# Practical using EGEE middleware

*Dr. Mike Mineter & Gergely Sipos*

*Taipei, 1 May 2006*

# PLEASE DOWNLOAD THIS FILE

- **Please download this file from the agenda page.**
- **You will need to refer to it during the practical.**

- **Browse to:**
- **http://agenda.cern.ch/fullAgenda.php?ida=a061940**

  - Look at the first practical on the agenda
  - Left click on “transparencies”
  - Select ppt or pdf as you prefer

# Scope

- **We are using the GILDA testbed today**
  - The production EGEE grid looks like this!
  - Almost-current EGEE production middleware
- **The practical exercises are to illustrate "how"**
  - Not using typical jobs for running on a grid!!
  - But to show how EGEE grid services are used, jobs are submitted, output retrieved,…
- **We will use the Command-Line Interfaces on a "User Interface" (UI) machine**
  - "UI" is your interface to the GILDA Grid
    - Where your digital credentials are held
    - Client tools are already installed

# Practicals: outline

- **Introduction to the basic services**
  - Authorisation and Authentication
  - Workload Management – simple job submission
  - Information System
  - Data Management
  - “Putting it all together” – more realistic job submission

# The GILDA Test-bed (https://gilda.ct.infn.it/testbed.html)


15 sites in 3 continents !

**GILDA is coordinated by Roberto Barbera and colleagues at the University of Catania and INFN.**

# Trust

- **Grids by definition have no central control, and connect users to resources where neither has prior knowledge of the other**

- **Need to establish trust, so**
  - Resource can trust user
  - User can trust the resource

- *The basis:*
  - ***CAs sign user and resource (site) certificates***
  - ***Both users and sites trust Certificate Authorities***
  - ***So users and providers trust each other***

- *…and for international collaboration:*
  - ***CAs trust each other***

# To use the EGEE grid

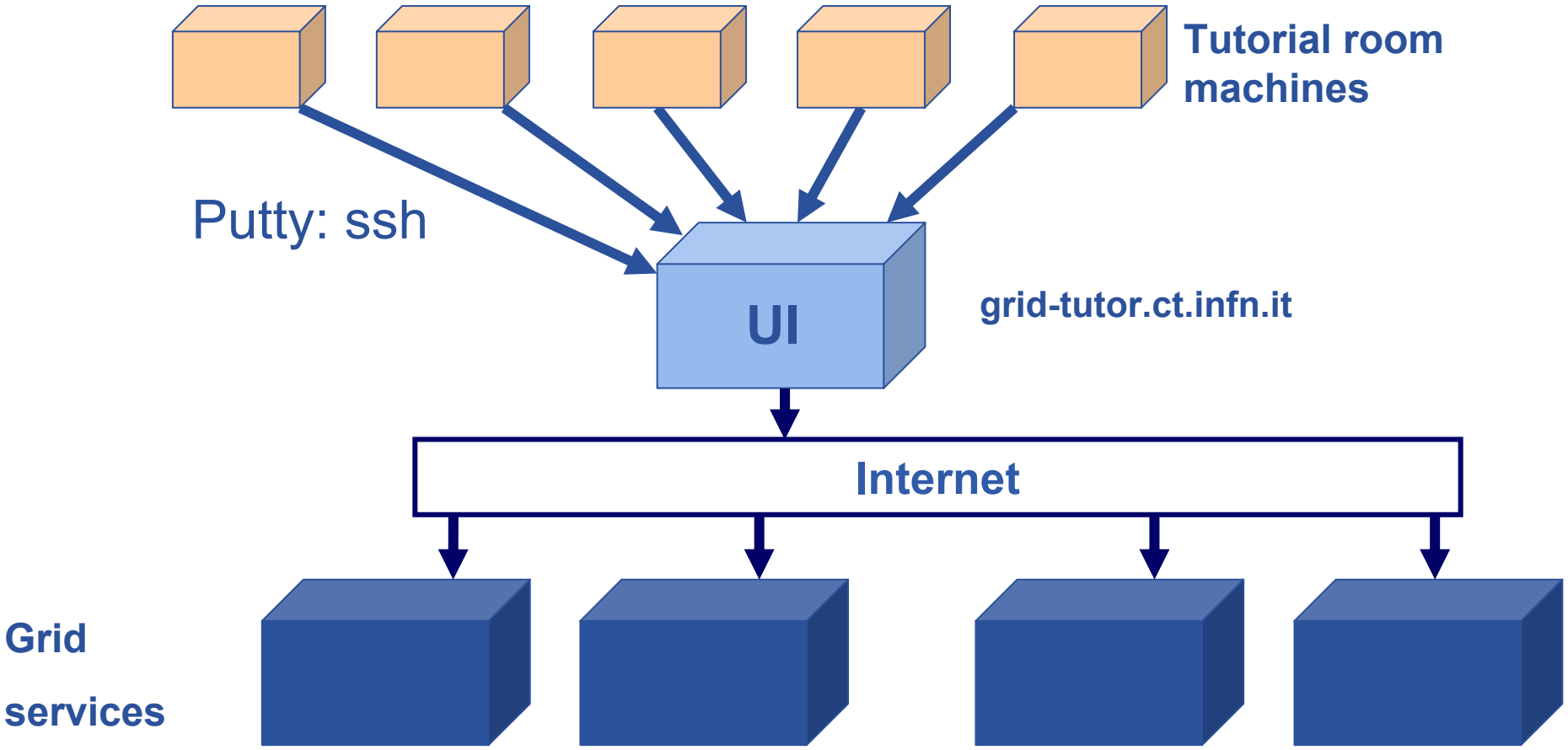
- **Get an internationally recognised certificate**
  - From a local RA – you will need to see them personally, bringing passport or other identification
- **Contact the virtual organisation (VO) manager**
- **Accept the VO and the EGEE conditions of use**
- **The VO manager authorises you to use resources**
- **Upload your certificate to a "User Interface" machine**

- **We are continuing the practical from this stage**
- **You are a member of the GILDA VO**
- **We have training certificates on the GILDA testbed**

# Using GILDA

- **If you are new to Linux – or if you prefer – work in pairs**
- **You will need to edit files and use command-line interfaces**

# Our setup


Tutorial room machines
Putty: ssh
UI
grid-tutor.ct.infn.it
Internet
Grid services

# User Interface Access

**Host: grid-tutor.ct.infn.it**

**Username: taipeiXX (XX=02…40)**

**Password: GridTAIXX (XX=02…40)**

**ssh grid-tutor.ct.infn.it -l taipeiXX**

**You are a member of the "gilda" VO**

- Letter "l"

**Create a directory that you will work in:**

**mkdir myfiles**

**and wait here please!!**

# Preliminary : .globus directory

- .globus directory contains your personal public / private keys

```
[taipei49@grid019 taipei49]$ ls -l .globus/
-rw-r--r--    1 taipei49 users   1111 Mar  6 14:51 usercert.pem
-r--------    1 taipei49 users    963 Mar  6 12:57 userkey.pem
```

Type: "ls –l .globus"

Notice the file permissions !

userkey.pem: private key

usercert.pem: public key + credential + CA signature

# Verify your certificate

- To get information on your certificate, run
- $ openssl x509 -in .globus/usercert.pem -noout -text

Be careful if you cut-and-paste: check "-" for example !

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3532 (0xdcc)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=IT, O=GILDA, CN=GILDA Certification
  Authority
        Validity
            Not Before: Mar  6 10:44:49 2006 GMT
            Not After : Mar 26 10:44:49 2006 GMT
        Subject: C=IT, O=GILDA, OU=Personal Certificate,
  L=taipei, CN=taipei49/Email=mjm@nesc.ac.uk
```

NOTICE THE ISSUER AND THE SUBJECT

# Proxy creation

```
voms-proxy-init --voms gilda
```

- You must **be in your home directory** for this.
  - to get there from another directory just type: cd
- You must **include "-- voms gilda"**
  - To create your GILDA VO credential
  - Otherwise there will be authorisation failures

**You will be asked for a passphrase – your private key is encrypted.**

**PEM PASSPHRASE : TAIPEI**

# proxy creation

```
voms-proxy-init --voms gilda

Your identity: /C=IT/O=GILDA/OU=Personal
   Certificate/L=taipei/CN=taipei49/Email=mjm@nesc.ac.uk
Enter GRID pass phrase: TAIPEI
Creating temporary proxy
   ..................................................... Done
Contacting  voms.ct.infn.it:15001
   [/C=IT/O=GILDA/OU=Host/L=INFN
   Catania/CN=voms.ct.infn.it/Email=emidio.giorgio@ct.infn.it]
   "gilda" Done
Creating proxy
   ..................................................... Done
Your proxy is valid until Fri Mar 10 23:32:12 2006
```

# voms-proxy-init : more options

- **For information – do not run these**
- **Main options**

  **-voms <vo-name**:[command]>

  - **command** syntax is :/<voname>/group for group specify (default none)
  - **command** syntax is :/<voname>/Role=<role name> for Role choice (default none)
  - Multiple –voms can be set to aggregate rights

```
voms-proxy-init --voms gildav:/gildav/Role=VO-Admin
voms-proxy-init --voms gildav:/gildav/tutors
```

-valid x:y, create a proxy valid for x hours and y minutes

-vomslife x, create a proxy with AC valid for x hours (max 24 h)

# What is in your proxy??

- Please type:

```
voms-proxy-info --all
```

- There are two credentials
  1. the proxy with your identity
  2. from the VOMS server for authorisation
  - Option "– all" shows details of both
- Compare the first credential with that which you saw in your certificate. What's different??

# What is in your proxy??

```
[...]$ voms-proxy-info --all
subject   : /C=IT/O=GILDA/OU=Personal
  Certificate/L=taipei/CN=taipei49/Email=mjm@nesc.ac.uk/CN=pr
  oxy
issuer    : /C=IT/O=GILDA/OU=Personal
  Certificate/L=taipei/CN=taipei49/Email=mjm@nesc.ac.uk
identity  : /C=IT/O=GILDA/OU=Personal
  Certificate/L=taipei/CN=taipei49/Email=mjm@nesc.ac.uk
type      : proxy
strength  : 512 bits
path      : /tmp/x509up_u3519
timeleft  : 11:55:42
VO        : gilda
subject   : /C=IT/O=GILDA/OU=Persona
  Certificate/L=taipei/CN=taipei49/Email=mjm@nesc.ac.uk
issuer    : /C=IT/O=GILDA/OU=Host/L=INFN
  Catania/CN=voms.ct.infn.it/Email=emidio.giorgio@ct.infn.it
attribute : /gilda/Role=NULL/Capability=NULL
timeleft  : 11:55:42
```

1. Proxy of CA certificate
2. Credential from VO

# Try to use the grid with a proxy

**Now you have a proxy with additional VOMS credential, try a command**

- **hostname.jdl is a simple job description file.**
- **To see which compute elements (CE)s can run this job we would use the command:**

  **edg-job-list-match hostname.jdl**

  Please try this command!!

  The result is a list of the CEs (batch queues) where this job can be run… more later!

## Creating a proxy certificate is your logon to the grid

# Try to use the grid without a proxy!

- **Delete your proxy: voms-proxy-destroy**
- **Re-try**

**edg-job-list-match hostname.jdl**

What happens? Remember this – for one day it will happen again!!

- **Create a new proxy:**

**voms-proxy-init --voms gilda**

# The main AA commands

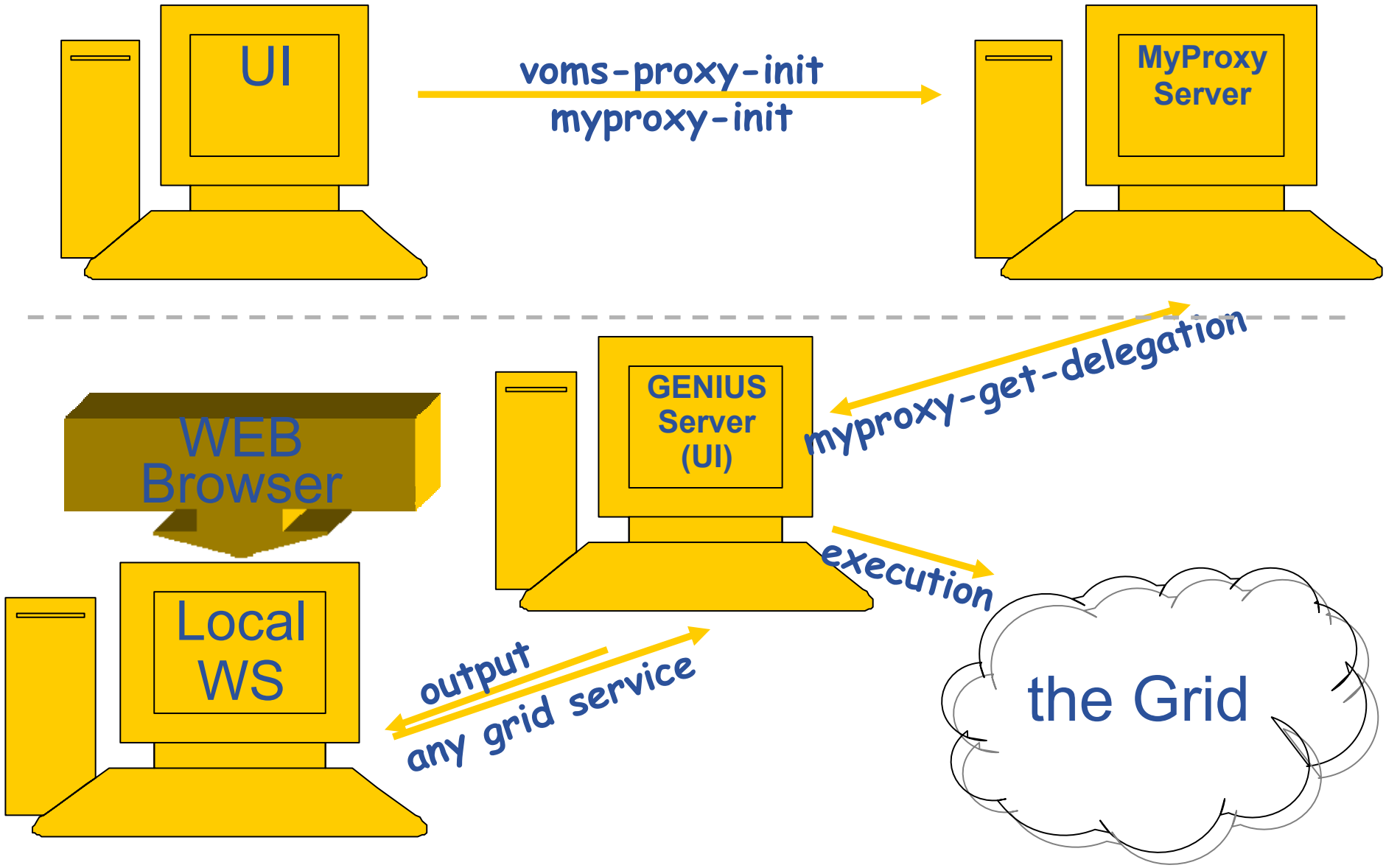
- **Create a VOMS proxy:**
  - **voms-proxy-init** --voms <VOServer>
- **Display information:**
  - **voms-proxy-info --all**
- **Destroy the proxy:**
  - **voms-proxy-destroy**
- **And note the "-help" option on all commands**
  - **voms-proxy-init -help**

# "MyProxy"

- **You may need:**
  - To interact with a grid from many machines
    - And you realise that you must NOT, EVER leave your certificate where anyone can find and use it…. Its on a USB drive only.
  - To use a portal, and delegate to the portal the right to act on your behalf (by logging in to an account that can make a proxy certificate for you)
  - To run jobs that might last longer than the lifetime of a short-lived proxy

- **Solution: you can store a long-lived proxy in a "MyProxy repository" and derive a proxy certificate when needed.**

# Grid authentication with MyProxy



# Upload a proxy to MyProxy Server

**myproxy-init -s grid001.ct.infn.it**

- **-s specifies the MyProxy server**

- **Use “myproxyXX” as your myproxy pass phrase…**
- **XX – your user number**

# GENIUS portal – a quick practical

- In a browser go to https://grid-tutor.ct.infn.it/
- (When asked: Accept for this session only)
- Select “Set VO/VOMS” and confirm you are acting in the GILDA VO by “set”
- Enter your taipei*XX* username and the MyProxy pass phrase (myproxyXX)
- Choose job service- job submission –single job
- Enter /home/taipeiXX/hostname.jdl
- Allow the resource broker to choose
- Submit the job

# GENIUS

- **Can be tailored to suit a particular VO and its applications**
- **Allows Grid jobs to be run from any browser**
- **MyProxy enables this by issuing the portal server with a proxy on your behalf**
- **Many VOs members do not have time or inclination to code!**
  - Need to be provided with an easy interface
- **GENIUS is provided by the University of Catania and NICE**

# Has your job run?

- Choose “single job” then “job queue”
- If it has completed, retrieve the output file.

- You can return to this later.

- Anyone can use GENIUS via https://grid-demo.ct.infn.it/

# MyProxy

Consists of a server and a set of client tools that can be used to delegate and retrieve credentials to and from a server.

**MyProxy Client commands:**

- *myproxy-init*
- *myproxy-info*
- *myproxy-destroy*
- *myproxy-get-delegation*
- *myproxy-change-pass-phrase*

```
// myproxy-info -s <host name> -d
```

```
// myproxy-get-delegation -s <host name> -d
   -t <hours> -o <output file> -a <user proxy>
```

The ***myproxy-init*** command allows you to create and send a delegated proxy to a MyProxy server for later retrieval; in order to launch it you have to assure you're able to execute the voms-proxy-init command.

