Advanced Job Submission on the Grid

Antun Balaz

Scientific Computing Laboratory

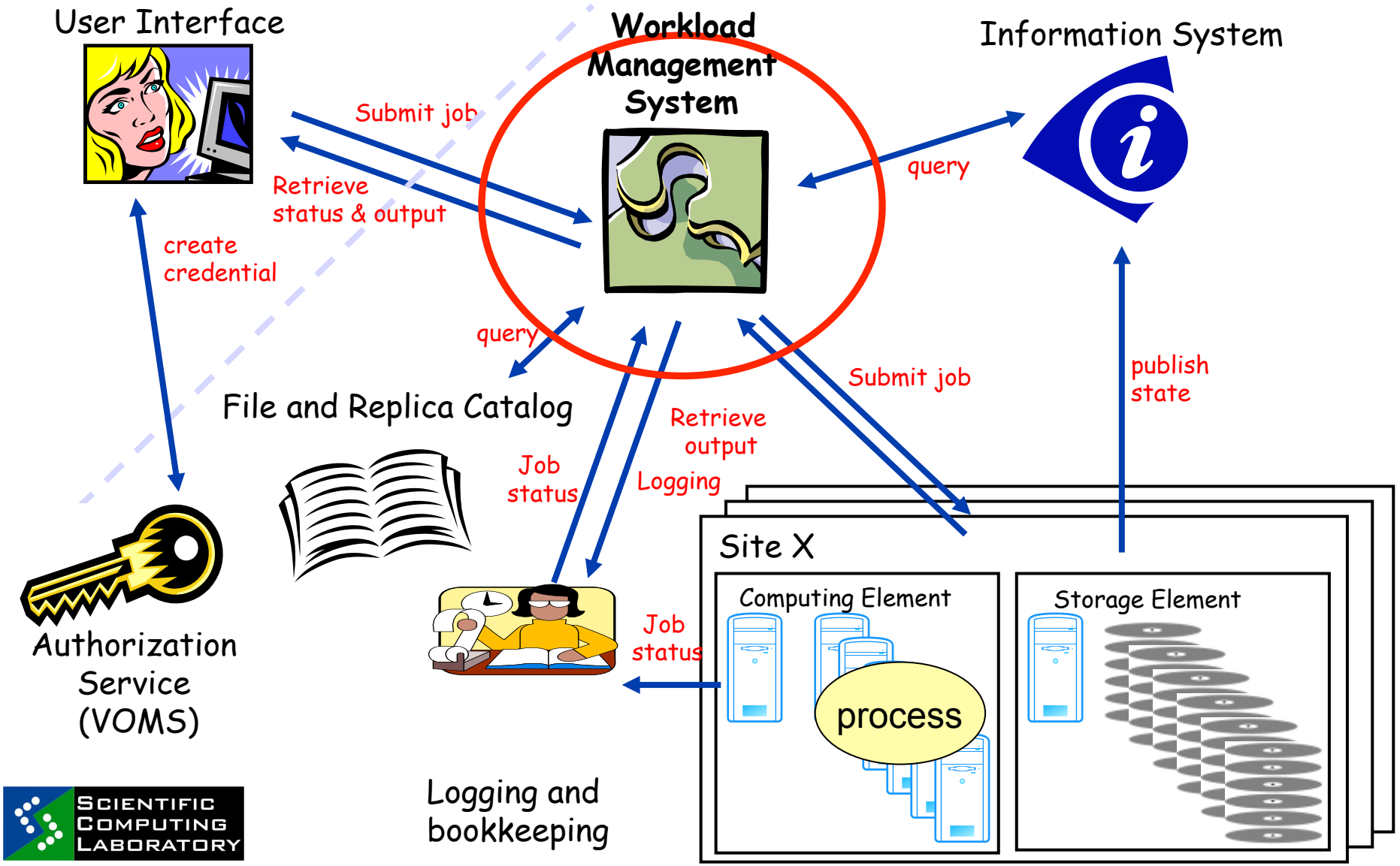
Institute of Physics Belgrade

http://www.scl.rs/

30 Nov – 11 Dec 2009

www.eu-egee.org

Scope


User Interface
Workload Management System
Information System
Submit job
Retrieve status & output
query
create credential
query
File and Replica Catalog
Submit job
publish state
Retrieve output
Job status
Logging
Site X
Computing Element
Storage Element
process
Job status
Authorization Service (VOMS)
Logging and bookkeeping

