Job management with gLite

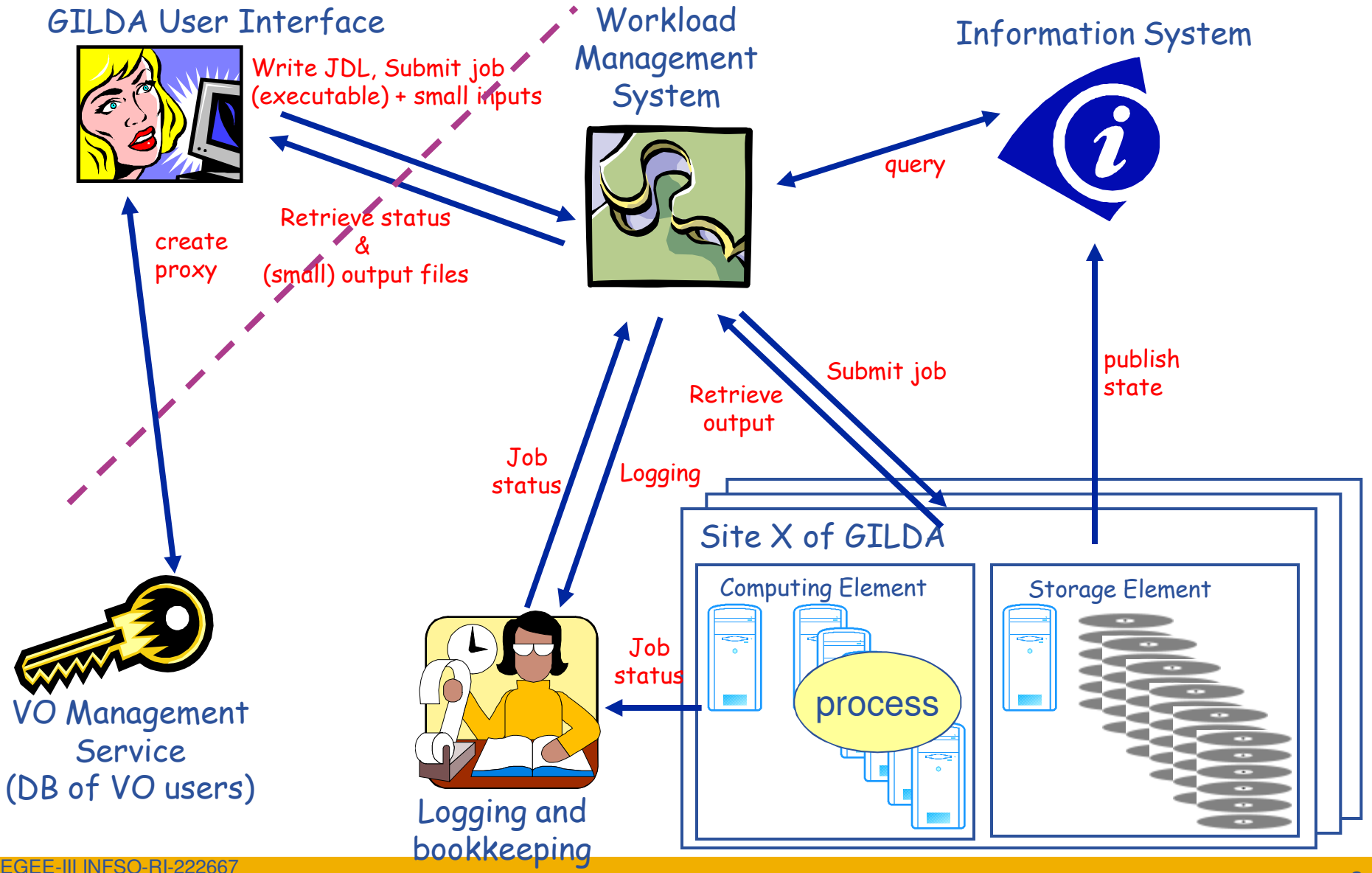
Gergely Sipos sipos@sztaki.hu
Training and induction
Application porting support www.lpds.sztaki.hu/gasuc

MTA SZTAKI
www.lpds.sztaki.hu

Outline

- Workload management concept in gLite
- Executing a single job
- Executing complex jobs
- Practicals

gLite use case: Execution of a job



Which CE do you want to use?

