/her files on the grid

- If you are done with all the exercises, try to use the edg-rm command for some of those exercises.

Physics Examples

Rüdiger Berlich, Christopher Jung

Physics Software and the Grid

Physics Software at big experiments is big

Cannot send it via InputSandbox

Solution:

an experiment software manager installs software

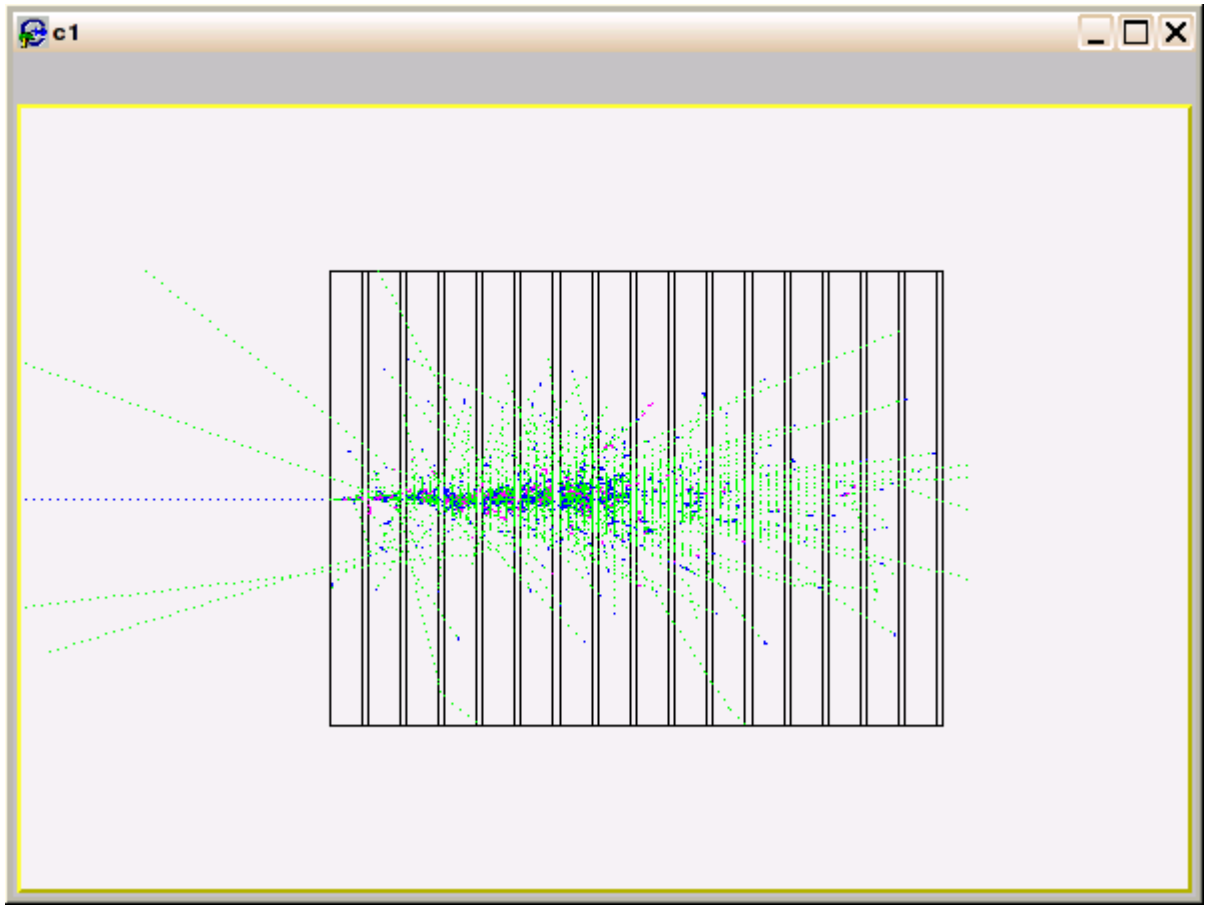
Software directory: $VO_<vo_name>_SW_DIR

You can send your job via the RB only to those sites where the software has been installed

(“software tag in JDL file”)

Geant 3 / Root Example

Simulation of an electromagnetic calorimeter



Rüdiger Berlich, Christopher Jung

Exercise

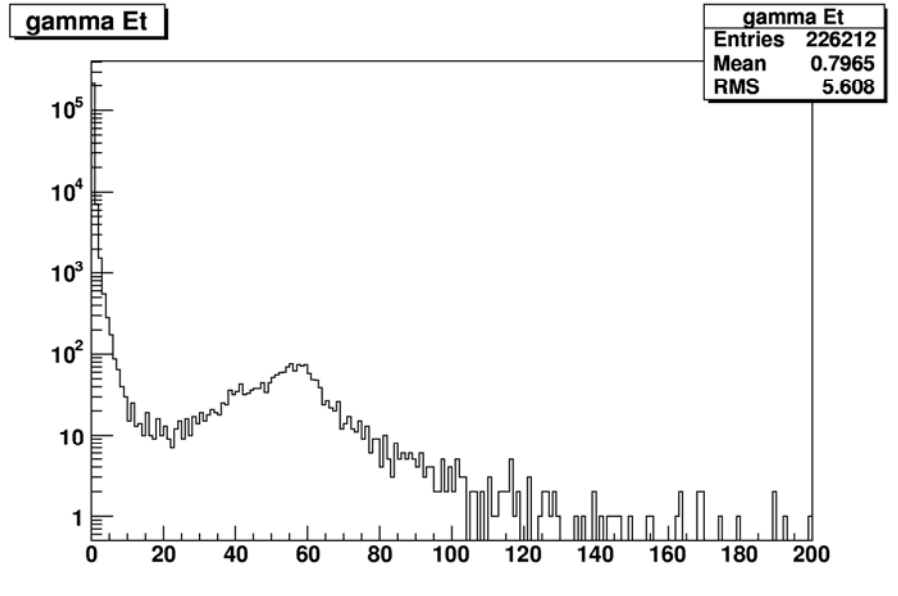
Write a job that does:

- Copy `lfn:geantroot.tgz` to your working directory
- Unpack geantroot.tgz (`tar –zxvf geantroot.tgz`)
- Change to subdirectory `gridtest`
- Source `setup.sh` (this is a bash script, so please use a bash script for your job)
- Run `calibrateEM.c` in batch mode (`root –b –q calibrateEM.C`)
- Return out.root and hcounts.pdf (both in subdirectory gridtest!) via your OutputSandbox
- Please use the following requirement in your JDL-file:

```
Requirements = Member("VO-gilda-GEANT",
   other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

Pythia

In CMS software, MC generator wrapper CMKIN is used. It gets the information via a datacard (`h_gamgam.txt`).



Pythia

Write a job that does:

- Sending the files “beispiel” and “h_gamgam.txt”.
- Start CMS environment
  (source $VO_GILDA_SW_DIR/cms/cmkin_set.sc)
  (this script is a tcsh script, so please use tcsh for your script as well)
- Start Pythia (./kine_pyth.run h_gamgam.txt)
- Convert h_gamgam.ntpl to h_gamgam.root (via h2root)
- Run beispiel (set it to executable first)
- Return files “h_gamgam.root” and “Gammas.ps”
- (Gammas.ps can be converted to pdf using ps2pdf on UI)
- You need the CMS software. So use the following requirement in your JDL-file:

```
Requirements = Member("VO-gilda-CMKIN",
 other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

… and now use the grid in everyday's work!

(also read the user guide:

https://edms.cern.ch/file/454439/1/LCG-2-UserGuide.pdf)