Contents

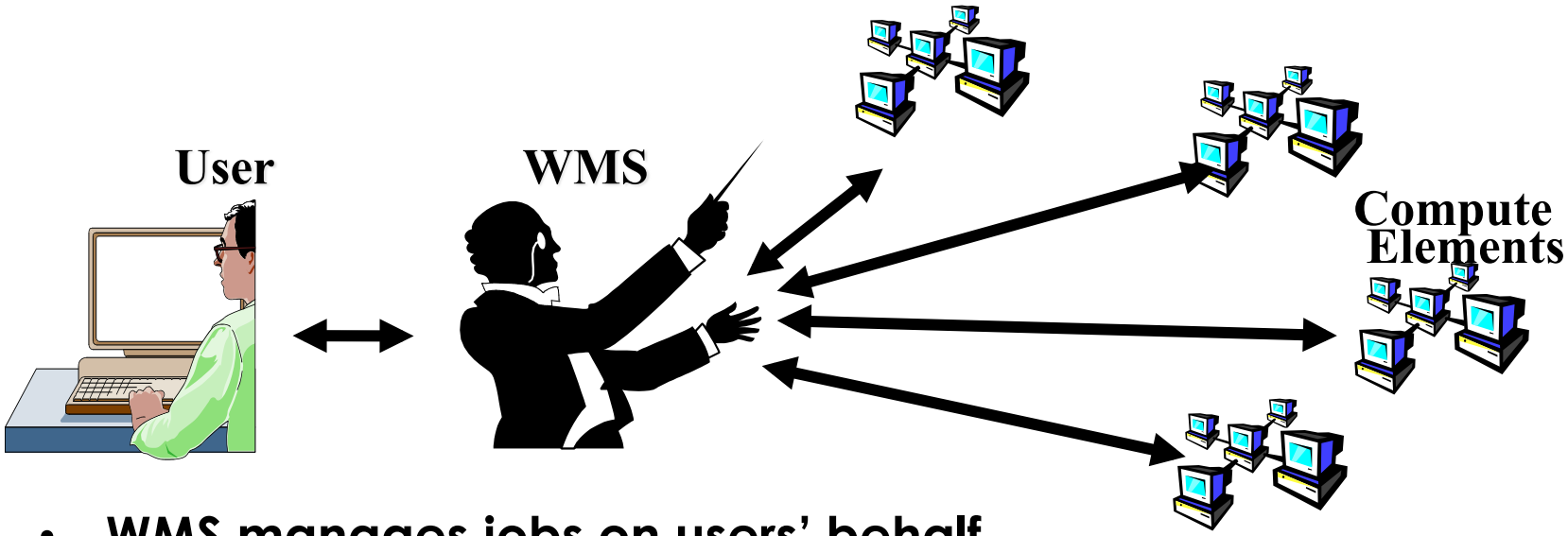
- Again a little bit on WMS
- How are your jobs handled
- JDL attributes
- Which types of jobs exist
- Examples of complex JDLs

Basics

Why does the Workload Management System exist?

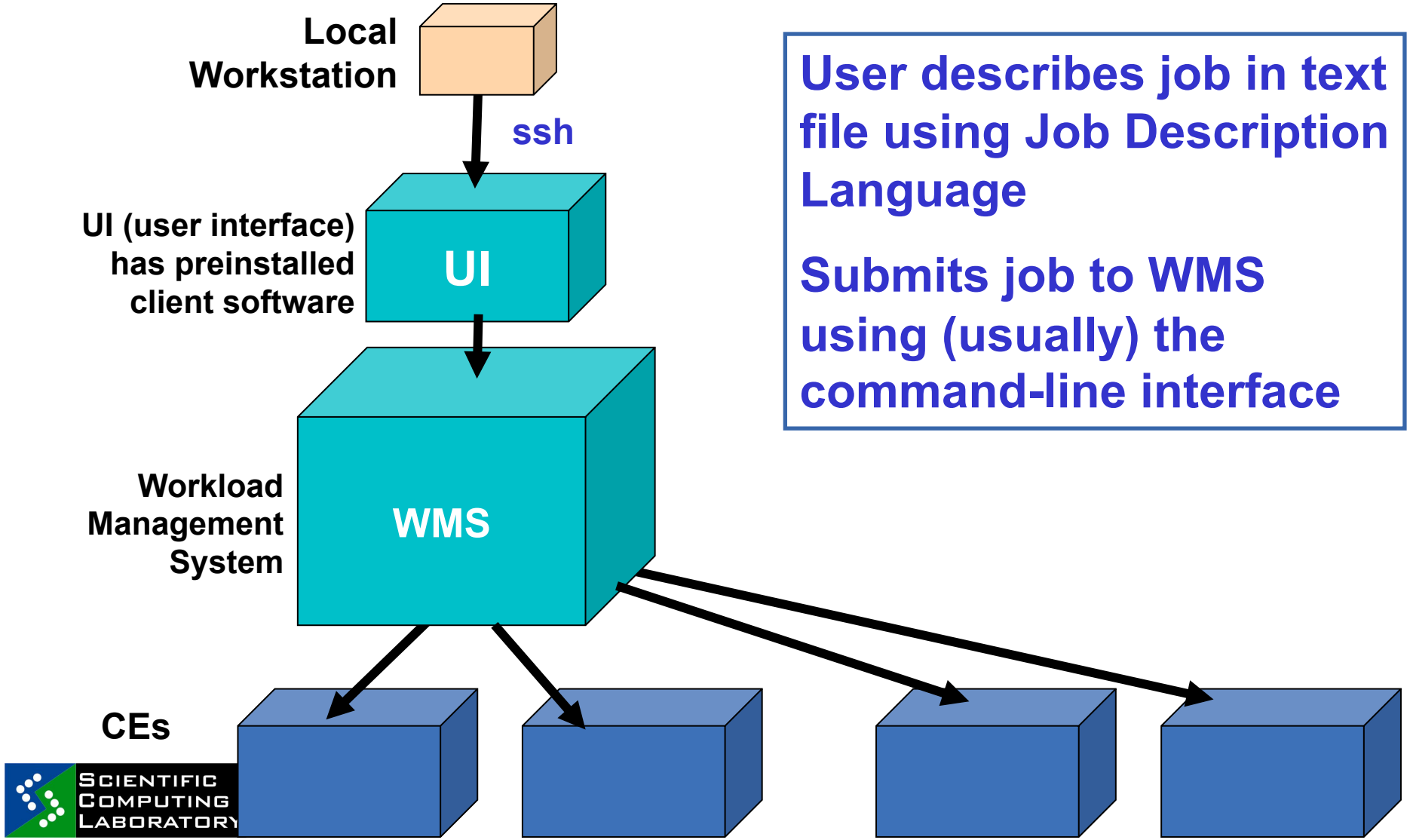
- **Grids have**
  - Many users
  - Many jobs – a "job" = an executable you want to run
  - Where many compute nodes are available
  - Workload Management System is a software service that makes running jobs easier for the user
- **It builds on the basic grid services**
  - E.g. Authorisation, Authentication, Security, Information Systems, Job submission
- **Terminology: "Compute element": defined as a batch queue - One cluster can have many queues**