- **Without the WMS, use the Information System to see what's available, then choose…**
  lcg-infosites --vo gilda ce

```
#CPU    Free    Total Jobs      Running Waiting ComputingElement
----------------------------------------------------------
  28      28      0               0       0       ce.hpc.iit.bme.hu:2119/jobmanager-lcgpbs-gilda
  10      10      0               0       0       grid011f.cnaf.infn.it:2119/jobmanager-lcgpbs-gilda
  52      51      1               1       0       grid010.ct.infn.it:2119/jobmanager-lcgpbs-long
  16      16      0               0       0       gilda-01.pd.infn.it:2119/jobmanager-lcgpbs-gilda
  56      54      1               0       1       iceage-ce-01.ct.infn.it:2119/jobmanager-lcgpbs-short
 …….[70% shown].
```

- **WMS does this for you!**
  - chooses CE for each job, balances workload, manages jobs and their files

Logging into the Grid: Creating a proxy credential

```
[sipos@glite-tutor ~]$ ls -l .globus/
-rw-r--r--  1 sipos users 1761 Dec  2  2008 usercert.pem
-r--------    1 sipos users   951 Oct 24  2006 userkey.pem

[sipos@glite-tutor sipos]$ voms-proxy-init --voms gilda
Enter GRID pass phrase: ***********
Your identity: /C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=Gergely
Sipos/Email=sipos@sztaki.hu
Creating temporary proxy ..................................................... Done
Contacting  voms.ct.infn.it:15001 [/C=IT/O=INFN/OU=Host/L=Catania/CN=voms.ct.infn.it]
"gilda" Done
Creating proxy ................................ Done
Your proxy is valid until Sat Jun 23 04:55:19 2007
```

% voms-proxy-init → login to the Grid

Enter PEM pass phrase: ****** → private key is protected by a password

- Options for voms-proxy-init:
  - VO name
  - -hours <lifetime of new credential>
  - -help

% voms-proxy-destroy → logout from the grid

Writing a JDL file

```
[sipos@glite-tutor sipos]$ nano OR vi... hostname.jdl

Type = "Job";
JobType = "Normal";
Executable = "/bin/hostname";
StdOutput = "hostname.out";
StdError = "hostname.err";
OutputSandbox = {"hostname.err","hostname.out"};
Arguments = "-f";
ShallowRetryCount = 3;
```