```
myproxy-init -s <host name> -t <hours> -d -n
```

The myproxy-init command stores a user proxy in the repository specified by <host name> (the –s option). Default lifetime of proxies retrieved from the repository will be set to <hours> (see -t) and no password authorization is permitted when fetching the proxy from the repository (the -n option). The proxy is stored under the same user-name as is your subject in your certificate (-d).

# User Responsibilities

- Keep your private key secure.
- Do not loan your certificate to anyone.
- Report to your local/regional contact if your certificate has been compromised.
- Do not launch a proxy for longer than your current task needs.

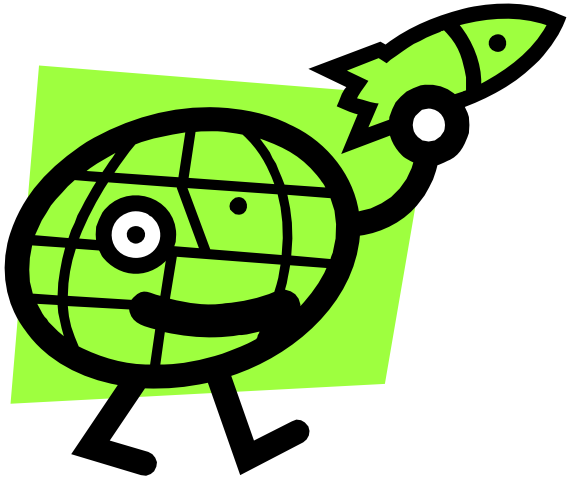
**If your certificate or proxy is used by someone other than you, it cannot be proven that it was not you.**

# Summary

- **The EGEE multi-VO grid is built on**
  - Authentication based on X.509 digital certificates
    - Issued by CAs that are internationally recognised (enabling international collaboration)
    - With proxies
  - Authorisation provided by VOMS
    - VOMS supports
      - *multiple groups, roles within a VO*
- **voms-proxy-init: is your logon to the grid**
- **MyProxy**
  - Secure storage of long-lived proxy certificates
  - Delegation so services can create and use a proxy on your behalf

# References

- VOMS on EGEE: User Guide available at http://glite.web.cern.ch/glite/documentation/default.asp
- VOMS
  - Available at http://infnforge.cnaf.infn.it/voms/
  - Alfieri, Cecchini, Ciaschini, Spataro, dell'Agnello, Fronher, Lorentey, From gridmap-file to VOMS: managing Authorization in a Grid environment
  - Vincenzo Ciaschini, A VOMS Attribute Certificate Profile for Authorization
- GSI
  - Available at www.globus.org
  - A Security Architecture for Computational Grids. I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. *Proc. 5th ACM Conference on Computer and Communications Security Conference*, pp. 83-92, 1998.
  - A National-Scale Authentication Infrastructure. R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch. *IEEE Computer*, 33(12):60-66, 2000.
- RFC
  - S.Farrell, R.Housley, An internet Attribute Certificate Profile for Authorization, RFC 3281



# Workload Management System

# Workload Management System

- **The user interacts with the EGEE Grid via a Workload Management System (WMS)**

- **What does it allow users to do?**
  - To submit their jobs
  - To execute them on the “best resources”
    - The WMS tries to optimize the usage of resources
  - To get information about their status
  - To retrieve their output

- **WMS “virtualises” the many compute resources of the grid**

- **Why do commands start with edg?**
  - European Data Grid project is a precursor of LCG and EGEE

# Job Description Language (JDL)

- **Job submission: a JDL file is sent to the Resource Broker**
- Job Attributes
  - Define the job itself
- Resources
  - Taken into account by the RB for carrying out the matchmaking algorithm (to choose the “best” resource where to submit the job)
  - *Computing Resource*
    - *Used to build expressions of Requirements and/or Rank attributes by the user*
    - *Have to be prefixed with “other.”*
  - *Data and Storage resources*
    - *Input data to process, SE where to store output data, protocols spoken by application when accessing SEs*
- **Based upon Condor’s *CLASSified ADvertisement language (ClassAd)***

# Job Submission

- **edg-job-submit** performs the job submission to the WMS
- Returns a job identifier, not waiting for the job to execute

Usage: **edg-job-submit** ***[options]*** **<jdl file>**

**Principal Options** :

**--vo** <vo name> : perform submission with a different VO than the UI default one ($echo

**--output, -o** <myjobid file> save jobId in a file, instead of STDIN

***Please type:***

***cd myfiles***

**edg-job-submit -o myjob.ids ../hostname.jdl**

**cat ../hostname.jdl**

**cat myjob.ids**

# JDL syntax

- **An attribute is a pair (key, value), where value can be a Boolean, an Integer, a list of strings, ....**
  - <attribute> = <value>;
- **In case of literal string for values:**
  - if a string itself contains double quotes, they must be escaped with a backslash
    - `Arguments = " \"Hello\" 10";`
  - the character “” cannot be specified in the JDL
  - special characters such as &, |, >, < are only allowed
    - if specified inside a quoted string
    - if preceded by triple \
      - `Arguments = "-f file1\\\&file2";`
- **Comments must be preceded by a sharp character (#) or have to follow the C++ syntax**
- **The JDL is sensitive to blank characters and tabs**
  - they should not follow the semicolon (;) at the end of a line

# JDL – Relevant Attributes

**JobType**

*Normal* (simple, sequential job) *Interactive*, *MPICH*

Or combination of them

**Executable** (mandatory)

The command name

**Arguments** (optional)

Job command line arguments

**StdInput**, **StdOutput**, **StdError** (optional)

Standard input/output/error of the job

**InputSandbox** (optional)

List of files on the UI local disk needed by the job for running

The listed files will automatically staged to the remote resource

**OutputSandbox** (optional)

List of files, generated by the job, which have to be retrieved

# WMS commands

- **edg-job-submit  < job id>**
- **edg-job-status  <job id>**  check job execution status
- **edg-job-get-output  <job id>**

  If job status is 'done', retrieve output, specifying directory to receive it, e.g.:

  edg-job-get-output --dir <outputdir in your UI> -i <file>
- **edg-job-cancel  <job id>**  perform job deletion
- **edg-job-get-logging-info <jobid>**
  **s**ee log of the job

All of these commands accept the option **–i <myjobidfile>** input from a file created by edg-job-submit to avoid entering long job id by hand.

**edg-job-status  -i myjob.ids**

*If "done" then retrieve output and see where your job ran:*

***mkdir myresults***

***cd myresults***

**edg-job-get-output --dir `pwd`  -i myjob.ids**

**Explore the files!**

# JDL – running a script

```
Type = "Job";
JobType = "Normal";
Executable = "/bin/bash";
StdOutput = “std.out";
StdError = “std.err";
InputSandbox = {“yourscript.sh"};
OutputSandbox = {“std.err",“std.out"};
Arguments = "yourscript.sh";
```

# JDL – Requirements

- **“Requirement” constrains the RB**
- **Only one requirement can be specified - if there is more than one, only the last one is taken into account**
  - If you need several Requirements, combine them through logical operators (&&, ||, !, .....).
- **Examples:**

```
#Insert a requirement to select a short queue
Requirements = (other.GlueCEPolicyMaxWallClockTime < 1440);

#Insert a requirement to select a long queue
Requirements = (other.GlueCEPolicyMaxWallClockTime > 1440);

#Insert a requirement to select an infinite queue
Requirements = (other.GlueCEPolicyMaxWallClockTime > 2880);

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

# Exercise

**cd ~/myfiles**

***create a new script, yourscript.sh***

```
#!/bin/sh
hostname
date
whoami
```

**cp ../hostname.jdl taipei1.jdl**

***Then:***

***Modify taipei1.jdl file so yourscript.sh will be run***

***Add a requirement that the job should be run in a short queue***

**Submit the job, check its status, find which queue it is in**

**Read on whilst your job runs!**

# Matching Jobs to Resources

- **edg-job-list-match** returns suitable resources for execution
- No job submission is performed

•Usage: **edg-job-list-match** ***[options]*** **<jdl file>**

•**Principal Options** :

**--vo** <vo name> : perform list-match with a different VO than the UI default one

--**rank** show resources in order of ranking

**--output, -o** <output file> redirect output to a file, instead of STDIN

**--debug** show function calls and parameters

# Exercise continued

***Use edg-job-list-match and compare the output with the two jdl files you submitted before: hostname.jdl and taipei1.jdl.***

***(The second was directed to a short-job queue.)***

***For each job you submitted (unless you've already retrieved the output, then submit another using taipei1.jdl):***

*Use edg-job-get-logging-info and follow the job's history*

*Check status and when "done" retrieve the output*

# .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 transferred 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-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 part 1

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

```
#!/bin/sh
MY_NAME= "your username"
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
```

# Exercise continued

## Exercise part 2

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

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

Remove leading/trailing "spaces" in the JDL file

# Exercise continued

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

# Practical:
# The Information Systems

# Uses of the Information System

## If you are a user

Retrieve information about
- Grid resources and status
- Resources that can run your job
- Status of your jobs

## If you are a middleware developer

Workload Management System:
Matching job requirements and Grid resources

Monitoring Services:
Retrieving information about Grid Resources status and availability

## If you are site manager or service

You “generate” the information for example relative to your site or to a given service

# Evolution

- **The data published in the Information System (IS) conforms to the GLUE (Grid Laboratory for a Uniform Environment) Schema. The GLUE Schema aims to define a common conceptual data model to be used for Grid resources.**
  **http://infnforge.cnaf.infn.it/glueinfomodel/**
- **In LCG-2, the BDII (Berkeley DB Information Index), based on an updated version of the Monitoring and Discovery Service (MDS), from Globus, was adopted as main provider of the Information Service.**
- **R-GMA (Relational Grid Monitoring Architecture) is now adopted as IS in both the EGEE production grid (mainly "LCG-2") and in the pre-production grid (moving to "gLite 3.0")**

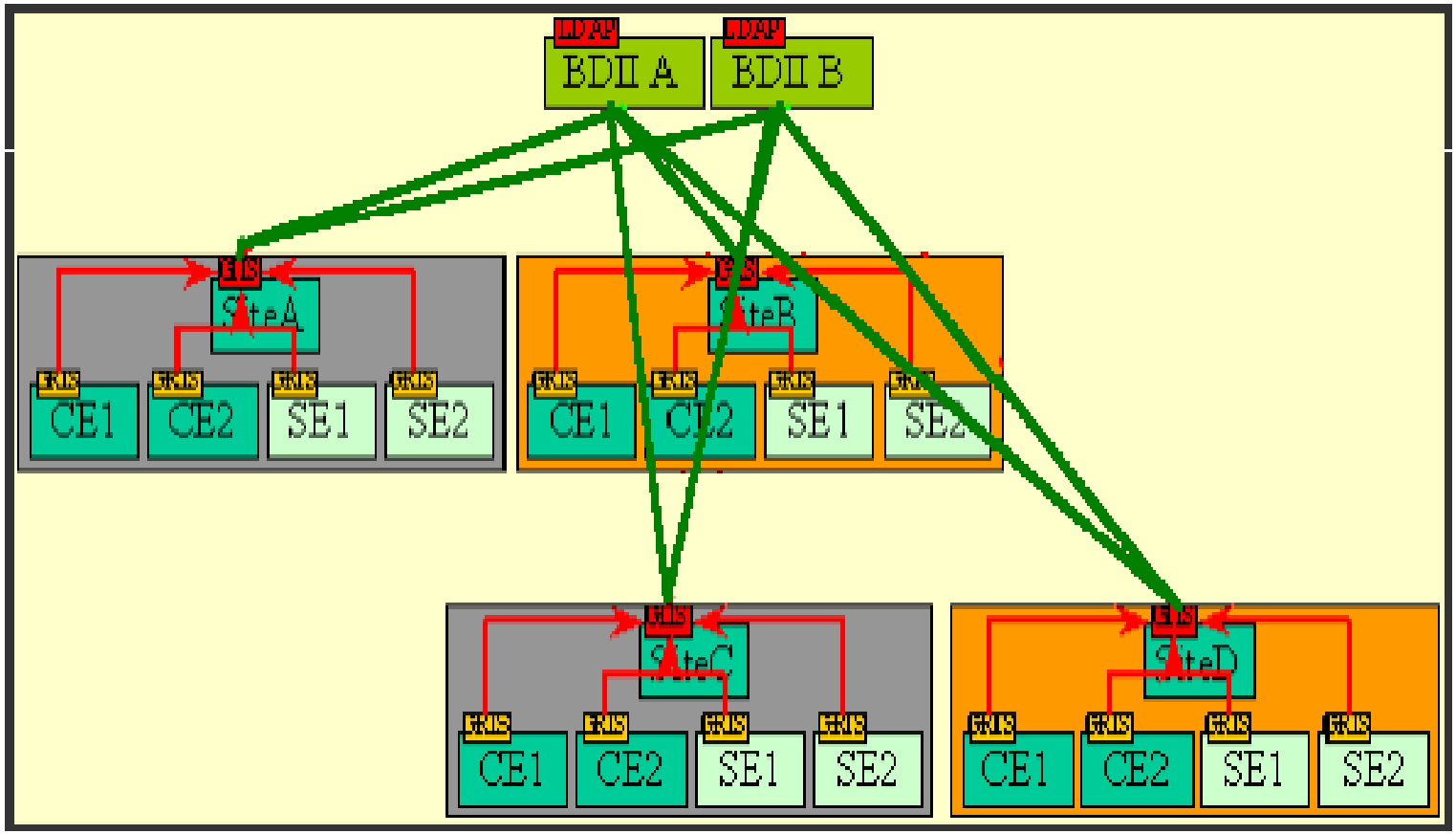
# Overview Of Practicals

- **BDII Information System**
  - *main Information System for the current production grid*
  - **Two sets of commands:**
    - **lcg-infosites: simple, meets most needs**
    - ( lcg-info: supports more complex queries – NOT TODAY!!)