What WMS does?



- **WMS manages jobs on users' behalf**
  - User doesn't decide where jobs are run
  - User defines the job and its requirememements, WMS matches this with available CEs
- **Effect:**
  - Easier submission
  - Users insulated from change in Compute elements
  - WMS – can optimise your jobs – e.g. which CE?

Managing jobs with gLite command line tools



User describes job in text file using Job Description Language

Submits job to WMS using (usually) the command-line interface

Using WMS

- Jobs run in batch mode on grids.
- Steps in running a job on a gLite grid with WMS:

1. Create a text file in "Job Description Language"
2. Optional check: list the compute elements that match your requirements ("list match" command)
3. Submit the job ~ "glite-wms-job-submit –a myfile.jdl" Non-blocking - Each job is given an id.
4. Occasionally check the status of your job
5. When "Done" retrieve output

Simple example of a JDL file

```
Type = "Job";
Executable = "/bin/hostname";
Arguments = "";
StdError = "stderr.txt";
StdOutput = "stdout.txt";
InputSandbox = "";
OutputSandbox = {"stderr.txt", "stdout.txt"};
```

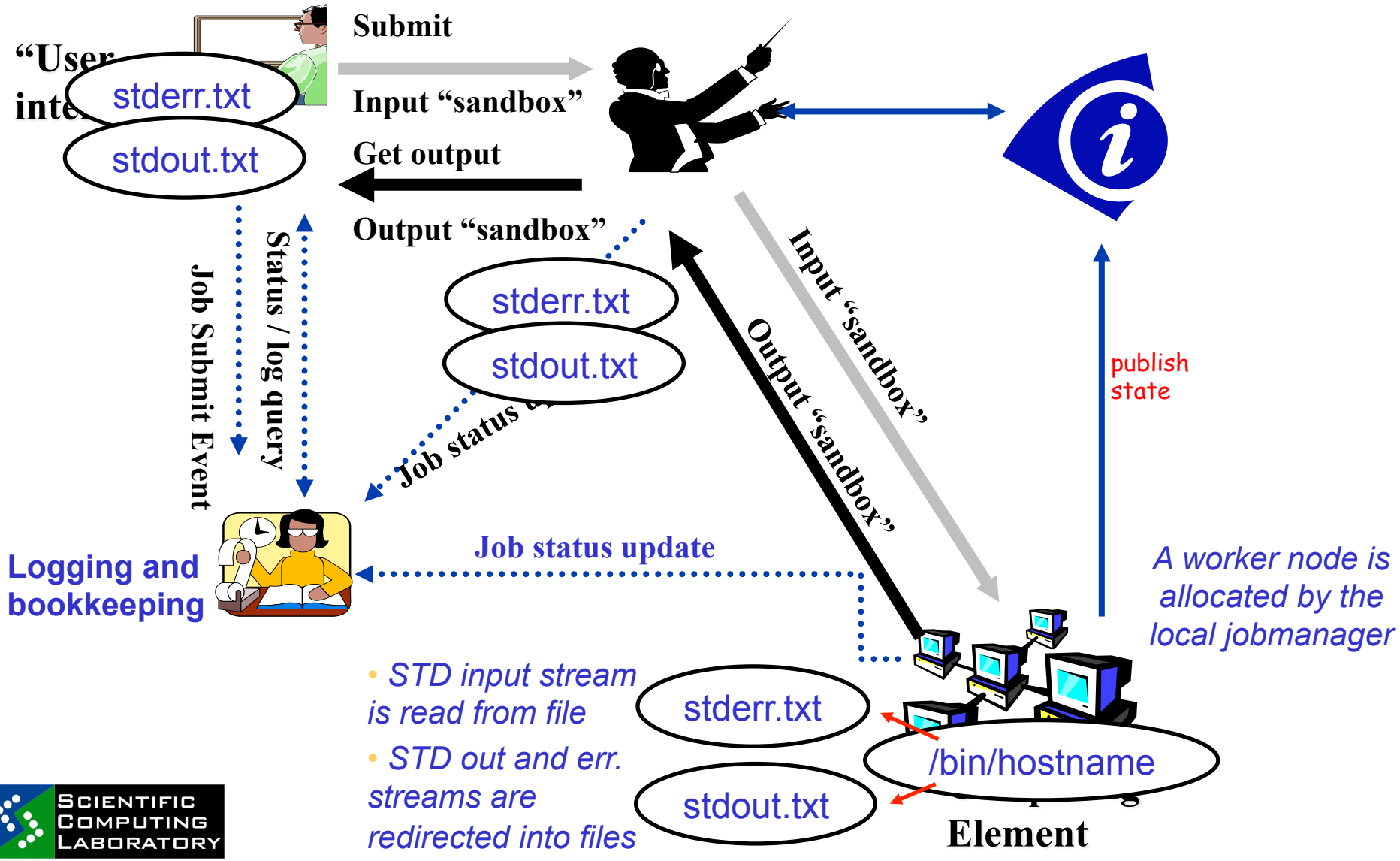
$ glite-wms-job-submit –a *my.jdl*

Returns a "job-id" used to monitor job, retrieve output...