Basic JDL-file attributes

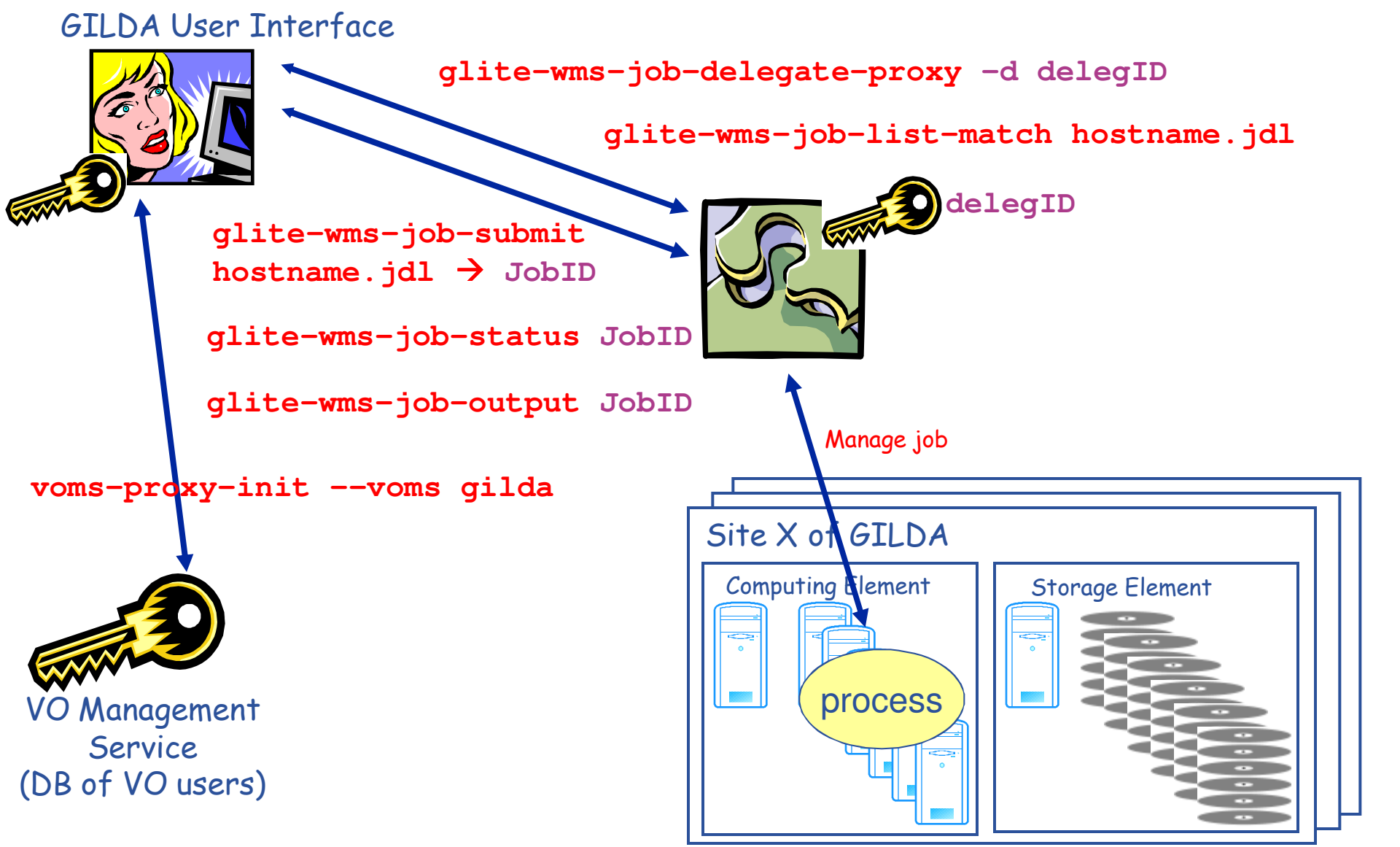
- **`Executable`** – sets the name of the executable file;
- **`Arguments`** – command line arguments of the program;
- **`StdOutput, StdError`** - files for storing the standard output and error messages output;
- **`InputSandbox`** – set of input files needed by the program, including the executable;
- **`OutputSandbox`** – set of output files which will be written during the execution, including standard output and standard error output; these are sent from the CE to the WMS for you to retrieve
- **`ShallowRetryCount`** – in case of grid error, retry job this many times (“Shallow”: before job is running)

Job management commands

Recommended

WMS version	LCG-2 WMS	gLite WMS via NS gLite 3.0	gLite WMS via WMProxy gLite 3.1+
Delegate proxy		**D**	**glite-wms-job-delegate-proxy** -d delegID
Submit	**edg-job-submit** [-o joblist]jdlfile	**E** **glite-job-submit** [-o joblist] jdlfile	**glite-wms-job-submit** [-d delegID] [-a] [-o joblist] jdlfile
Status	**edg-job-status** [-v verbosity] [-i joblist] jobIDs	**P** **glite-job-status** [-v verbosity] [-i joblist] jobIDs	**glite-wms-job-status** [-v verbosity] [-i joblist] jobIDs
Logging	**edg-job-get-logging-info** [-v verbosity] [-i joblist] jobIDs	**R** **E** **glite-job-logging-info** [-v verbosity] [-i joblist] jobIDs	**glite-wms-job-logging-info** [-v verbosity] [-i joblist] jobIDs
Output	**edg-job-get-output** [-dir outdir] [-i joblist] jobIDs	**C** **A** **glite-job-output** [-dir outdir] [-i joblist] jobIDs	**glite-wms-job-output** [-dir outdir] [-i joblist] jobIDs
Cancel	**edg-job-cancel** [-i joblist] jobID	**T** **E** **glite-job-cancel** [-i joblist] jobID	**glite-wms-job-cancel** [-i joblist] jobID
Compatible resources	**edg-job-list-match** jdlfile	**D** **glite-job-list-match** jdlfile	**glite-wms-job-list-match** [-d delegID] [-a] jdlfile