# lcg-infosites

# LCG Information Service

- **a user or a service can query**
  - **the BDII (usual mode)**
  - **LDAP servers on each site**

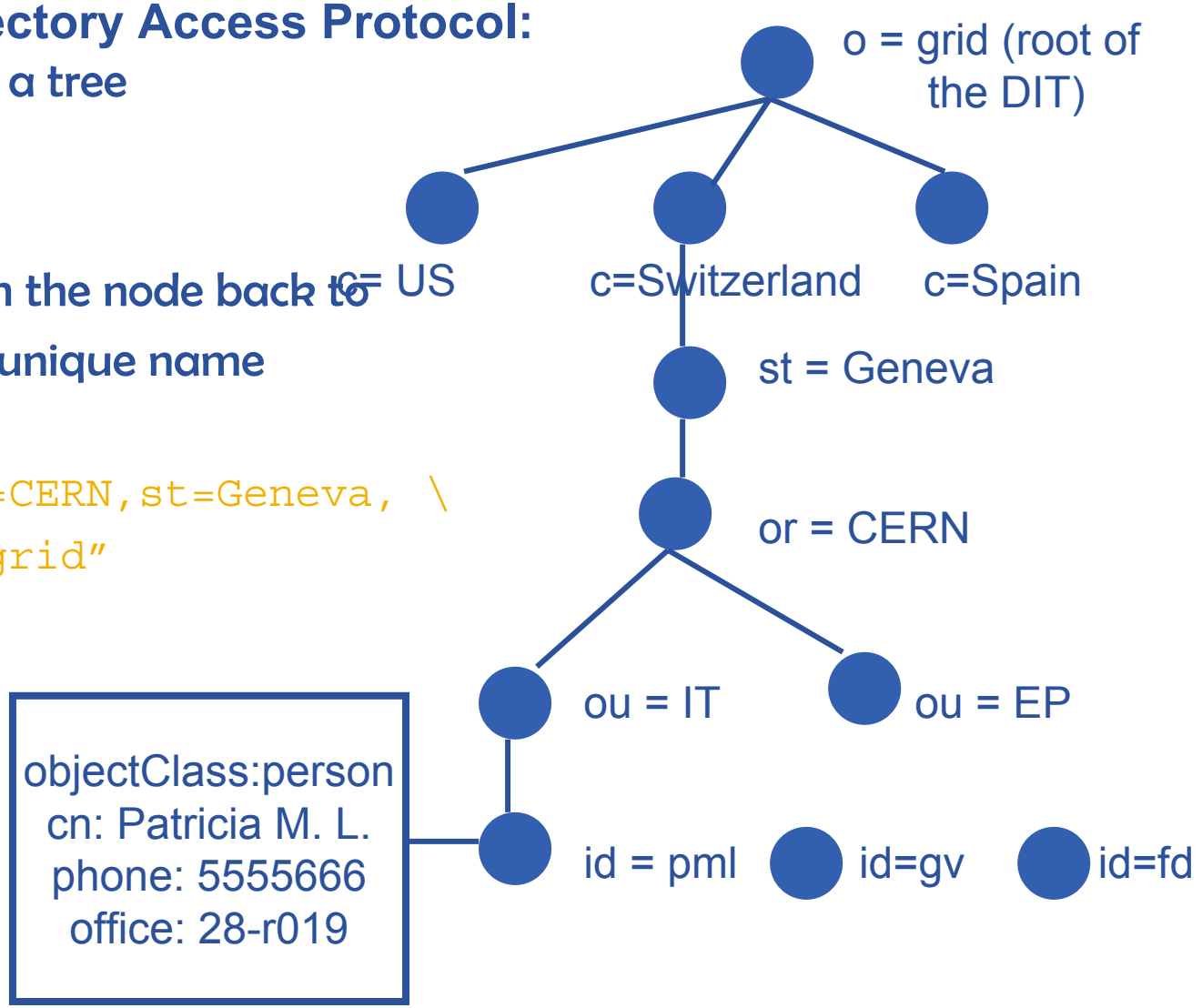


# The LDAP Protocol

- **Lightweight Directory Access Protocol:**
  structures data as a tree

- Following a path from the node back to the root of the DIT, a unique name is built (the DN):

```
"id=pml,ou=IT,or=CERN,st=Geneva, \
c=Switzerland,o=grid"
```



# lcg-infosites

- **The lcg-infosites command can be used as an easy way to retrieve information on Grid resources for most use cases.**

**USAGE: lcg-infosites --vo <vo name> options -v <verbose level> --is <BDII to query>**

- Check if LCG_GFAL_INFOSYS environment variable is correctly set to the local GILDA Information Index (BDII)
- **echo $LCG_GFAL_INFOSYS**
  - **export LCG_GFAL_INFOSYS=grid004.ct.infn.it:2170**

# lcg-infosites options

| | |
|---|---|
| ce | The information related to number of CPUs, running jobs, waiting jobs and names of the CEs are provided. All these data group all VOs together. With "-v 1" only the names of the queues will be printed while with "-v 2" The RAM Memory together with the operating system and its version and the processor included in each CE are printed. |
| se | The names of the SEs supported by the user's VO together with the kind of Storage System,the used and available space will be printed. With "-v 1" only the names of the SEs will be printed. |
| closeSE | The names of the CEs where the user's VO is allowed to run together with their corresponding closest SEs are provided. |
| lfc | Name of the lfc Catalog for the user's VO. |
| tag | The names of the tags relative to the software installed in site is printed together with the corresponding CE. |
| all | It groups together the information provided by ce, se, lrc and rmc. |
| is | If not specified the BDII defined in default by the variable LCG GFAL INFOSYS will be queries. However the user may want to query any other BDII without redefining this environment variable. This is possible specifying this argument followed by the name of the BDII which the user wants to query. All options admits this argument. |

# PRACTICAL: lcg-infosites

- In the next 15 minutes, run the commands shown in following slides to explore GILDA using lcg-infosites.

# Obtaining information about CE

**$ lcg-infosites --vo gilda ce**

```
**********************************************************************
These are the related data for gilda: (in terms of queues and CPUs)
**********************************************************************
#CPU    Free    Total Jobs      Running Waiting ComputingElement
----------------------------------------------------------------------------------
  4       3         0              0       0          cn01.be.itu.edu.tr:2119/jobmanager-lcglsf-long
  4       3         0              0       0          cn01.be.itu.edu.tr:2119/jobmanager-lcglsf-short
 34      33         0              0       0          grid010.ct.infn.it:2119/jobmanager-lcgpbs-long
 16      16         0              0       0          grid011f.cnaf.infn.it:2119/jobmanager-lcgpbs-long
  1       1         0              0       0          grid006.cecalc.ula.ve:2119/jobmanager-lcgpbs-log
  2       1         1              0       1           gildace.oact.inaf.it:2119/jobmanager-lcgpbs-short
[..]
```

**$ lcg-infosites --vo gilda ce --v 2**

```
RAMMemory     Operating System     System Version       Processor          CE Name
-------------------------------------------------------------------------------------------------------------------------
1024                  SLC          3                    P4          ced-ce0.datagrid.cnr.it
4096                  SLC          3                    Xeon          cn01.be.itu.edu.tr
1024                  SLC          3                    PIII          cna02.cna.unicamp.br
 917                  SLC          3                    PIII          gilda-ce-01.pd.infn.it
1024                  SLC          3                    Athlon          gildace.oact.inaf.it
1024                  SLC          3                    Xeon          grid-ce.bio.dist.unige.it
[..]
```

# Obtaining information about SE

```
$ lcg-infosites --vo gilda se

**********************************************************************
These are the related data for gilda: (in terms of SE)
**********************************************************************

Avail Space(Kb) Used Space(Kb)    Type            SEs
----------------------------------------------------------------------------
143547680       2472756           disk             cn02.be.itu.edu.tr
168727984       118549624         disk            grid009.ct.infn.it
13908644        2819288           disk            grid003.cecalc.ula.ve
108741124       2442872           disk            gildase.oact.inaf.it
28211488        2948292           disk            testbed005.cnaf.infn.it
349001680       33028             disk            gilda-se-01.pd.infn.it
31724384        2819596           disk            cna03.cna.unicamp.br
387834656       629136            disk            grid-se.bio.dist.unige.it
```

# Listing the close Storage Elements

**$ lcg-infosites --vo gilda closeSE**

**Name of the CE: cn01.be.itu.edu.tr:2119/jobmanager-lcglsf-long**
**Name of the close SE: cn02.be.itu.edu.tr**

**Name of the CE: cn01.be.itu.edu.tr:2119/jobmanager-lcglsf-short**
**Name of the close SE: cn02.be.itu.edu.tr**

**Name of the CE: grid010.ct.infn.it:2119/jobmanager-lcgpbs-long**
**Name of the close SE: grid009.ct.infn.it**

**Name of the CE: grid011f.cnaf.infn.it:2119/jobmanager-lcgpbs-long**
**Name of the close SE: testbed005.cnaf.infn.it**

- "close" is defined by the CE's manager

# Listing tags of installed software

**$ lcg-infosites --vo gilda tag**

```
**********************************************************************************
Information for gilda relative to their software tags included in each CE
**********************************************************************************

Name of the TAG: VO-gilda-GEANT
Name of the TAG: VO-gilda-GKS05
      Name of the CE:cn01.be.itu.edu.tr

Name of the TAG: VO-gilda-slc3_ia32_gcc323
Name of the TAG: VO-gilda-CMKIN_5_1_1
Name of the TAG: VO-gilda-GEANT
Name of the TAG: VO-gilda-GKS05
      Name of the CE:grid010.ct.infn.it

[..]
```

- VO managers can cause installation of software for their VO onto Worker Nodes of a CE with the agreement of site managers
- A utility can be run to define a tag for each software package installed so these CEs can be identified

# lcg-info intro

- This command can be used to list either CEs or the SEs that satisfy a given set of conditions, and to print the values of a given set of attributes.
- The information is taken from the BDII specified by the LCG_GFAL_INFOSYS environment variable.

- The query syntax is like this:

  attr1 op1 valueN, ...

  attrN opN valueN

After the upgrading of the new GLUE SCHEMA it's not possible use the operator '>' and '<'

where *attrN* is an attribute name

op is =, >= or <=, and the cuts are ANDed.

The cuts are comma-separated and spaces are not allowed.

# lcg-info usage

## USAGE

**lcg-info --list-ce [--bdii bdii] [--vo vo] [--sed] [--query query] [--attrs list]**

**lcg-info --list-se [--bdii bdii] [--vo vo] [--sed] [--query query] [--attrs list]**

**lcg-info --list-attrs**

**lcg-info --help**

# lcg-info options

| | |
|---|---|
| --list-attrs | Prints a list of the attributes that can be queried. |
| --list-ce | Lists the CEs which satisfy a query, or all the CEs if no query is given. |
| --list-se | Lists the SEs which satisfy a query, or all the SEs if no query is given. |
| --query | Restricts the output to the CEs (SEs) which satisfy the given query. |
| --bdii | Allows to specify a BDII in the form :. If not given, the value of the environmental variable LCG_GFAL_INFOSYS is used. If that is not defined, the command returns an error. |
| --sed | Print the output in a "sed-friendly" format. |
| --attrs | Specifies the attributes whose values should be printed. |
| --vo | Restricts the output to CEs or SEs where the given VO is authorized. Mandatory when VO-dependent attributes are queried upon. |

# Get the list of supported attributes

```
$ lcg-info --list-attrs
```

| Attribute name | Glue object class | Glue attribute name |
|---|---|---|
| MaxTime | GlueCE | GlueCEPolicyMaxWallClockTime |
| CEStatus | GlueCE | GlueCEStateStatus |
| TotalJobs | GlueCE | GlueCEStateTotalJobs |
| CEVOs | GlueCE | GlueCEAccessControlBaseRule |
| TotalCPUs | GlueCE | GlueCEInfoTotalCPUs |
| FreeCPUs | GlueCE | GlueCEStateFreeCPUs |
| CE | GlueCE | GlueCEUniqueID |
| WaitingJobs | GlueCE | GlueCEStateWaitingJobs |
| RunningJobs | GlueCE | GlueCEStateRunningJobs |
| CloseCE | GlueCESEBindGroup | GlueCESEBindGroupCEUniqueID |
| CloseSE | GlueCESEBindGroup | GlueCESEBindGroupSEUniqueID |
| SEVOs | GlueSA | GlueSAAccessControlBaseRule |
| UsedSpace | GlueSA | GlueSAStateUsedSpace |
| AvailableSpace | GlueSA | GlueSAStateAvailableSpace |
| Type | GlueSE | GlueSEType |
| SE | GlueSE | GlueSEUniqueID |
| Protocol | GlueSEAccessProtocol | GlueSEAccessProtocolType |
| ArchType | GlueSL | GlueSLArchitectureType |
| Processor | GlueSubCluster | GlueHostProcessorModel |
| OS | GlueSubCluster | GlueHostOperatingSystemName |
| Cluster | GlueSubCluster | GlueSubClusterUniqueID |
| Tag | GlueSubCluster | GlueHostApplicationSoftwareRunTimeEnvironment |
| Memory | GlueSubCluster | GlueHostMainMemoryRAMSize |

# lcg-info examples

**List all the CE(s) that can run MPICH, giving the number of free CPUs and the tags of installed software**

```
$ lcg-info --vo gilda --list-ce --query 'Tag=MPICH' --attrs 'FreeCPUs,Tag'
```

- Careful here!
- No space allowed here!

```
-…..
 CE: grid-ce.bio.dist.unige.it:2119/jobmanager-lcgpbs-long
  - FreeCPUs          6
  - Tag               LCG-2
                      LCG-2_1_0
                      LCG-2_1_1
….
```

# lcg-info examples

## List all the CE(s) in the BDII satisfying given conditions

**$ lcg-info --list-ce --query 'FreeCPUs=2' --attrs 'FreeCPUs,OS'**

```
- - CE: gildace.oact.inaf.it:2119/jobmanager-lcgpbs-infinite
  - FreeCPUs          2
  - OS                SLC

- CE: gildace.oact.inaf.it:2119/jobmanager-lcgpbs-long
  - FreeCPUs          2
  - OS                SLC

- CE: gildace.oact.inaf.it:2119/jobmanager-lcgpbs-short
  - FreeCPUs          2
  - OS                SLC

- CE: trigrid-ce00.unime.it:2119/jobmanager-lcgpbs-infinite
  - FreeCPUs          2
  - OS                _UNDEF_
[..]
```

# lcg-info examples (cont.)

## List all the CE(s) which satisfying the condition FreeCPU >=30

```
$ lcg-info --list-ce --query 'FreeCPUs >= 30' --attrs 'FreeCPUs'
```

- fails

```
- CE: grid010.ct.infn.it:2119/jobmanager-lcgpbs-long
  - FreeCPUs          33

- CE: grid010.ct.infn.it:2119/jobmanager-lcgpbs-short
  - FreeCPUs          33

- CE: grid010.ct.infn.it:2119/jobmanager-lcgpbs-infinite
  - FreeCPUs          33

[..]
```

# Print all the tags published by a specific query

```
$ lcg-info --list-ce --query 'CE=*grid010.ct.infn.it:2119*' --attrs 'Tag'
```

```
 PBS
INFN
CATANIA
LCG-2
LCG-2_1_0
LCG-2_1_1
LCG-2_2_0
LCG-2_3_0
LCG-2_3_1
LCG-2_4_0
R-GMA
AFS
CMS-1.1.0
ATLAS-6.0.4
GATE-1.0.0-3
LHCb-1.1.1
IDL-5.4
CMSIM-125
ALICE-4.01.00
ALIEN-1.32.14
POVRAY-3.5
DEMTOOLS-1.0
CMKIN-VALID
CMKIN-1.1.0
CMSIM-VALID
CSOUND-4.13
MPICH
VIRGO-1.0
CMS-OSCAR-2.4.5
LHCb_dbase_common-v3r1
GEANT4-6
VLC-0.7.2
EGEODE-1.0
RASTER3D
SCILAB-2.6
G95-3.5.0
MAGIC-6.19
CODESA3D-1.0
VO-gilda-slc3_ia32_gcc323
VO-gilda-CMKIN_5_1_1
VO-gilda-GEANT
VO-gilda-GKS05
```