Standard JDL-file attributes

- **`Type`** – "Job" for sequential jobs; later more details
- **`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)

Scenario


"User interface"
Submit
Input "sandbox"
Get output
Output "sandbox"
stderr.txt
stdout.txt
Job Submit Event
Status / log query
Job status
Input "sandbox"
Output "sandbox"
publish state
Logging and bookkeeping
Job status update
A worker node is allocated by the local jobmanager
• STD input stream is read from file
• STD out and err. streams are redirected into files
stderr.txt
stdout.txt
/bin/hostname
Element

The "Executable"

- **Script:**
  - No compilation is necessary
  - Can invoke binary that is statically installed on the CE
- **Binary:**
  - Must be compiled on the User Interface → binary compatibility with CEs is guaranteed
  - Statically linked → to avoid errors caused by library versions

- **Coming from client side**
  - Part of InputSandbox
- **Installed on the CE**
  - Standard software in Linux
  - VO specific software: advertised in information system
    - *Use JDL to navigate job to such a site*

SCIENTIFIC COMPUTING LABORATORY

Job control 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** **R** **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	**E** **C** **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	**A** **T** **glite-job-output** [-dir outdir] [-i joblist] jobIDs	**glite-wms-job-output** [-dir outdir] [-i joblist] jobIDs
Cancel	**edg-job-cancel** [-i joblist] jobID	**E** **D** **glite-job-cancel** [-i joblist] jobID	**glite-wms-job-cancel** [-i joblist] jobID
Compatible resources	**edg-job-list-match** jdlfile	**glite-job-list-match** jdlfile	**glite-wms-job-list-match** [-d delegID] [-a] jdlfile

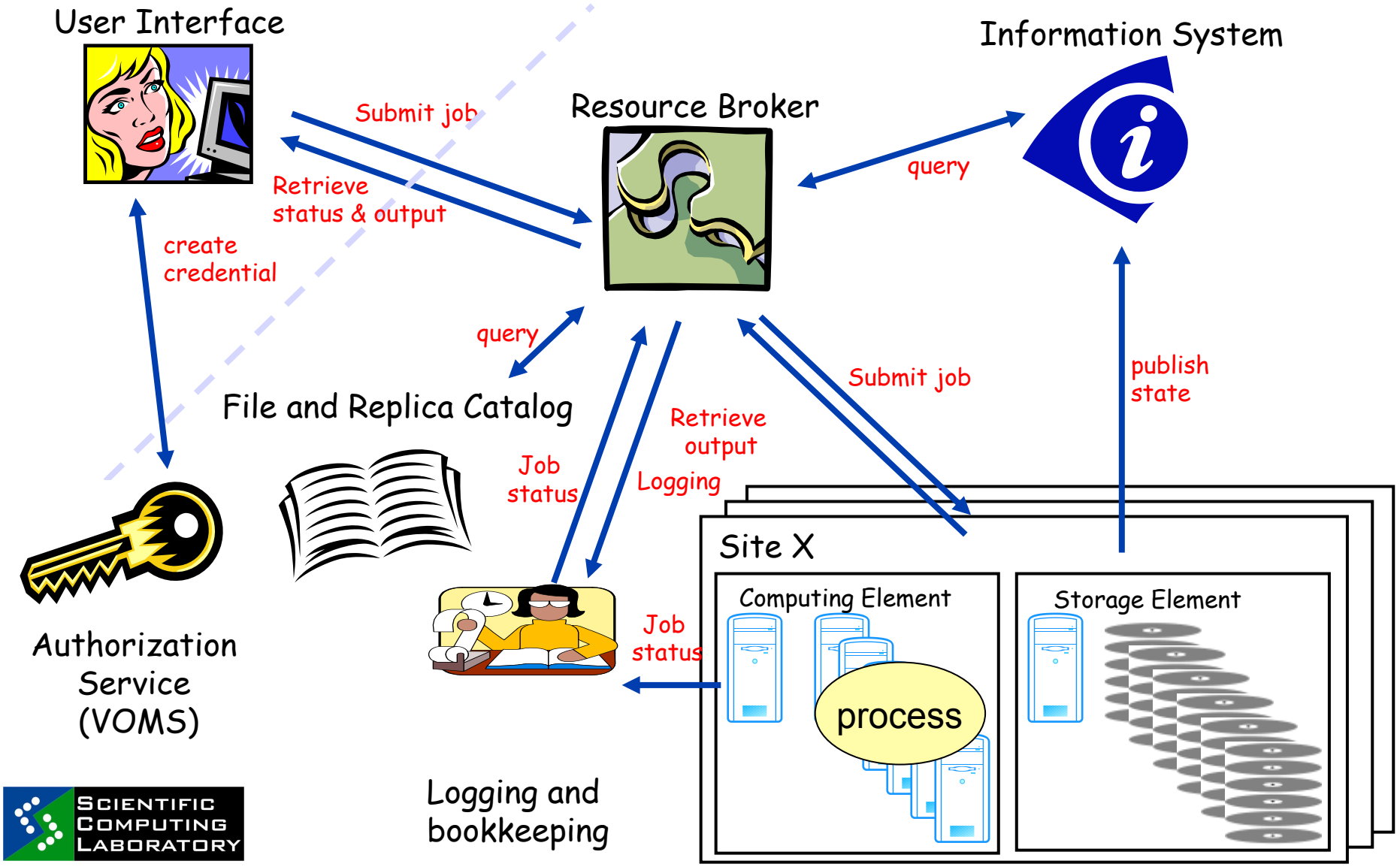
Specifying CE and data

```
Type = "Job";
Executable = "/bin/hostname";
Arguments = "";
StdError = "stderr.t
StdOutput = "stdout.
InputSandbox = "";
OutputSandbox = {"std                    stdout.txt"};

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

InputData = "lfn:/grid/gridbox/antun/file.txt";
```