gLite use case 1 with user commands



Job states
Output of glite-wms-job-status

Flag	Meaning
SUBMITTED	**submission logged in the Logging & Bookkeeping service**
WAIT	**job match making for resources**
READY	**job being sent to executing CE**
SCHEDULED	**job scheduled in the CE queue manager**
RUNNING	**job executing on a Worker Node of the selected CE queue**
DONE	**job terminated without grid errors**
CLEARED	**job output retrieved**
ABORT	**job aborted by middleware, check** *reason*

The "Executable"

```
[sipos@glite-tutor sipos]$ nano/vi/etc hostname.jdl
…
Executable = "/bin/hostname";
…
```

- **Installed on the CE**
  - Standard software in Linux (Scientific Linux!)
  - VO specific software: advertised in information system
    - *Use JDL expressions to navigate job to such a site*
- **Or Comes from client side**
  - Part of InputSandbox
    - Script
      - *No compilation is necessary*
      - *Can invoke binary that is statically installed on the CE*
    - Or Binary
      - *Must be compiled on the User Interface → binary compatibility with CEs*
      - *Statically linked → to avoid errors caused by different library versions*

Submitting your own script

```
$ cat testsandbox.jdl
Type = "Job";
JobType = "Normal";
Executable = "/bin/sh";
Arguments = "testsandbox.sh";
StdOutput = "testsandbox.out";
StdError = "testsandbox.err";
InputSandbox = "testsandbox.sh";
OutputSandbox = {"testsandbox.err","testsandbox.out"};
ShallowRetryCount = 1;
```

```
$ cat testsandbox.sh
#!/bin/bash
ls -l
```

```
$ /bin/sh testsandbox.sh
```

Submitting your executable with a wrapper script

```
$ cat yourexe.jdl
Type = "Job";
JobType = "Normal";
Executable = "/bin/sh";
Arguments = "script.sh INSERT_YOUR_NAME";
StdOutput = "script.out";
StdError = "script.err";
InputSandbox = {"script.sh","myexecutable"};
OutputSandbox = {"script.out","script.err","exe.out"};
ShallowRetryCount = 1;
```

Compiled on UI

```
cat script.sh
#!/bin/sh
echo "setting right permissions"
chmod 755 myexecutable
echo "executing program now..."
./myexecutable $1 > exe.out
```

```
$ /bin/sh script.sh Gergely
```