# List the SEs satisfying given query

**$ lcg-info -vo gilda --list-se --query 'AvailableSpace>=100000' --attrs 'CloseCE'**

```
- SE: cn02.be.itu.edu.tr
  - CloseCE          cn01.be.itu.edu.tr:2119/jobmanager-lcglsf-long
                     cn01.be.itu.edu.tr:2119/jobmanager-lcglsf-short
                     cn01.be.itu.edu.tr:2119/jobmanager-lcglsf-infinite

- SE: grid009.ct.infn.it
  - CloseCE          grid010.ct.infn.it:2119/jobmanager-lcgpbs-long
                     grid010.ct.infn.it:2119/jobmanager-lcgpbs-short
                     grid010.ct.infn.it:2119/jobmanager-lcgpbs-infinite

- SE: ced-se0.datagrid.cnr.it
  - CloseCE          ced-ce0.datagrid.cnr.it:2119/jobmanager-lcgpbs-long
                     ced-ce0.datagrid.cnr.it:2119/jobmanager-lcgpbs-short
                     ced-ce0.datagrid.cnr.it:2119/jobmanager-lcgpbs-infinite

- SE: grid003.cecablc.ula.ve
  - CloseCE          grid006.cecalc.ula.ve:2119/jobmanager-lcgpbs-cert
                     grid006.cecalc.ula.ve:2119/jobmanager-lcgpbs-long
                     grid006.cecalc.ula.ve:2119/jobmanager-lcgpbs-short
                     grid006.cecalc.ula.ve:2119/jobmanager-lcgpbs-infinite

[..]
```

# Data Management

# Scope of data services

- **Files that are write-once, read-many**
- **Files are replicated to be**
  - "Close" to compute elements for efficiency
  - Resilient to SE failure

# DM Overview

- **Two sets of commands**
- **LFC = LCG File Catalogue**
  - LCG = LHC Compute Grid
  - LHC = Large Hadron Collider
  - Use LFC commands to interact with the catalogue only
    - To create catalogue directory
    - List files
  - Used by you and by lcg-utils
- **lcg-utils**
  - File management functions
  - Couples file upload, replication … and catalog operations
  - Keeps SEs and catalogue in step!
- **(also GFAL functions to read blocks from files on SE's… can't always copy files to a worker node!)**

# LFC basics

## LFC has a directory tree structure

/grid/<VO_name>/ <you create it>

LFC Namespace

Defined by the user

- All members of a given VO have read-write permissions in their directory
- Commands look like UNIX with “lfc-” in front (often)

# Check your environment

- **Check / set the following environment variables to specify the catalog type and its location:**

  To check:

  **echo $LCG_CATALOG_TYPE** should be lfc

  **echo $LFC_HOST** should be lfc-gilda.ct.infn.it

  To set:

  **export LCG_CATALOG_TYPE=lfc**

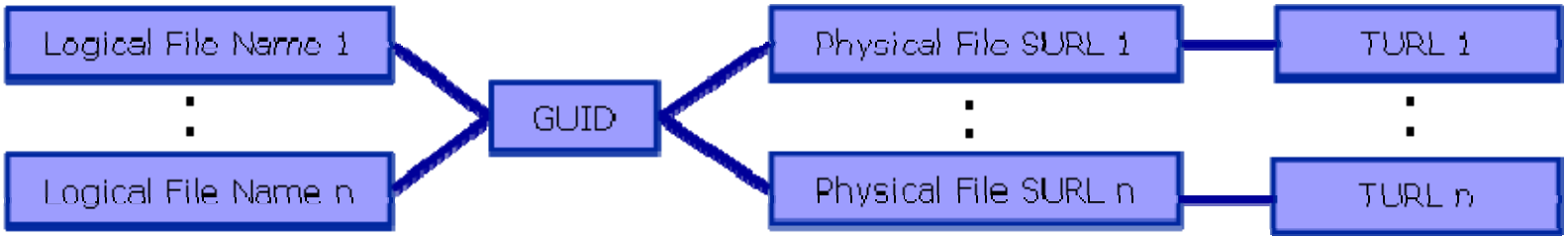
  **export LFC_HOST=lfc-gilda.ct.infn.it**

- **Ensure you have created a proxy certificate and it is still valid.**
  - voms-proxy-info –all. Look at both “timeleft” lines.
  - If run out
    - voms-proxy-destroy)
  - To create it: **`voms-proxy-init --voms gilda`**

    Remember: The Passphrase is TAIPEI

# Name conventions

- **Logical File Name (LFN)**
  - An alias created by a user to refer to some item of data, e.g. "lfn:cms/20030203/run2/track1"
- **Globally Unique Identifier (GUID)**
  - A non-human-readable unique identifier for an item of data, e.g. "guid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
- **Site URL (SURL) (or Physical File Name (PFN) or Site FN)**
  - The location of an actual piece of data on a storage system, e.g.
    "srm://pcrd24.cern.ch/flatfiles/cms/output10_1" (SRM)
    "sfn://lxshare0209.cern.ch/data/alice/ntuples.dat" (Classic SE)
- **Transport URL (TURL)**
  - Temporary locator of a replica + access protocol: understood by a SE, e.g.
    "rfio://lxshare0209.cern.ch//data/alice/ntuples.dat"

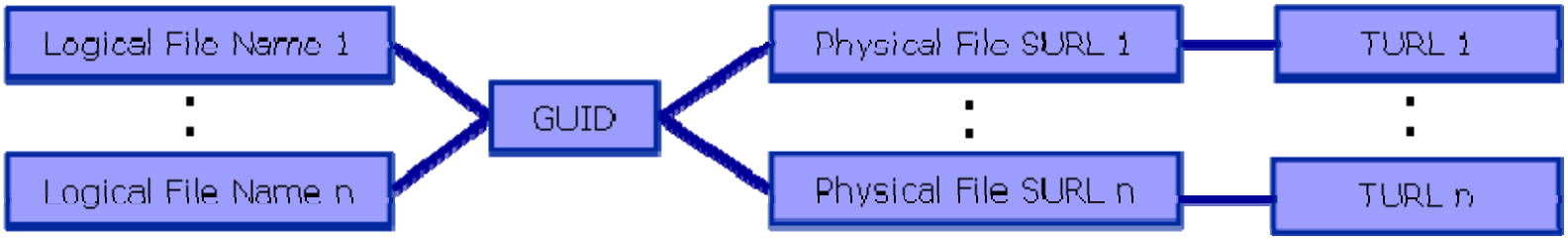


# We are about to…

- List directory
- Upload a file to an SE and register a logical name (lfn) in the catalog
- Create a duplicate in another SE
- List the replicas

- Create a second logical file name for a file
- Download a file from an SE to the UI

- And then: Use the lfn so that a job runs on a CE “close” to one of the SEs that holds a file



# lcg-utils

- The LCG Data Management tools (usually called ***lcg-utils***) allow users to copy files between UI, CE, WN and a SE, to register entries in the File Catalogs and replicate files between SEs.

- Check if LCG_GFAL_INFOSYS environment variable is correctly set to the local GILDA Information Index (BDII)

  **echo $LCG_GFAL_INFOSYS**

  **It should be grid004.ct.infn.it:2170 otherwise set it :**

  **export LCG_GFAL_INFOSYS=grid004.ct.infn.it:2170**

# Listing a directory

## Listing the entries of a LFC directory : lfc-ls

*lfc-ls [-cdiLlRTu] [--comment] path…*

where *path* specifies the LFC pathname (mandatory)

- ***-l*** (it is a lowercase "L") outputs long listing
- ***-R*** lists the contents of directories recursively (don't use it AT ALL)

Try it!

```
$ lfc-ls -l /grid/gilda/taipei
```

```
...
drwxrwxr-x   0 4461     4400                0 Mar 09 21:00 taipei49
drwxrwxr-x   0 4461     4400                0 Mar 09 20:48 mjm
```

# Setting LFC_HOME

- LFC_HOME to use relative paths

**Now SET LFC_HOME as follows:**

```
$ export LFC_HOME=/grid/gilda/taipei/
```

**Then try the equivalent of the lfc-ls you just did:**

```
$ lfc-ls -l
This is now the same as
 lfc-ls -l /grid/gilda/taipei
```

# lcg-utils: lfc-mkdir, lcg-cr

## Upload a file to a SE and register it into the catalog

- lcg-cr -d dest_file | dest_host -l lfn [-g guid] [-l lfn]
  [-v | --verbose] --vo vo src_file

where

- ***dest_host*** is the fully qualified hostname of the destination SE
- ***(dest_file*** is a valid SURL (both sfn:// or srm:// format are valid) )
- ***guid*** specifies the Grid Unique IDentifier. If this option is not present, a GUID is generated internally
- ***lfn*** specifies the Logical File Name associated with the file
- ***vo*** specifies the Virtual Organization the user belongs to
- ***src_file*** specifies the source file name: the protocol can be *file:///* or *gsiftp:///*

**ls –l .. > aNewFile.txt**

**lfc-mkdir taipei**XX

**lcg-cr --vo gilda file://`pwd`/aNewFile.txt -l lfn:taipei**XX**/my.dat -d grid009.ct.infn.it**

**guid:01843d11-57c4-4a0e-80af-a110a6287552**

# Reminder

- **To discover which SEs you can use :**

  **`lcg-infosites --vo gilda closeSE`**

  The output is a list of SEs and related information on available/used space

- Problems? Check if LCG_GFAL_INFOSYS environment variable is correctly set to the local GILDA Information Index (BDII)

  **echo $LCG_GFAL_INFOSYS**

  **export LCG_GFAL_INFOSYS=grid004.ct.infn.it:2170**

# Replicate a file

## Copying a file from one SE to another one and register it in the Catalog

lcg-rep -d dest_file | dest_host [-v | --verbose] --vo vo src_file

where

- ***dest_host*** is the fully qualified hostname of the destination SE
- ***dest_file*** is a valid SURL (both sfn:// or srm:// are valid)
- ***vo*** specifies the Virtual Organization the user belongs to
- ***src_file*** specifies the source file name: the protocol can be LFN, GUID or SURL. An SURL scheme can be sfn: for a classical SE or srm:

**lcg-rep --vo gilda -d gildase.oact.inaf.it **
**lfn:taipeiXX/my.dat**

# List replicas

## Listing of replicas for a given LFN, GUID or SURL

**lcg-lr --vo vo file**

where

- ***vo*** specifies the Virtual Organization the user belongs to
- **file** specifies the Logical File Name, the Grid Unique IDentifier or the Site URL. An SURL scheme can be sfn: for a classical SE or srm:

- **Example:**

**$ lcg-lr --vo gilda lfn:taipeiXX/my.dat**

# Create duplicate lfn

## Creating a duplicate logical file name (does not create a new physical file!)

***lfc-ln -s file linkname***

***lfc-ln -s directory linkname***

Create a link to the specified *file* or *directory* with *linkname*

– *Do this command please:*

*$ lfc-ln -s *

*/grid/gilda/taipei/mjm/scriptDictionary.tar *

*/grid/gilda/taipei/taipeiXX/script.tar*

**New lfn**

**Original lfn**

Let's check the link using lfc-ls with long listing (-l)

*$ lfc-ls -l /grid/gilda/taipei/taipeiXX*

```
… script.tar -> /grid/gilda/taipei/mjm/scriptDictionary.tar
```

# From SE to the UI or a Worker: lcg-cp

## Downloading a Grid file from a SE to a local destination

**lcg-cp [ -v | --verbose ] --vo vo src_file dest_file**

where

- ***vo*** specifies the Virtual Organization the user belongs to
- ***src_file*** specifies the source file name: the protocol can be LFN, GUID, SURL or local file. An SURL scheme can be sfn: for a classical SE or srm:
- ***dest_file*** specifies the destination. Example:

```
$ lcg-cp --vo gilda lfn:taipeiXX/my.dat
  file://`pwd`/<mylocalfilename>.txt
```

# LFC Catalog commands

## Adding/deleting metadata information

***lfc-setcomment path comment***

***lfc-delcomment path***

taipei: SKIP THIS

lfc-setcomment adds/replaces a *comment* associated with a file/directory in the LFC Catalog

lfc-delcomment deletes a comment previously added

- Example:

  **lfc-setcomment taipei**XX**/my.dat "Hello Taipei"**

- Check your job with..

  **lfc-ls --comment taipei**XX**/my.dat**

# LFC Catalog commands

taipei: SKIP THIS

- Example:

**lfc-delcomment /grid/gilda/user.example**

- Check your job with..

**lfc-ls –l --comment /grid/gilda/user.example**

```
-rw-rw-r--   1 4401     4400            0 Jun 21 09:38 /grid/gilda/user.example
```

# LFC Catalog commands

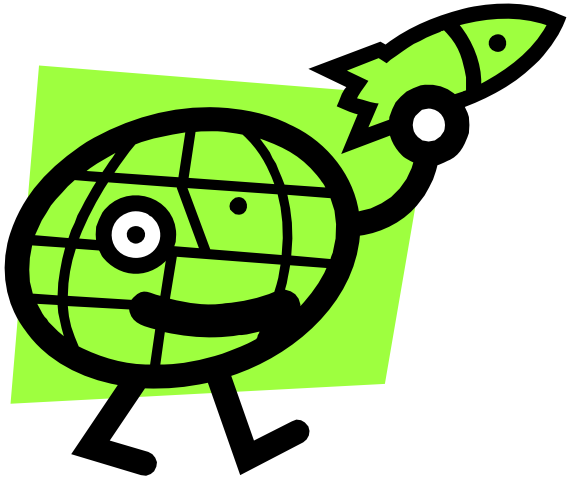
## Summary of the LFC Catalog commands

| | |
|---|---|
| lfc-chmod | Change access mode of the LFC file/directory |
| lfc-chown | Change owner and group of the LFC file-directory |
| lfc-delcomment | Delete the comment associated with the file/directory |
| lfc-getacl | Get file/directory access control lists |
| lfc-ln | Make a symbolic link to a file/directory |
| lfc-ls | List file/directory entries in a directory |
| lfc-mkdir | Create a directory |
| lfc-rename | Rename a file/directory |
| lfc-rm | Remove a file/directory |
| lfc-setacl | Set file/directory access control lists |
| lfc-setcomment | Add/replace a comment |

# Summary of lcg-utils commands

## Replica Management

| | |
|---|---|
| lcg-cp | Copies a grid file to a local destination |
| lcg-cr | Copies a file to a SE and registers the file in the catalog |
| lcg-del | Delete one file |
| lcg-rep | Replication between SEs and registration of the replica |
| lcg-gt | Gets the TURL for a given SURL and transfer protocol |
| lcg-sd | Sets file status to “Done” for a given SURL in a SRM request |



# Workload Management System
# More realistic examples

1. Job thats writes results to a SE
2. Scripting to run multiple jobs
3. Running job “close” to SE with required input data

# Grid Training
# for the MAGIC Grid
# How To submit Corsika?

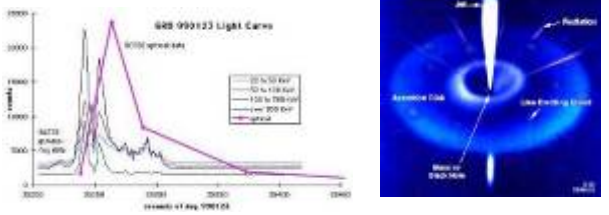
Harald Kornmayer

IWR, Forschungszentrum Karlsruhe

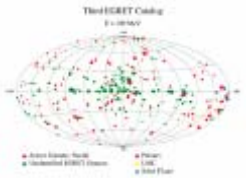
in cooperation with EGEE Training group (NA3)