WMS uses Information System to find CE

WMS uses File Catalog to find file location

Basic services of gLite



Location of files

Characteristics of resources

Network Daemon

Workload Manager

Job Contr. - CondorG

WMS

LFC

Inform. Service

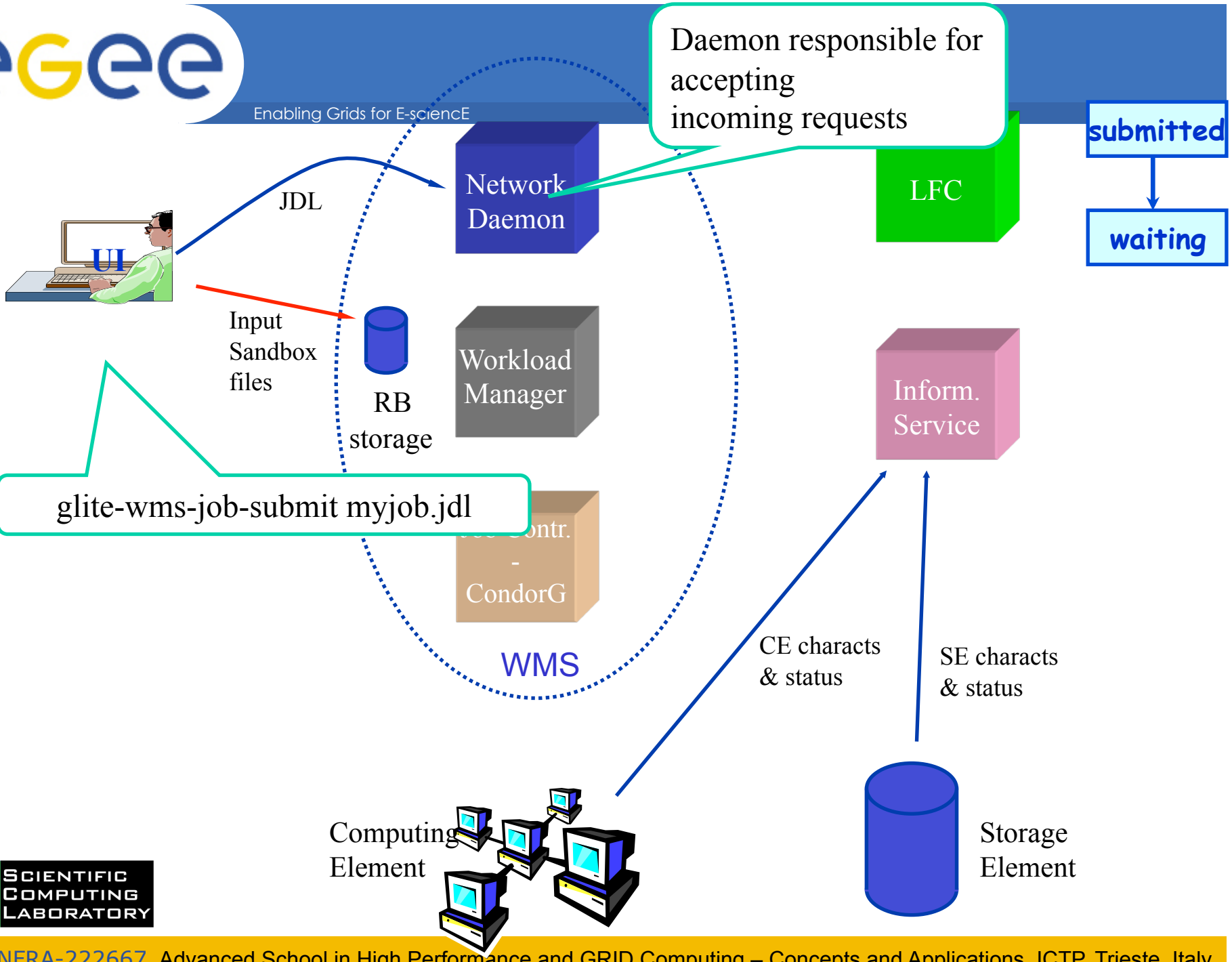
CE characts & status

SE characts & status

Computing Element

Storage Element

SCIENTIFIC COMPUTING LABORATORY


Daemon responsible for accepting incoming requests
submitted
waiting
Network Daemon
LFC
JDL
UI
Input Sandbox files
RB storage
Workload Manager
Inform. Service
glite-wms-job-submit myjob.jdl
CondorG
WMS
CE characts & status
SE characts & status
Computing Element
Storage Element