Influencing brokering with JDL

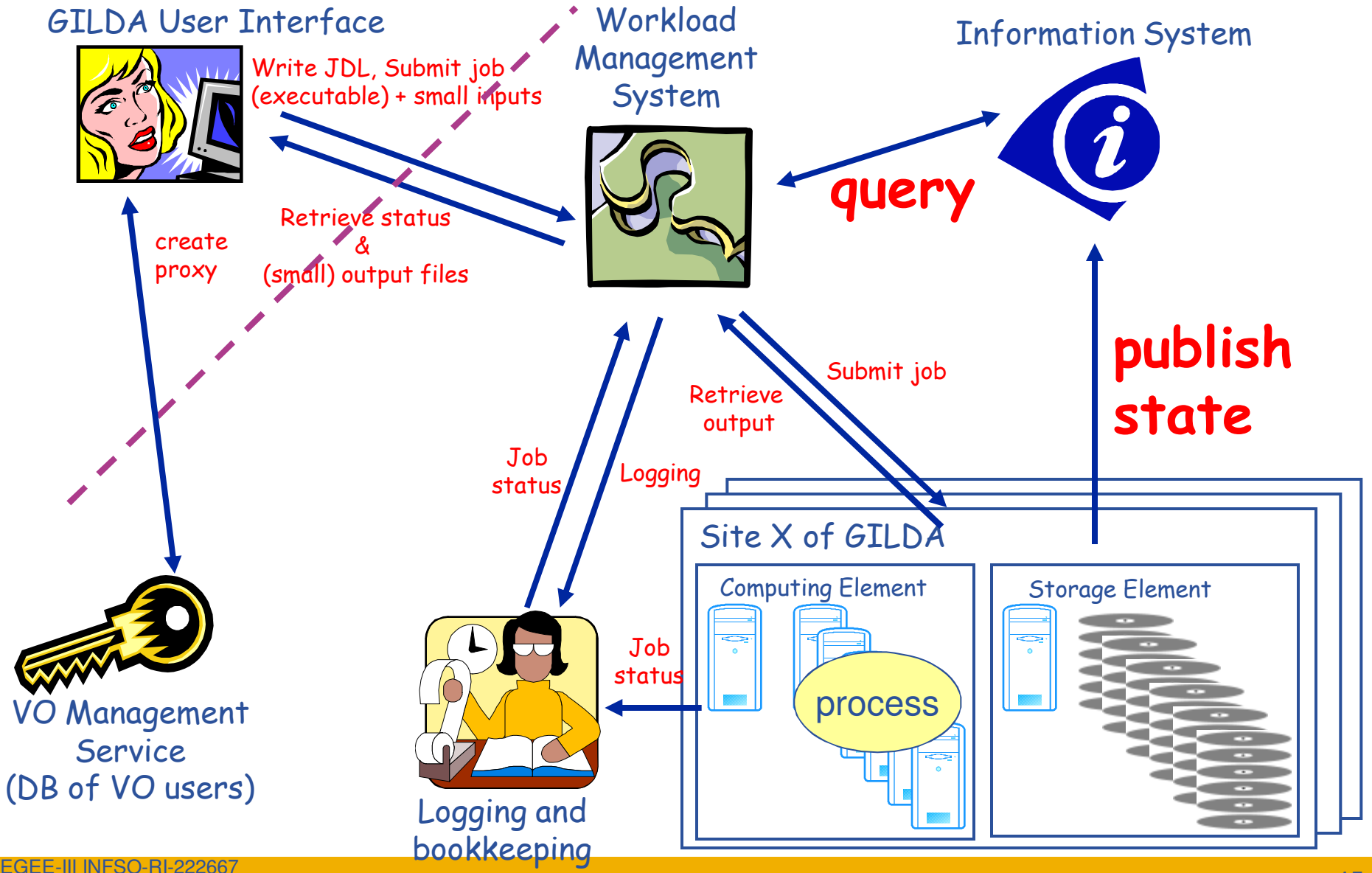
```
Executable = "gridTest";
StdError = "std
StdOutput = "st
InputSandbox =                                  Test"};
OutputSandbox =                                 t.log"};

Requirements = other.Architecture=="INTEL" &&
   other.GlueCEInfoTotalCPUs > 480;
Rank = other.GlueCEStateTotalJobs;
```

WMS uses Information System to find CE

WMS brokering policy :

- **Meet CE requirements**
- **Select CE with highest rank**

Handling Requirements and Rank



Brokering policy

1. **Meet CE requirements (defined by Requirements part of JDL)**

2. **Select CE which is close to InputData**
   - "Close" relationship is defined between CEs and SEs by site administrators
   - "Close" is not necessarily physical distance – rather bandwidth
   - "Close" typically means same site
     - *CE:* iceage-ce-01.ct.infn.it:2119/jobmanager-lcgpbs-short
     - *Close SE:* iceage-se-01.ct.infn.it

3. **Select CE with highest rank (rank formula is defined by Rank part of JDL)**

Some relevant CE attributes

- **GlueCEUniqueID** – Identifyer of a CE
  - Eliminating an erroneous CE:

```
other.GlueCEUniqueID !=
"grid010.ct.infn.it:2119/jobmanager-lcgpbs-long"
```

  - Sending the job to a given CE:

```
other.GlueCEUniqueID ==
"grid010.ct.infn.it:2119/jobmanager-lcgpbs-long"
```

- **GlueCEInfoTotalCPUs** – max number of CPUs at a CE

```
Rank = other. GlueCEInfoTotalCPUs;
```

- **GlueCEStateWaitingJobs** – number of waiting jobs
- **GlueCEPolicyMaxCPUTime** – job will be killed after this number of minutes
- **GlueHostMainMemoryRAMSize** – memory size

http://glite.web.cern.ch/glite/documentation/ → **JDL specification (submission via WMS WMProxy)**

Examples

- *Rank =*
  (other.GlueCEStateWaitingJobs == 0 ? other.GlueCEStateFreeCPUs : -other.GlueCEStateWaitingJobs);

  if there are no waiting jobs,
  - then the selected CE will be the one with the most free CPUs
  - else the one with the least waiting jobs.