# MAGIC

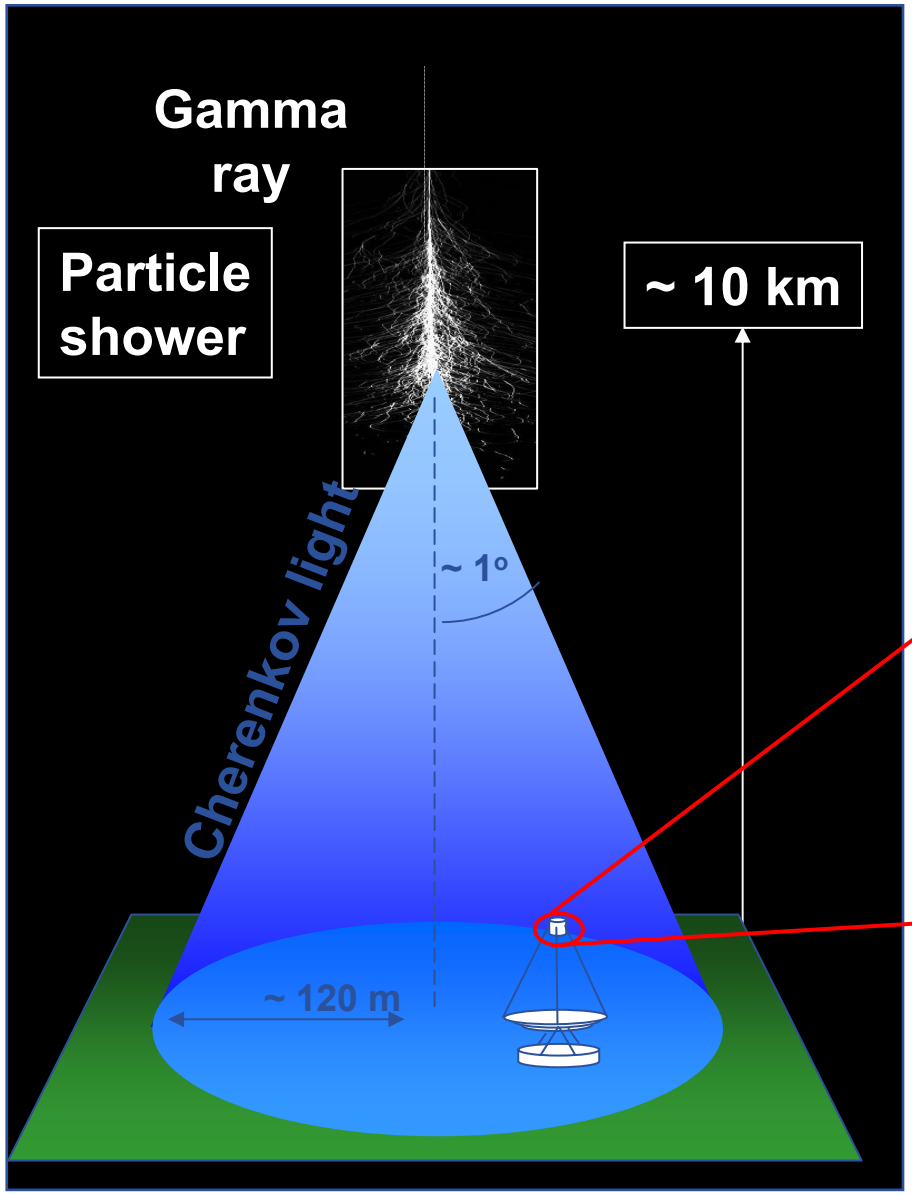
- **Ground based Air Cerenkov Telescope 17 m diameter**
- **Physics Goals:**
  - Origin of VHE Gamma rays
  - Active Galactic Nuclei
  - Supernova Remnants
  - Unidentified EGRET sources
  - Gamma Ray Burst
- **MAGIC II will come 2007**
- **Grid added value**
  - Enable “(e-)scientific” collaboration between partners
  - Enable the cooperation between different experiments
  - Enable the participation on Virtual Observatories



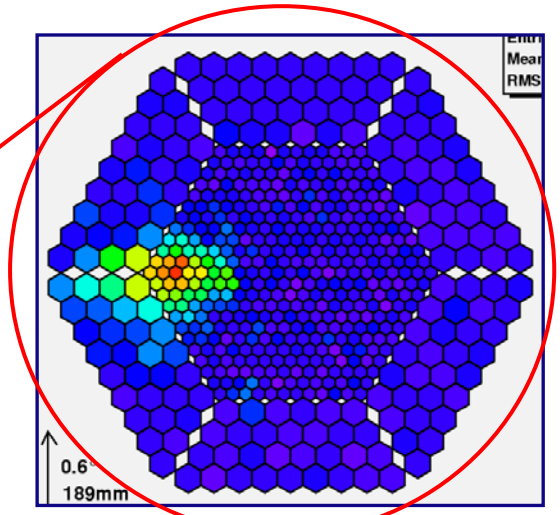




# Ground based γ-ray astronomy

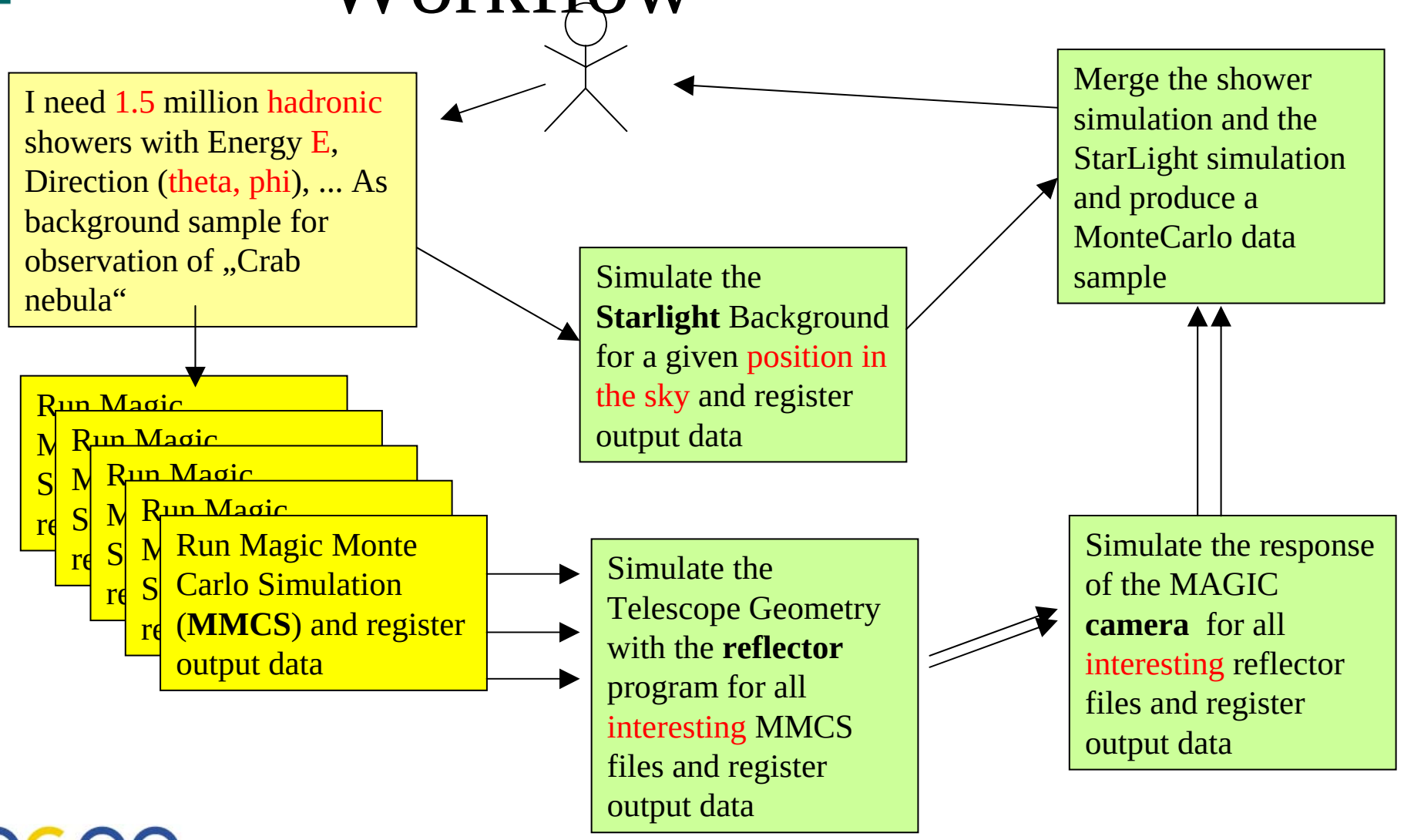



Cherenkov light Image of particle shower in telescope camera



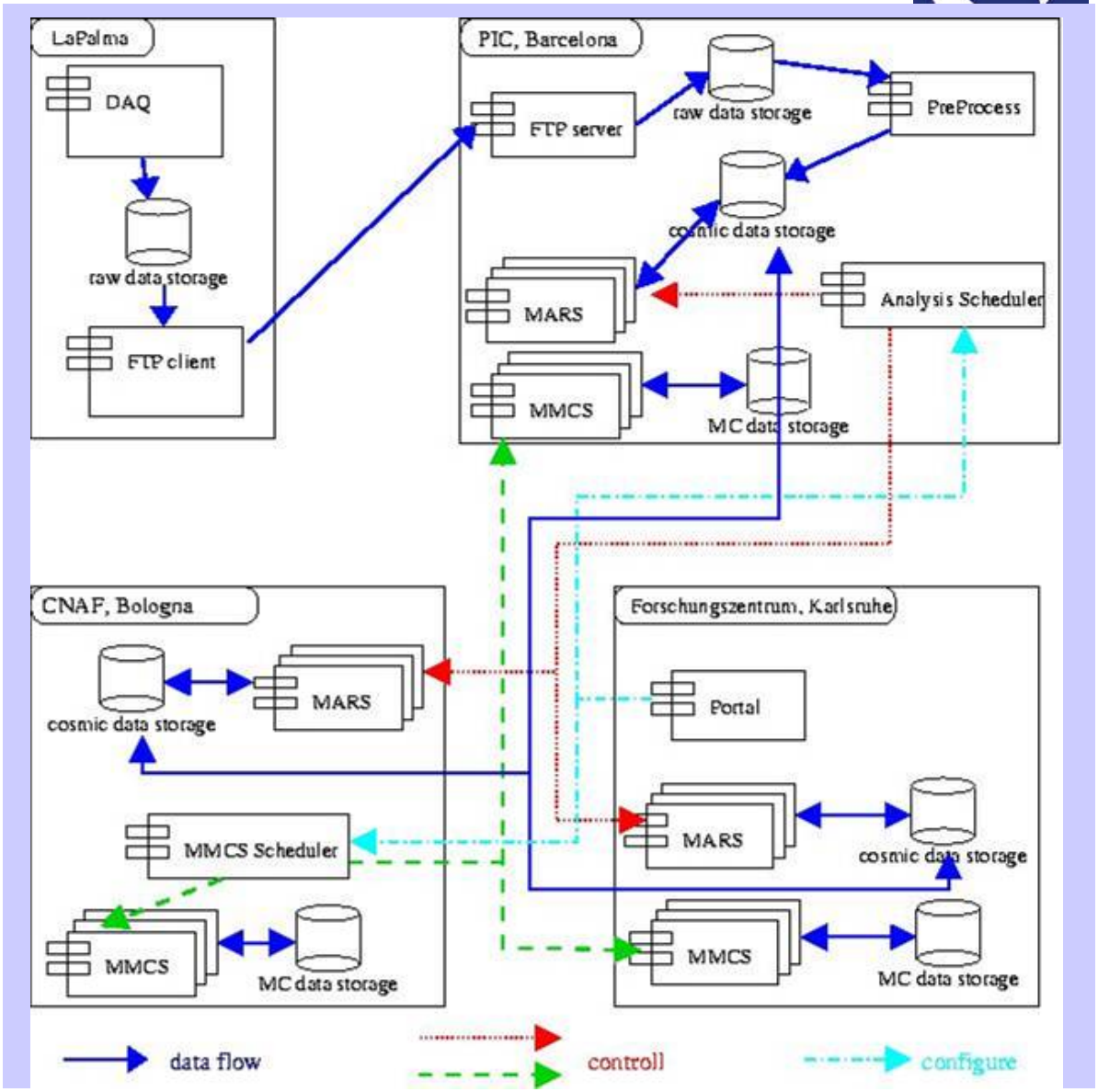

reconstruct:
arrival direction, energy
reject hadron background

# MAGIC Monte Carlo Workflow



# MAGIC Grid – the idea

- Build a Grid system with
  - FZK (Germany)
  - CNAF(Italy)
  - PIC (Spain)
- MAGIC applied as a generic application for EGEE
- MAGIC got accepted with the air shower Monte Carlo simulation based on CORSIKA



# A MAGIC practical

- Run one of the CORSIKA simulations.
- We will:
  - Obtain tar file from an SE
  - Inspect the JDL
    - How it uses sandboxes to transfer files
    - How it sets executable flag
  - Modify the jdl
  - Submit the job
  - Explore the output, jdl and script used

# How to keep the CORSIKA output?

- **To keep the data on the Grid**
  - important for big files!
  - so others can acess them
- **Amend the JDL to define your lfn and select SE**
  - Use full path name
  - Use info system to choose an SE (or one you used earlier!)

```
Executable = "registerCorsika.sh" ;
....
....
OutputSandbox = {"registerCorsika.out",
"registerCorsika.err"};
OutputData={
[
  Outputfile = "./cer000001";
  LogicalFileName =
"lfn:mmcs_cer000001";
  StorageElement = "castorgrid.pic.es";
],
[
  Outputfile = "./dat000001";
  LogicalFileName =
"lfn:mmcs_dat000001";
  StorageElement = "castorgrid.pic.es";
]
```

# Exercise

- **The tar file has the logical filename: lfn:/grid/gilda/taipei/mjm/magic.tar**

- **Copy it to your "myfiles" directory**
  - mkdir /home/taipeixx/myfiles/magic
  - cd /home/taipeixx/myfiles/magic
  - lcg-?? – which command do you need to use?
  - tar -xvf magic.tar to expand the directory
- **Amend the .jdl to**
  - write result files to an SE and register those files in your namespace in the LFC
  - use a short queue
- **Submit the job, saving the id in a file**

# MAGIC exercise continued

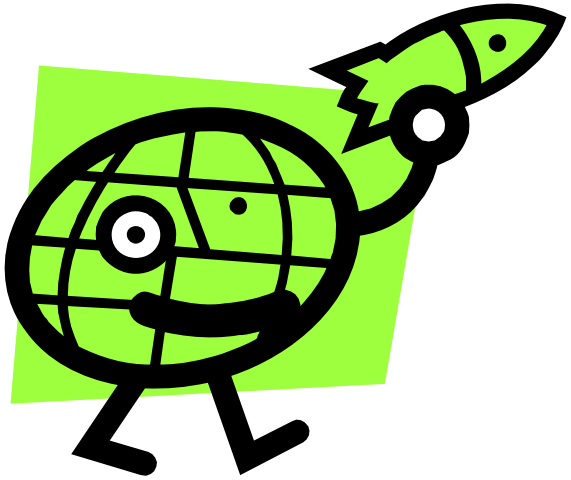
- When the job is submitted, go on to the next exercise whilst you wait for it to run.
- Once it has completed, retrieve the output and step through it with the jdl and the script

# How to execute CORSIKA on the Grid?

- **Notice from the *ls -l* listed in the output that it is necessary to set the execute flag on the file.**

```
Executable = „executeCorsika.sh" ;
StdOutput  = „executeCorsika.out";
StdError   = „executeCorsika.err";
InputSandbox  = {
"executeCorsika.sh",
"corsika/cc6023p-linux",
"corsika/EGSDAT3_.05",
"corsika/EGSDAT3_.15",
"corsika/EGSDAT3_.4",
"corsika/EGSDAT3_1.",
"corsika/EGSDAT3_3.",
"corsika/NUCLEAR.BIN",
"corsika/NUCNUCCS",
"corsika/VENUSDAT",
"corsika/atmprof1.dat",
"corsika/atmprof2.dat",
"corsika/atmprof3.dat",
"corsika/atmprof4.dat",
"corsika/atmprof5.dat",
"corsika/atmprof6.dat",
"corsika/atmprof9.dat",
"input_card"
};
OutputSandbox = {„executeCorsika.out",
„executeCorsika.err"};
RetryCount = 5;
```

```
echo "Execution of Corsika on the Grid"

echo " started on Host"
hostname

echo " "
echo "   Content of working directory"
ls -l

echo " "
echo " Change the rights of executable "
chmod u+x cc6023p-linux
ls -l cc6023p-linux

echo " "
echo "   Start the job "
echo "cc6023p-linux < input_card"
./cc6023p-linux  < input_card

echo " "
echo "   Content of ./data"
ls cer* dat*

echo " "
echo "Finished "
```