Network Daemon

Job

Workload Manager

RB storage

Job Contr. - CondorG

WMS

WM: responsible to take the appropriate actions to satisfy the request

LFC

Inform. Service

Job Status

submitted

waiting

CE characts & status

SE characts & status

Computing Element

Storage Element

Network Daemon

Match-Maker/ Broker

LFC

submitted

waiting

RB storage

Workload Manager

Where this job can be executed ?

Inform. Service

Job Contr. - CondorG

WMS

CE characts & status

SE characts & status

Computing Element

Storage Element

SCIENTIFIC COMPUTING LABORATORY

Matchmaker: responsible to find the "best" CE where to submit a job

Match-Maker/ Broker

LFC

submitted

waiting

RB storage

Workload Manager

Inform. Service

Job Contr. - CondorG

WMS

CE characts & status

SE characts & status

Computing Element

Storage Element

Where is the needed InputData ?

Network Daemon

Match-Maker/ Broker

LFC

submitted

waiting

RB storage

Workload Manager

Inform. Service

What is the status of the Grid ?

Job Contr. - CondorG

WMS

CE characts & status

SE characts & status

Computing Element

Storage Element

Network Daemon

Match-Maker/ Broker

LFC

submitted

waiting

RB storage

Workload Manager

CE choice

Inform. Service

Job Contr. - CondorG

WMS

CE characts & status

SE characts & status

Computing Element

Storage Element

SCIENTIFIC COMPUTING LABORATORY

Network Daemon

LFC

submitted

waiting

RB storage

Workload Manager

Inform. Service

Job Adapter

Job Contr. - CondorG

JA: responsible for the final "touches" to the job before performing submission (e.g. creation of wrapper script, etc.)

SE characts & status

Computing Element

Storage Element

SCIENTIFIC COMPUTING LABORATORY

Network Daemon

LFC

submitted

waiting

ready

RB storage

Workload Manager

Inform. Service

Job

Job Contr. - CondorG

WMS

JC: responsible for the actual job management operations (done via CondorG)

CE characts & status

SE characts & status

Computing Element

Storage Element

SCIENTIFIC COMPUTING LABORATORY

Network Daemon

Workload Manager

RB storage

Job Contr. - CondorG

WMS

Input Sandbox files

Job

LFC

Inform. Service

CE characts & status

SE characts & status

Computing Element

Storage Element

submitted

waiting

ready

scheduled

SCIENTIFIC COMPUTING LABORATORY

Network Daemon

RB storage

Workload Manager

Job Contr. - CondorG

Input Sandbox

WMS

LFC

Inform. Service

submitted

waiting

ready

scheduled

running

Computing Element

Job

"Grid enabled" data transfers/ accesses

Storage Element

SCIENTIFIC COMPUTING LABORATORY

Network Daemon

Workload Manager

RB storage

Job Contr. - CondorG

WMS

LFC

Inform. Service

submitted

waiting

ready

scheduled

running

done

Output Sandbox files

Computing Element

Storage Element

glite-wms-get-output <jobID>

Network Daemon

RB storage

Workload Manager

Job Contr. - CondorG

Output Sandbox

WMS

LFC

Inform. Service

submitted

waiting

ready

scheduled

running

done

Computing Element

Storage Element

Output Sandbox files

RB storage

Network Daemon

Workload Manager

Job Contr. - CondorG

WMS

LFC

Inform. Service

submitted

waiting

ready

scheduled

running

done

cleared

Computing Element

Storage Element

Job monitoring

glite-wms-job-status <jobID>
glite-wms-job-logging-info <jobID>

LB: receives and stores job events; processes corresponding job status

Job status

Logging & Bookkeeping

LB proxy

Network Daemon

Workload Manager

Job Contr. - CondorG

WMS

Log of job events

Computing Element

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
     - SE: se-3.grid.box
     - CE: ce-1.grid.box:2119/jobmanager-lcgpbs-gridbox
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 !=
  "ce-1.grid.box:2119/jobmanager-lcgpbs-gridbox"
  ```

- **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

  ```
  other.GlueCEPolicyMaxCPUTime > 300;
  ```

- **GlueHostMainMemoryRAMSize** – memory size