- *Requirements =*
  (Member(„IDL2.1”, other.GlueHostApplicationSoftwareRunTimeEnvironment)) && (other.GlueCEPolicyMaxWallClockTime > 10000);

  CE where,
  - IDL2.1 software is available
  - At least 10000s can be spent on the site (waiting + running)

Complex workloads with gLite

From gLite 3.1

Complex jobs 1: Job collection

- **A set of independent jobs**
- **For some reason must be managed as a single unit**
- **Possible reasons:**
  - Belong to the same experiment
  - Share common input files
  - Optimize network traffic
- **Sharing of sandboxes**

JDL of a job collection

```
[
  Type = "collection";
  InputSandbox = {
        "sharedFile1";  . . .; "sharedFileM" };

  nodes = {
   [ JobType = "Normal";
     InputSandbox = {root.InputSandbox, . . .}
         …; ],
          . . .
   [ JobType = "Normal";
         …; ],

    . . .
  };
]
```

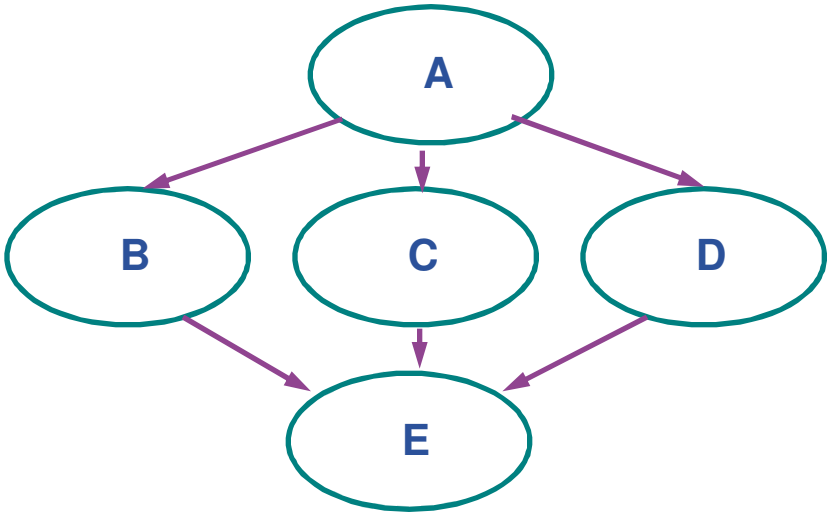
Transfer from UI only once

JDL of 1st job

JDL of Nth job

Complex jobs 2: DAG workflow

- **Direct Acyclic Graph (DAG)** is a set of jobs where the input, output, or execution of one or more jobs depends on one or more other jobs
- Sharing and inheritance of sandboxes
  - **Include sandbox output in the next inputsandbox**
- Dependencies defined between pairs of jobs

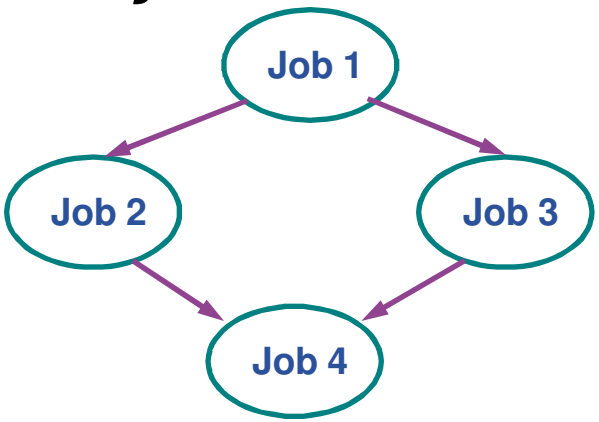


JDL of a DAG

```
[ Type = "dag";
  InputSandbox = {
        "sharedFile1"; . . .; "sharedFileN" };
  nodes = [
   job1 = [
        descpription = [
              JobType = "Normal";
              . . .; ],
   ];    . . .
    . . .
  ];
  dependencies =  {
   {job1,{job2,job3}}, {job2,job4}, {job3,job4}
  };
};
]
```