# Workload Management System

## More realistic examples

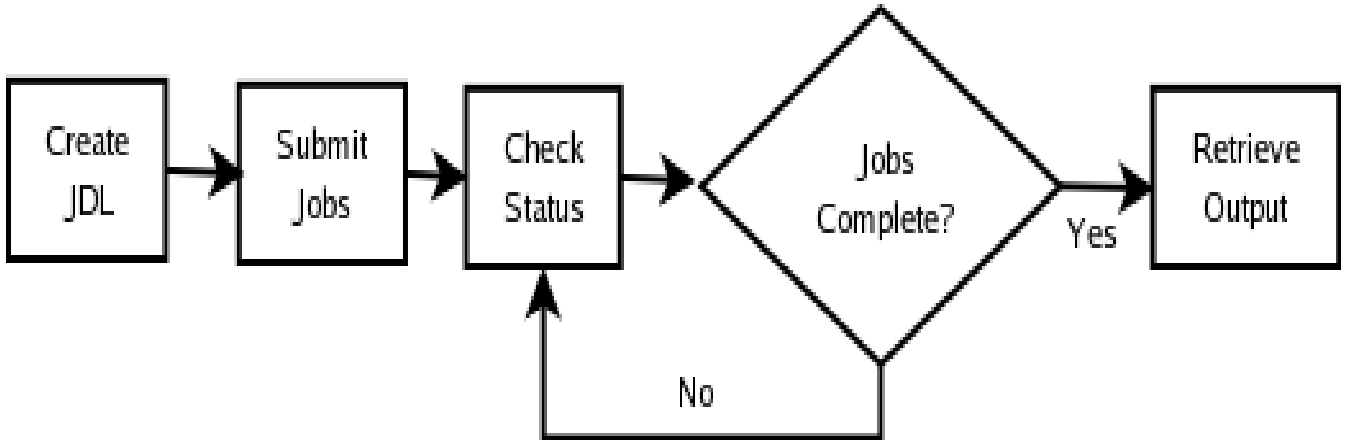
1. Job thats writes results to a SE
2. Scripting to run multiple jobs

# A scripting example

- A common requirement is to run many concurrent jobs.
- This example gives you a pattern for this.

# A scripting example

- **We have seen that, to run a job on the grid**
  - Create a JDL file
  - Submit job
  - Check the jobs status until it is complete
  - Retrieve output
- **This process can be automated**



# Description

- **submit-dictionary-jobs.sh**
  - submits a cascade of simple jobs, each with the same executable but different arguments
  - called with one argument n, the number of jobs to submit
  - gets random dictionary words and creates n jdl files with those words as parameters to the script echoword.sh
  - echoword.sh simply echoes the word back to stdout
  - submits each job
  - waits for all jobs to complete by running edg-job-status -i jobidfile and parsing the output
  - when all jobs have completed it retrieves the n output files
  - finally it concatenates the output from each output file into the one results file and echoes that to the screen

# Exercise-part 2

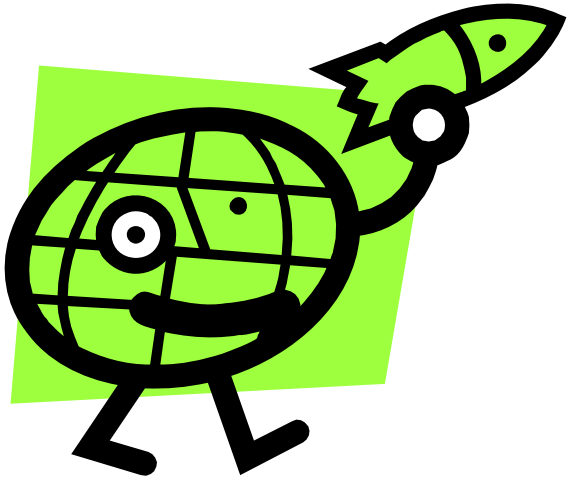
- You have already created a logical filename /grid/gilda/taipei/taipeixx/script.tar (slide 85)

- Download that file using lcg-utils - command lcg-??
- Untar it into its own directory to see two script files

- Open a second window onto GILDA

- Run the script
- ./submit-dictionary-jobs.sh 4
- [do not use more than 4 please!]

- Whilst it is running explore the script in the second window.
- [Then check completion of any previous jobs]



# Workload Management System
# More realistic examples

1. Job thats writes results to a SE
2. Scripting to run multiple jobs
3. Running job “close” to SE with required input data

# Exercise

- **GOAL:**

  **Submit a job that does data management: it will retrieve a file previously registered into the catalog.**

- **Steps to follow up:**
  - Remember the lfn of a file you entered earlier: lfc-ls will help!
  - create a script.sh file with the following content:

```
#!/bin/sh
/bin/hostname
#Change the LFN_NAME to download from the Catalog.
echo "Start to download.."
lcg-cp --vo gilda lfn:<lfn you choose> file:`pwd`/output.dat
echo "Done.."
```

# exercise (II)

- Create the JobWithData.jdl:

```
Type = "job";
JobType = "Normal";
Executable = "/bin/sh";
Arguments = "script.sh";
InputData={"lfn:<your file>"};
DataAccessProtocol={"gsiftp"};
VirtualOrganisation = "gilda";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {"script.sh"};
OutputSandbox = {"std.out","std.err","output.dat"};
```

- Tells RB that you want to run close to this.
- Does not retrieve the file…it might be HUGE!!

- Submit it to the grid
- Retrieve the output and verify the content of output.dat

# Housekeeping

- **At the end of the practical**
- **Destroy your proxy certificate: voms-proxy-destroy**
  - Always do this when you've finished
  - [Usually you would remove files in .globus also, but today don't]
- **Please delete all the files you created on SE's by using the lfc-ls command to find them in $LFC_HOME/taipeiXX (refer to next slide)…**
- **1. delete the lfn /grid/gilda/taipei/taipeixx/script.tar**
- **2. delete all files you uploaded and filenames you registered**

# Deleting replicas

## Deleting replicas

- **lcg-del [ -a ] [ -s se ] [ -v | --verbose ] --vo vo file**

where

- ***a*** is used to delete all replicas of the given file
- ***se*** specifies the SE from which you want to remove the replica
- ***vo*** specifies the Virtual Organization the user belongs to
- ***file*** specifies the Logical File Name, the Grid Unique IDentifier or the Site URL. An SURL scheme can be sfn: for a classical SE or srm:.

**Example:**

- delete one replica

```
$ lcg-del --vo gilda -s grid009.ct.infn.it lfn:<name>
```

- delete all the replicas

```
$ lcg-del -a --vo gilda lfn:<name>
```

- let's check if the previous command was successful

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

lcg_lr: No such file or directory

# Monitoring

- **Two examples of monitoring systems**
- **http://gridportal.hep.ph.ic.ac.uk/rtm/**
- **http://infnforge.cnaf.infn.it/gridice/index.php/Main/GridICEWork**
  - Select a URL for GILDA
    - VO view (menu)
    - Select GILDA (column, far left)
    - Charts
  - Try also for LCG

# R-GMA

# Introduction to R-GMA

- **Relational Grid Monitoring Architecture (R-GMA)**
  - Developed as part of the EuropeanDataGrid Project (EDG)
  - Now as part of the EGEE project.
  - Based the Grid Monitoring Architecture (GMA)

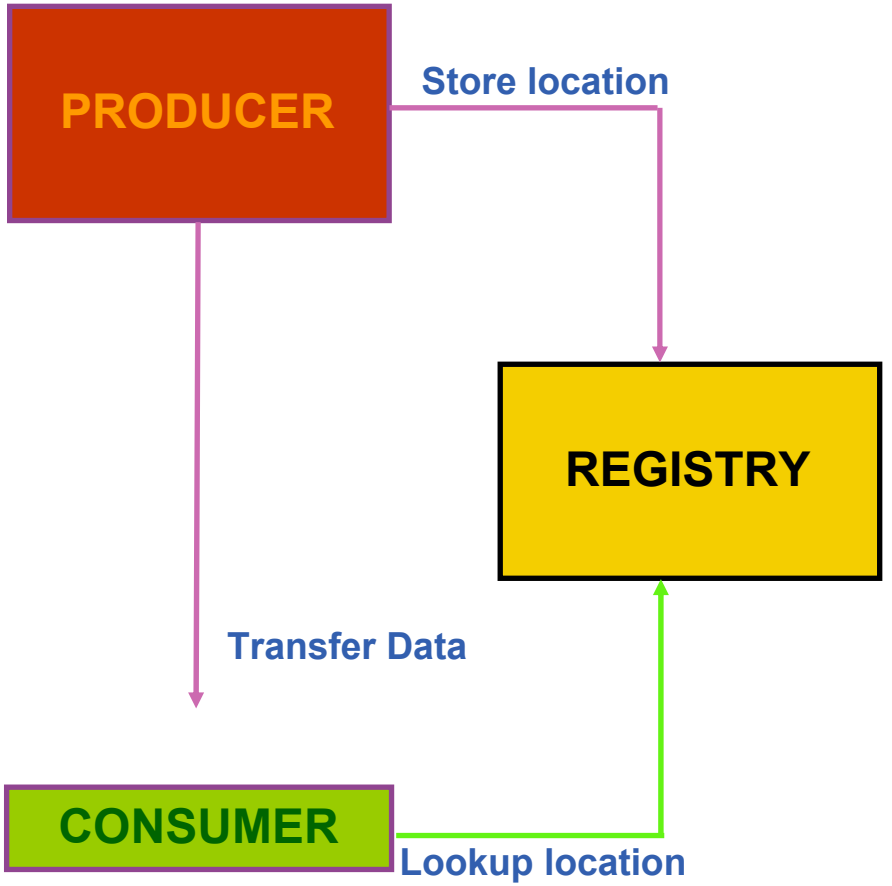
- **Uses a relational data model.**
  - Data are viewed as a table.
  - Data structure defined by the columns.
  - Each entry is a row (tuple).
  - Queried using Structured Query Language (SQL).

| name | ID | birth | Group |
|---|---|---|---|
| Tom | 4 | 1977-08-20 | HR |

SELECT * FROM people WHERE group='HR'

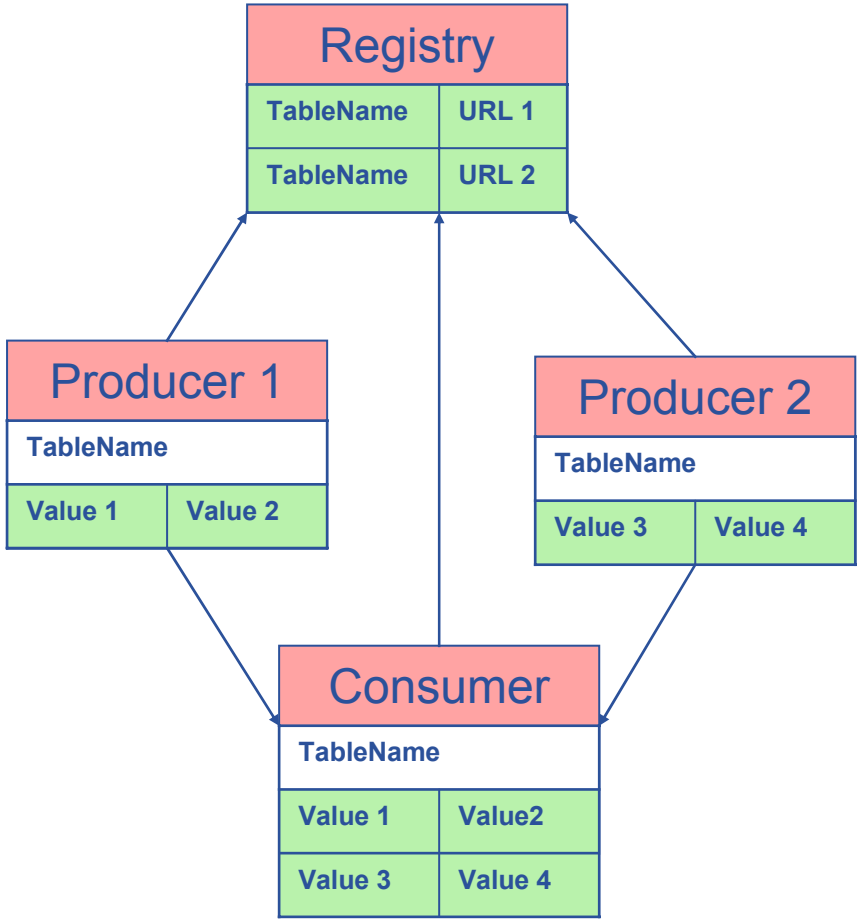
# Service orientation

- The Producer stores its location (URL) in the Registry.
- The Consumer looks up producer URLs in the Registry.
- The Consumer contacts the Producer to get all the data or the Consumer can listen to the Producer for new data.



# Virtual database

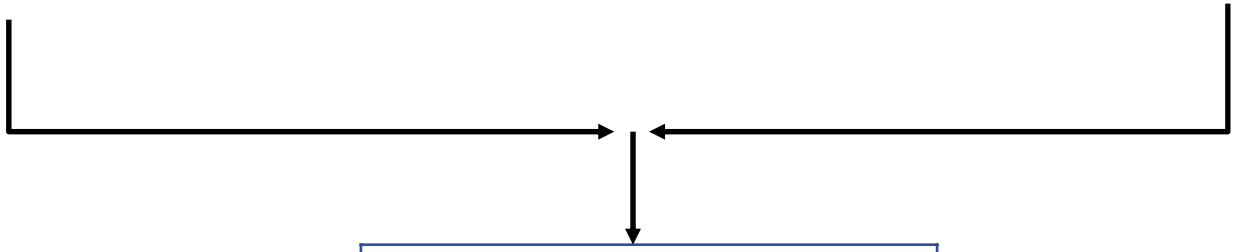
- The Consumer interrogates the Registry to identify all Producers that could satisfy the query.
- Consumer connects to the Producers.
- Producers send the tuples to the Consumer.
- The Consumer will merge these tuples to form one result set.