  ```
  other.GlueHostMainMemoryRAMSize > 1024;
  ```

http://glite.web.cern.ch/glite/documentation/ → JDL specification

Examples 1

- 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)

Examples 2

- other.GlueHostMainMemoryRAMSize >=
  512 * (other.GlueHostArchitectureSMPSize > 0 ?
  other.GlueHostArchitectureSMPSize : 1)

  At least 512 MB of RAM memory per CPU core should be available

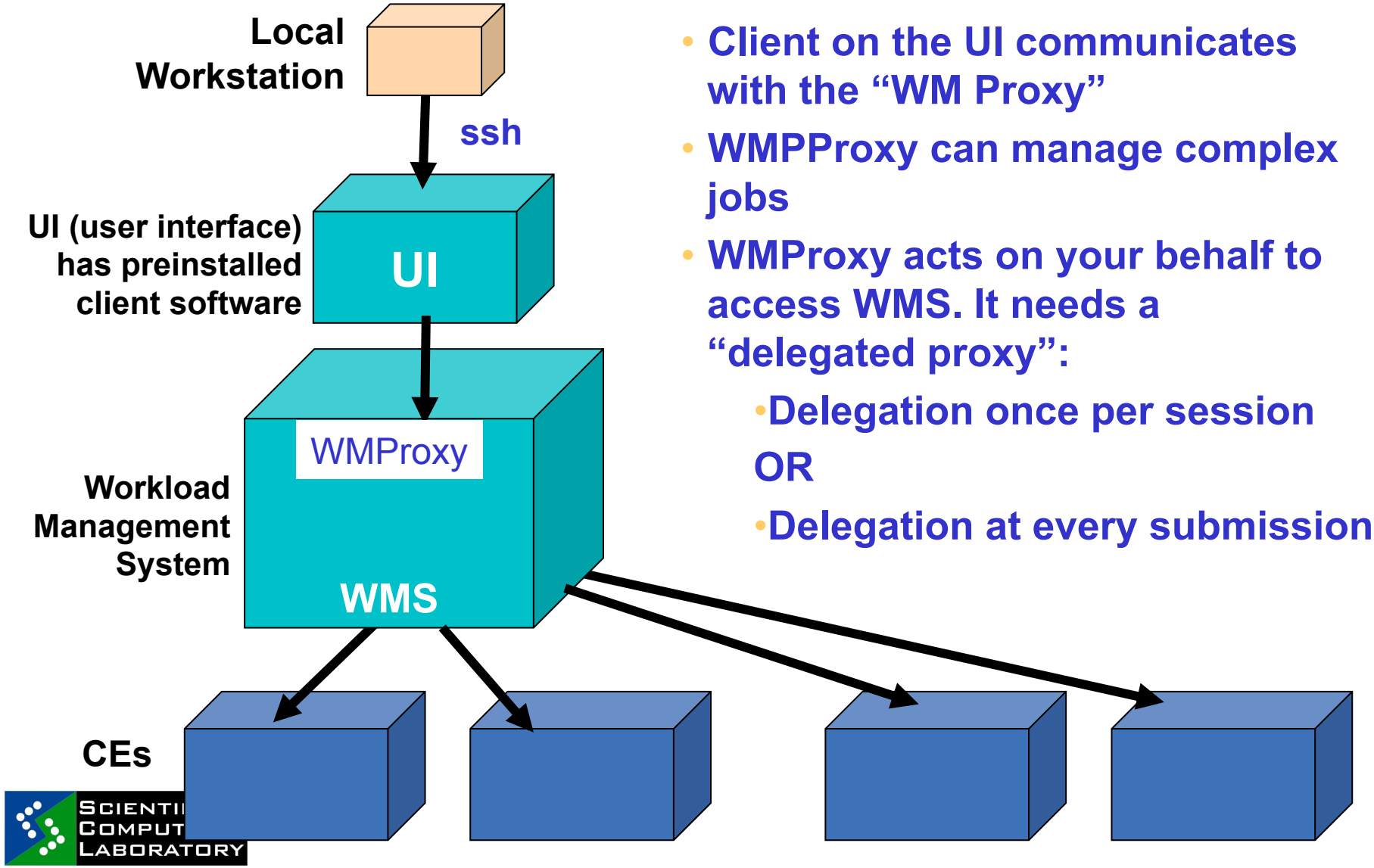
- other.GlueHostArchitecturePlatformType == "x86_64"

  x86_64 arch requested

Job states

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*

WMProxy (1)

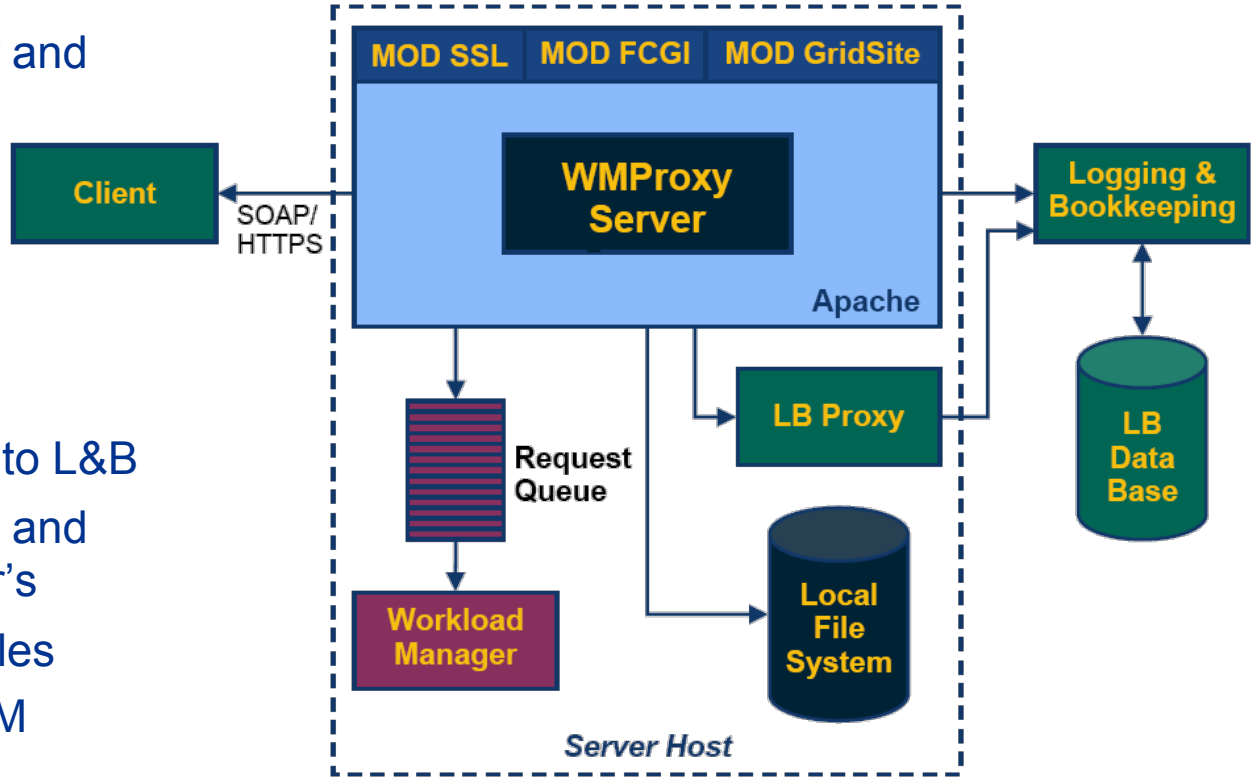


- Client on the UI communicates with the "WM Proxy"
- WMPProxy can manage complex jobs
- WMProxy acts on your behalf to access WMS. It needs a "delegated proxy":
  - Delegation once per session

  OR
  - Delegation at every submission

WMProxy (2)

- **WMProxy is a SOAP Web service providing access to the Workload Management System (WMS)**
- **Job characteristics specified via JDL**
  - jobRegister
    - create id
    - map to local user and create job dir
    - register to L&B
    - return id to user
  - input files transfer
  - jobStart
    - register sub-jobs to L&B
    - map to local user and create sub-job dir's
    - unpack sub-job files
    - deliver jobs to WM



Relevant JDL attributes 1

- **Type**
  - Job, DAG, Collection
- **JobType (only when the Type is set to Job!)**
  - Normal (sequential batch job), Parametric, Interactive, MPICH, ~~Checkpointable, Partitionable~~
- **Executable**
  - The name of the executable (absolute path)
- **Arguments**
  - Job command line arguments
- **StdInput, StdOutput, StdError**
  - Standard input/output/error of the job (stdin absolute path; stdout & stderr relative path)
- **Environment**
  - List of environment variables to be set for the binary
- **InputSandbox**
  - List of files on the UI local disk needed by the job for running
  - The listed files will be staged to the remote resource
- **OutputSandbox**
  - List of files, generated by the job, which have to be retrieved
  - Files will be transfered back

Relevant JDL attributes 2

- **Input Data**
  - For the broker, WMS does not transfer these files
- **Output Data**
  - For the broker, WMS does not transfer these files
- **Requirements**
  - Required CE caracteristics
- **Rank**
  - "Goodness" value for compatible CEs
- **ShallowRetryCount**
  - in case of grid error, retry job this many times
  - "Shallow": before job is running
- **RetryCount**
  - resubmit if the job failed in Running mode
  - If job fails after it has already done something (e.g. creating a Grid file) then resubmission can generate inconsistencies
- **MyProxyServer**
  - where to download proxy from in case of the existing proxy expires
  - Done by WMS

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";
         …; ],
   . . .
  };
]
```