Transfer from UI only once

JDL of 1st job



Graph structure

Complex jobs 3: parametric jobs

- **A set of jobs generated from one JDL**
- **Useful where many similar (but not identical) jobs must be executed**
  - Parameter study, parametric sweep applications
  - Majority of grid applications are parametric!
- **One or more parametric attributes in the JDL:**
  - Use the _PARAM_ keyword
  - E.g. InputSandbox = “input_PARAM_";

JDL of a parametric job

```
[
   Type = "Parametric";
   . . .

   ParameterStart = 0;
   ParameterStep =2;
   Parameters= 6;    → _PARAM_ runs from 0 to 10

   Arguments = "inputfigure_PARAM_.jpg";
   StdOutput = "transformed_PARAM_.jpg";
   OutputSandbox  = {" transformed_PARAM_.jpg ",…};
   . . .
]
```

Summary – Users' todo list

1. **Create JDL file**
2. **Create proxy**
3. **(Delegate proxy)**
   - glite-wms-job-delegate-proxy
4. **Check some CEs match your requirements:**
   - glite-wms-job-list-match
5. **Submit job**
   - glite-wms-job-submit
6. **Do something else for a while! – gLite is not written for short jobs!**
7. **Check job status - occasionally**
   - glite-wms-job-status
8. **When job is "done", get output**
   - glite-wms-job-output

Related GILDA tutorials 1

1. **Security**

   https://grid.ct.infn.it/twiki/bin/view/GILDA/AuthenticationAuthorization

   - Investigate your certificate
   - Create proxy
   - Investigate your proxy

2. **Job submission**

   https://grid.ct.infn.it/twiki/bin/view/GILDA/SimpleJobSubmission

   - Create a simple JDL file
     - *copy&paste JDL file from tutorial into a file. Executable is a server side prg.*
   - Delegate proxy *(JobID saved in file)*
   - List the CEs that can accept it
   - Submit it
   - Check its status until its done
   - Retrieve output

Related GILDA tutorials 2

3. More complex, but still single jobs

https://grid.ct.infn.it/twiki/bin/view/GILDA/MoreOnJDL

- Submit a script from client side
  - Listing work directory of the job
- Submit a binary from client side with wrapper script
- Requirements, Ranks
  - ***Send the job to a particular CE***
  - ***Send the job to any CE where "GEANT4-6" is available***
  - ***Send a job anywhere but a particular CE (dealing with errors)***

4. Complex job types

https://grid.ct.infn.it/twiki/bin/view/GILDA/WmProxyUse

- Execute a job collection
- Execute a DAG
- Execute parametric jobs
- A bit of data management…

Extra tutorials

Certificate management

https://grid.ct.infn.it/twiki/bin/view/GILDA/CertificateManagement

- How to import certificate in a web browser
  - *Visit www.ggus.org to test your certificate*
    (GGUS - Global Grid User Support
- How to convert pkcs12 to pem
- How to send signed email
- How to export a certificate from the web browser

Query information system

https://grid.ct.infn.it/twiki/bin/view/GILDA/InformationSystems

Query of the Information System to discover CE and SE characteristics and status

Gaining access

Login to GILDA User Interface machine:

- Open SSH client and connect to
  - **glite-tutor.ct.infn.it**
  - *User name:* **********
  - *Password:* **********
- *Private key passphrase:* **********

- glite-tutor2.ct.infn.it – backup user interface

Further resources

- gLite manuals, documentation
  - http://glite.web.cern.ch/glite/documentation/ (gLite user guide)
- EGEE
  - http://www.eu-egee.org/
- gLite middleware
  - http://www.glite.org

Thank you