# Joins

| Service | | | | |
|---|---|---|---|---|
| URI | VO | type | emailContact | site |
| gppse01 | alice | SE | sysad@rl.ac.uk | RAL |
| gppse01 | atlas | SE | sysad@rl.ac.uk | RAL |
| gppse02 | cms | SE | sysad@rl.ac.uk | RAL |
| lxshare0404 | alice | SE | sysad@cern.ch | CERN |
| lxshare0404 | atlas | SE | sysad@cern.ch | CERN |

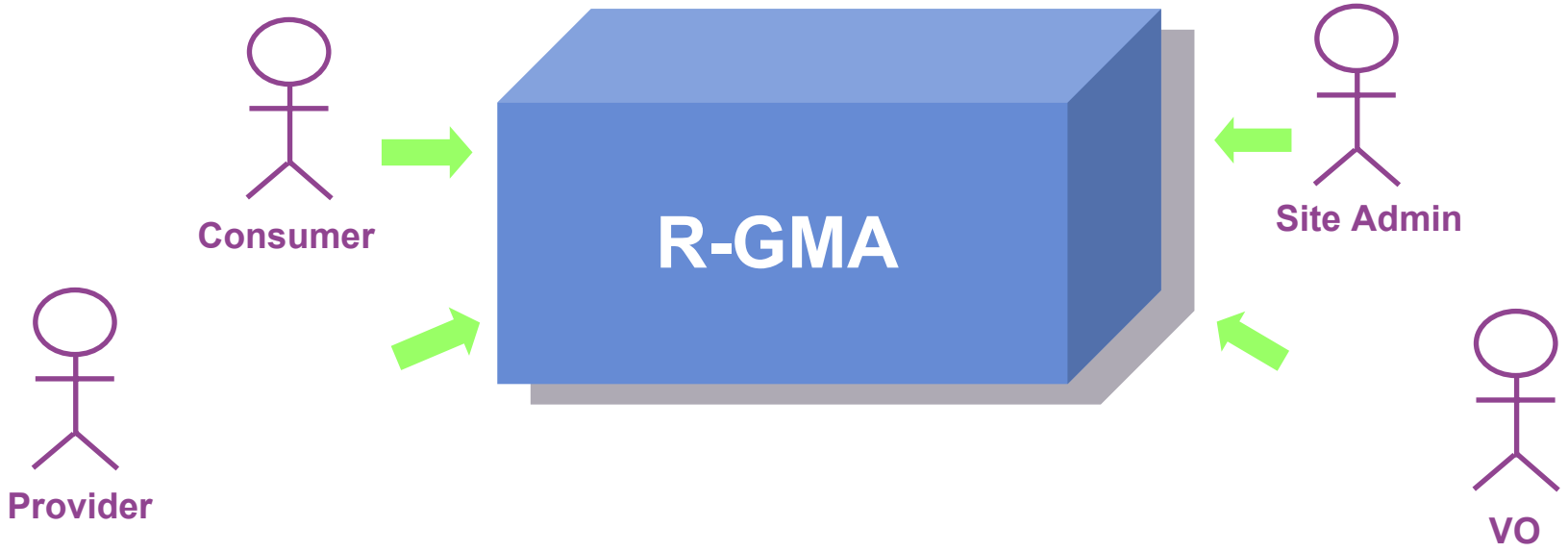
| ServiceStatus | | | | |
|---|---|---|---|---|
| URI | VO | type | up | status |
| gppse01 | alice | SE | y | SE is running |
| gppse01 | atlas | SE | y | SE is running |
| gppse02 | cms | SE | n | SE ERROR 101 |
| lxshare0404 | alice | SE | y | SE is running |
| lxshare0404 | atlas | SE | y | SE is running |



| Result Set (Consumer) | |
|---|---|
| URI | emailContact |
| gppse02 | sysad@rl.ac.uk |

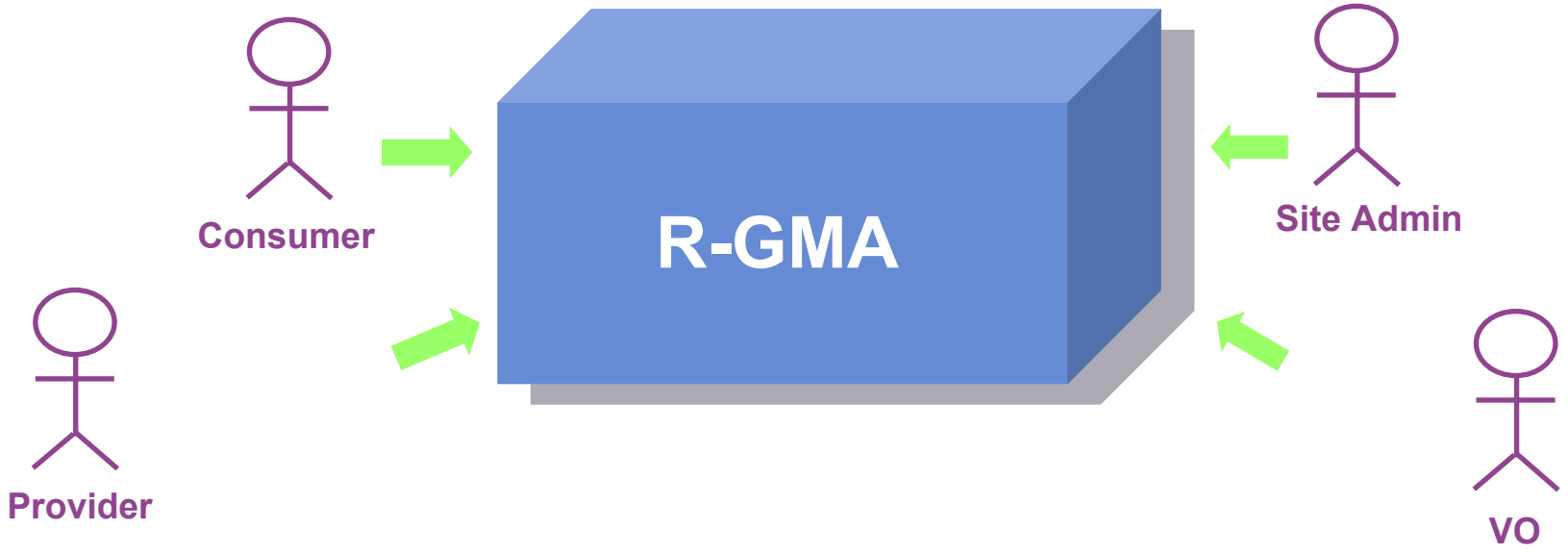
SELECT Service.URI Service.emailContact FROM Service S, ServiceStatus SS
WHERE (S.URI= SS.URI and SS.up='n')

# Roles



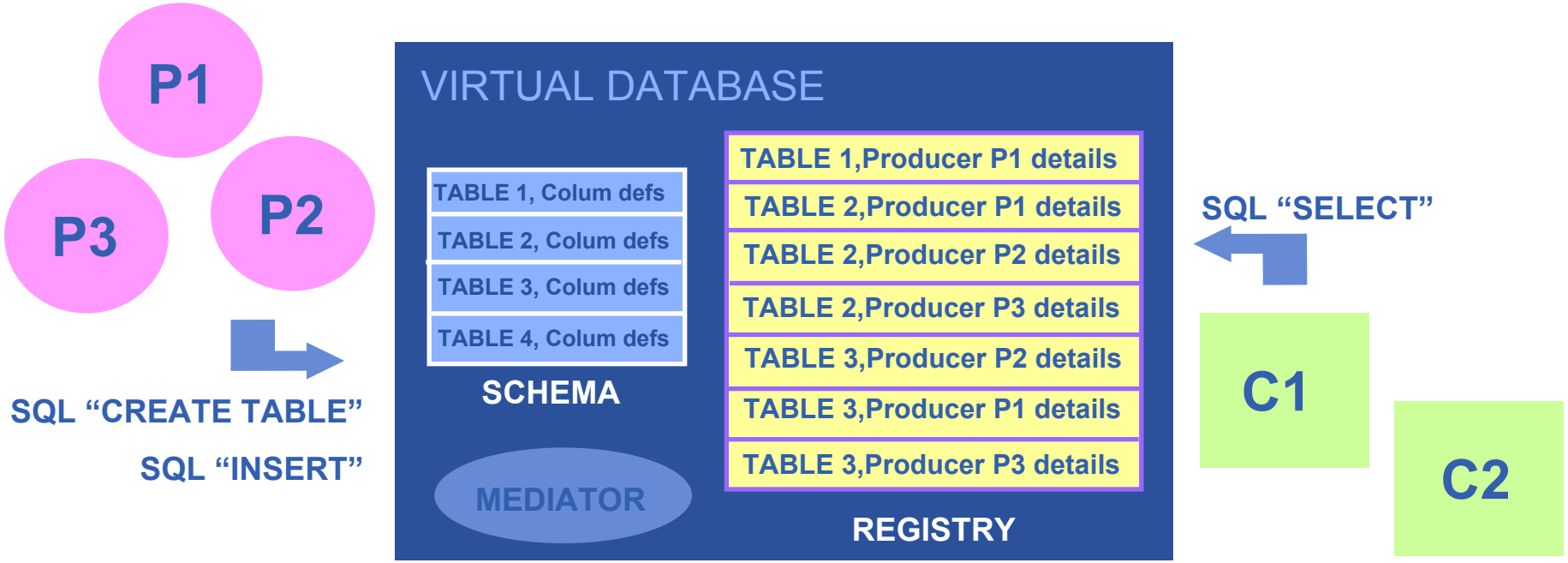

- Consumer users: who request information.
- Producer users: who provide information.
- Site administrators: who run R-GMA services.
- Virtual Organizations: who “own” the schema and registry.

# Security




- Mutual Autentication: guaranteeing who is at each end of an exchange of messages.
- Encryption: using an encrypted transport protocol (HTTPS).
- Authorization: implicit or explicit.

# R-GMA




**There is no central repository!!! There is only a "*Virtual Database*".**

**Schema is a list of table definitions: additional tables/schema can be defined by applications**

**Registry is a list of data producers with all its details.**

**Producers publish data.**

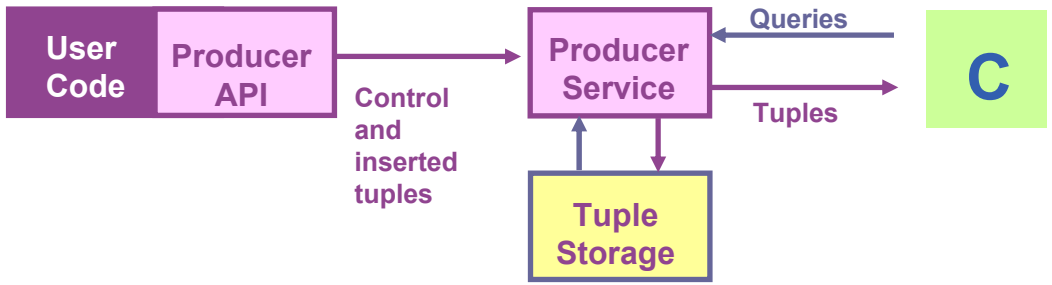
**Consumer read data published.**

# Deployment

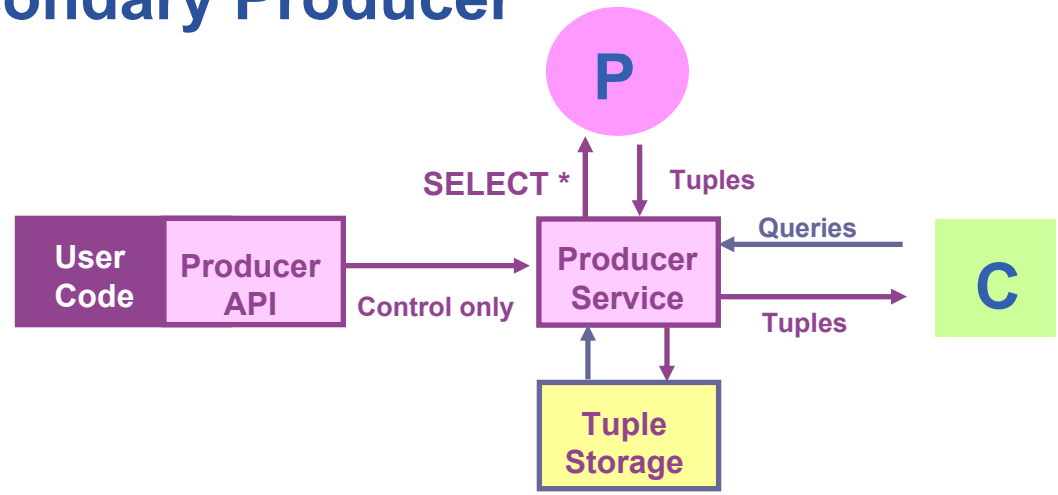
- **Producer and Consumer Services are typically on a one per site basis**
- **Centralized Registry and Schema.**
- **The Registry and Schema may be replicated, to avoid a single point of failure**
  - … when you use RGMA CLI you will see which are being used

# Producer Types

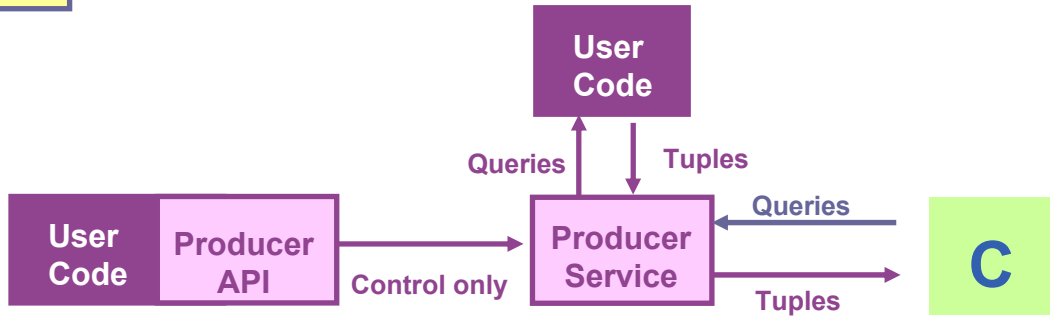
- **Primary Producer**




- **Secondary Producer**



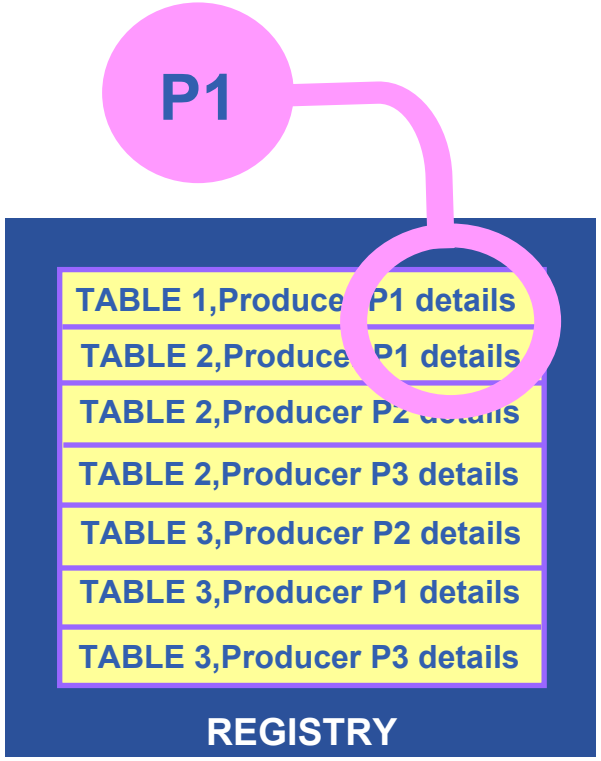

- **On-Demand Producer**
  - No internal storage
  - Queries passed to user code



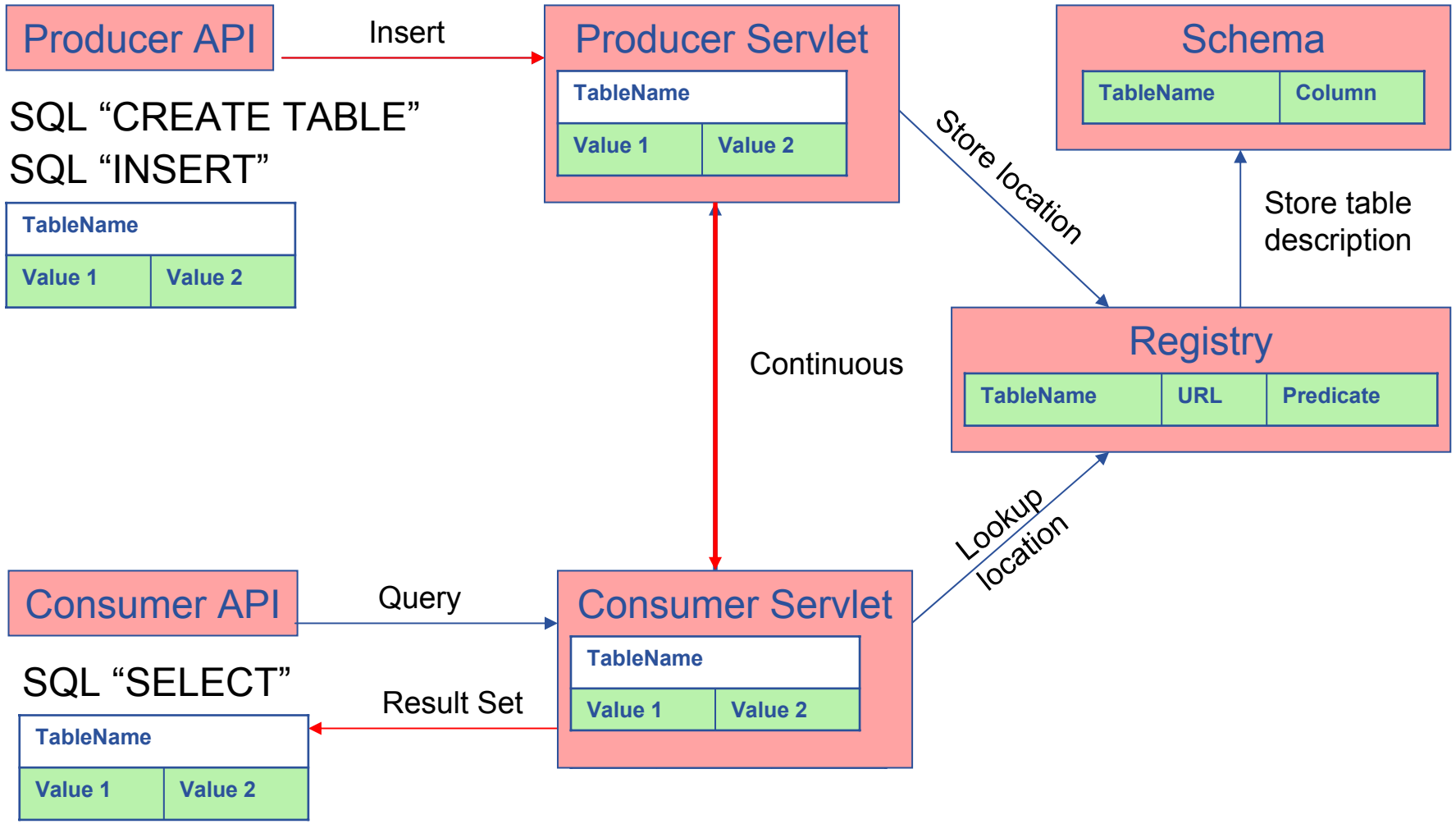
# Query Types

**Continuous**

- **Latest**
- **History**
- **Static**

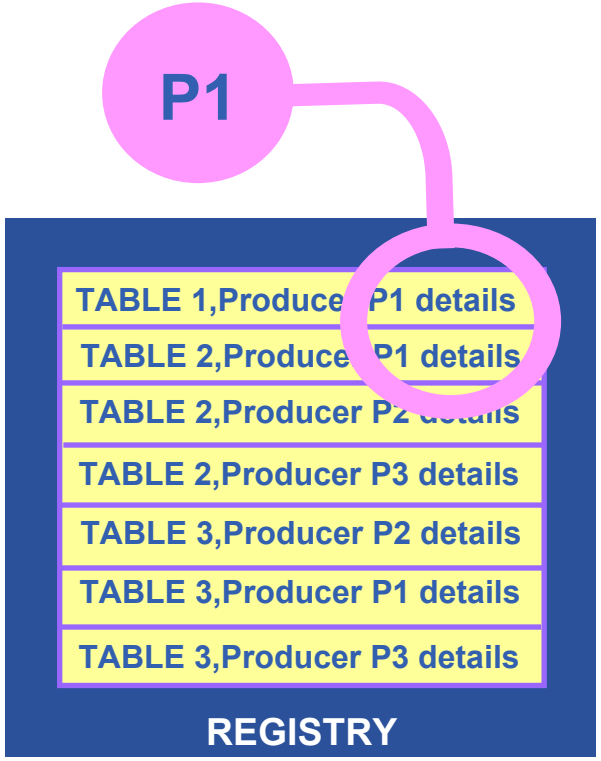


# Continuous

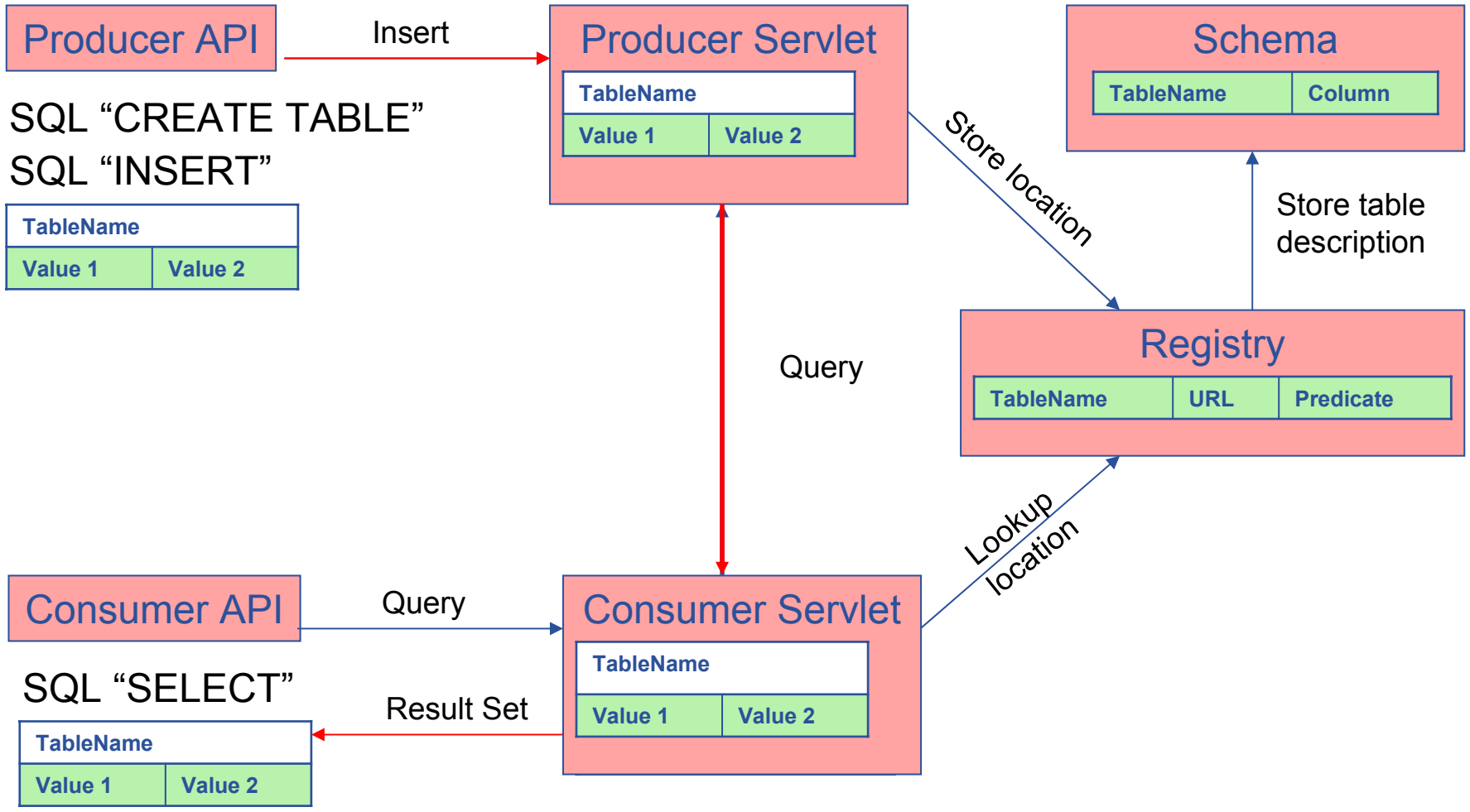

Producer API
Insert
Producer Servlet
TableName
Value 1
Value 2
Schema
TableName
Column
SQL “CREATE TABLE”
SQL “INSERT”
TableName
Value 1
Value 2
Store location
Store table description
Continuous
Registry
TableName
URL
Predicate
Lookup location
Consumer API
Query
Consumer Servlet
TableName
Value 1
Value 2
SQL “SELECT”
Result Set
TableName
Value 1
Value 2

# Query Types

- Continuous
- Latest
- History
- Static

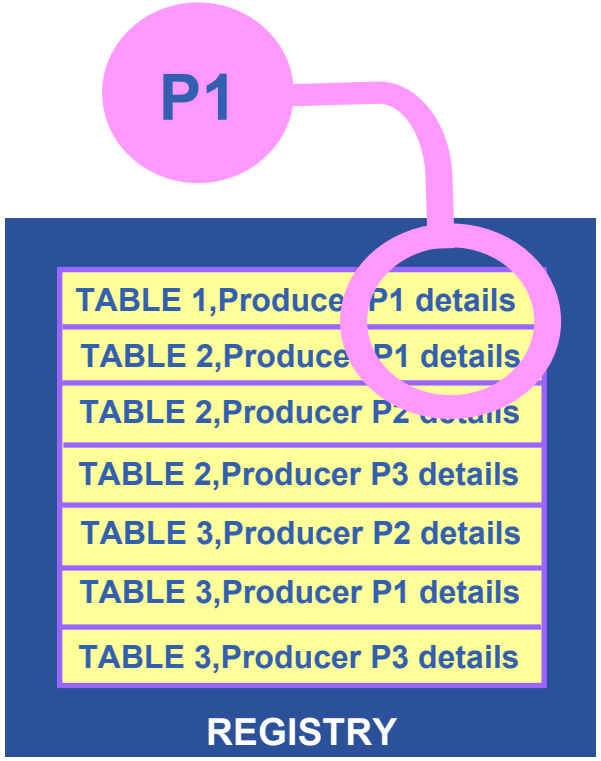


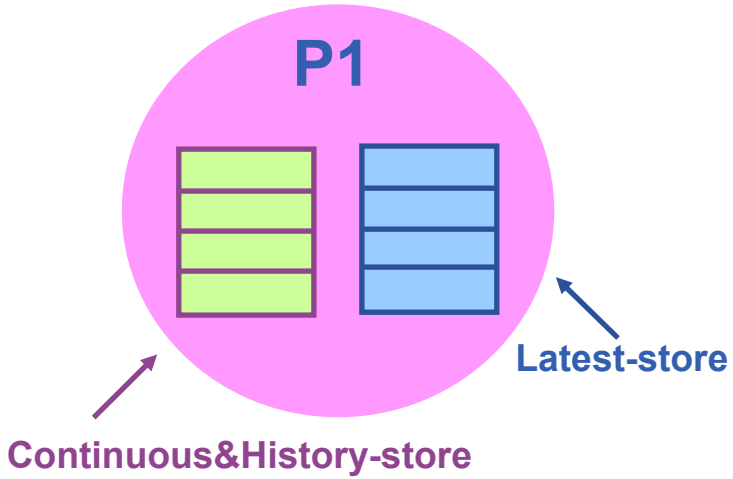
# History or Latest



# Query Types

- Continuous
- Latest
- History
- Static




Latest Retention Period

History Retention Period

# R-GMA APIs

- **APIs exist in Java, C, C++, Python.**
  - For clients (servlets contacted behind the scenes)
- **They include methods for…**
  - Creating consumers
  - Creating primary and secondary producers
  - Setting type of queries, type of produces, retention periods, time outs…
  - Retrieving tuples, inserting data
  - …
- **You can create your own Producer or Consumer.**

# More information

- **R-GMA overview page.**
  - http://www.r-gma.org/
- **R-GMA in EGEE**
  - http://hepunx.rl.ac.uk/egee/jra1-uk/
- **R-GMA Documentation**
  - http://hepunx.rl.ac.uk/egee/jra1-uk/LCG/doc/

# R-GMA practical

# The R-GMA Browser

- **The easiest way to try out R-GMA.**
  - It is installed on the machine running the Registry and Schema:

    https://rgmasrv.ct.infn.it:8443/R-GMA

- **YOU NEED A GILDA CERTIFICATE IN YOUR BROWSER**

- **Using the Browser you can do the following.**
  - Browse the tables in the schema.
  - Look at the table definitions.
  - See all the available producers for a table.
  - Query a table.
  - Query only selected producers.

# R-GMA Command Line Tool (1)

- **CHECK YOU HAVE A VOMS PROXY CERTIFICATE**
- To Start the R-GMA command line tool run the following command:

**>rgma**

- On startup you should receive the following message:

```
Welcome to the R-GMA virtual database for Virtual Organisations.
You are connected to the R-GMA registry service at

http://<registry-host>:8080/R-GMA/RegistryServlet

Type "help" for a list of commands.
rgma>
```

# Entering Command

- Commands are entered by typing at the rgma> prompt and hitting ‘enter’ to execute the command.
- A history of the commands executed can be accessed using the Up and Down arrow keys.
- To search a command from history use CTRL-R and type the first few letters of the command to recall.
- Command autocompletion is supported (use Tab when you have partly entered a command).

# General Commands

## General Commands

- **exit or quit**

  **Exit from R-GMA command line interface.**

- **help**

  **Display general help information.**

- **help <command>**

  **Display help for a specific command.**

- **Show tables**

  **Display the name of all tables existing in the Schema**

- **Describe <tablename>**

  **Show all information about the structure of a table**

# Querying Data (1)

- Querying data uses the standard SQL SELECT statement, e.g.:

  rgma> SELECT * FROM GlueService

  The behaviour of SELECT varies according to the type of query being executed. In R-GMA there are three basic types of query:
- LATEST Queries only the most recent tuple for each primary key
- HISTORY Queries all historical tuples for each primary key
- CONTINUOUS Queries returns tuples continuously as they are inserted.

# Querying Data (2)

- The type of query can be changed using the SET QUERY command as follow:

```
rgma> SET QUERY LATEST
```

or

```
rgma> SET QUERY CONTINUOUS
```

- The current query type can be displayed using

```
rgma> SHOW QUERY
```

# Exercises

1. Display all the table of the Schema

   rgma>show tables

2. Display information about GlueSite table

   rgma>describe GlueSite

3. Basic select query on the table named GlueSite

   rgma>set query latest

   rgma>show query

   rgma> select Name,Latitude,Longitude from GlueSite

# Maximum AGE of tuples

- The maximum age of tuples to return can also be controlled. To limit the age of latest or historical tuples use the SET MAXAGE command. The following are equivalent:

  rgma> SET MAXAGE 2 minutes

  rgma> SET MAXAGE 120

- The current maximum tuple age can be displayed using rgma> SHOW MAXAGE
- To disable the maximum age, set it to none:

  rgma> SET MAXAGE none

# Query Timeout

- The final property affecting queries is timeout.
  - For a latest or history query the timeout exists to prevent a problem (e.g. network failure) from stopping the query from completing.
  - For a continuous query, timeout indicates how long the query will continue to return new tuples. Default timeout is 1 minute and it can be changed using

rgma>SET TIMEOUT 3 minutes or SET TIMEOUT 180

- The current timeout can be displayed using

rgma>SHOW TIMEOUT

# Producer & Inserting Data

- **The SQL INSERT statement may be used to add data to the system:**

  **rgma> INSERT INTO userTable VALUES ('a', 'b', 'c', 'd')**

- **In R-GMA, data is inserted into the system using a Producer component which handles the INSERT statement.**
- **Using the command line tool you may work with one producer at a time.**
- The current producer type can be displayed using:

  **rgma>show producer**
- The producer type can be set using:

  **rgma>set producer latest**

# Exercise

**Choose a role for the exercise as consumer or as producer (alternate if you wish)**

**PRODUCERS**

rgma> set producer continuous

rgma> set maxage 3 minutes

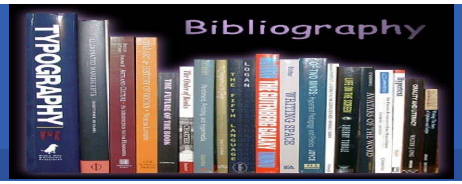
rgma> insert into userTable values('taipeixx','any string',1.4,66)

**CONSUMERS**

rgma> set query continuous OR set query history

rgma> set timeout 5 seconds

rgma> select * from userTable




# References

- **LCG-2 User Guide Manual Series**

  **https://edms.cern.ch/file/454439/LCG-2-UserGuide.html**