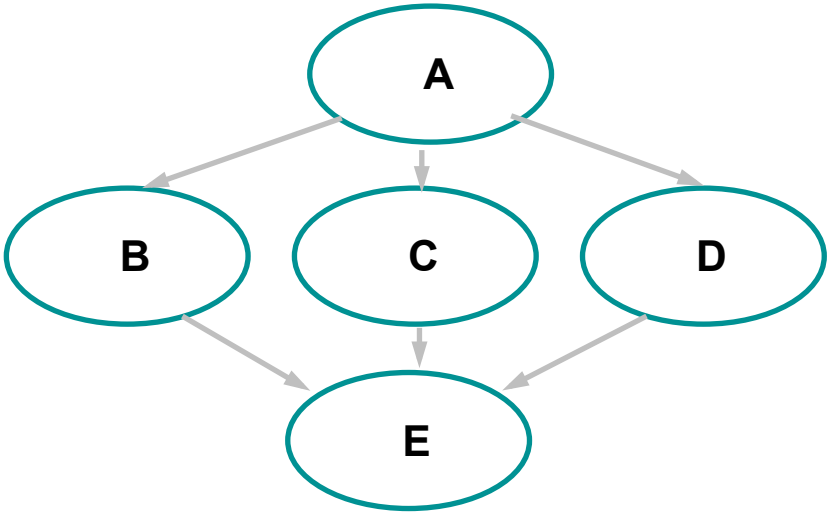
Transfer from UI only once

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 OutputSandbox in the next InputSandbox**
- Dependencies defined between pairs of jobs



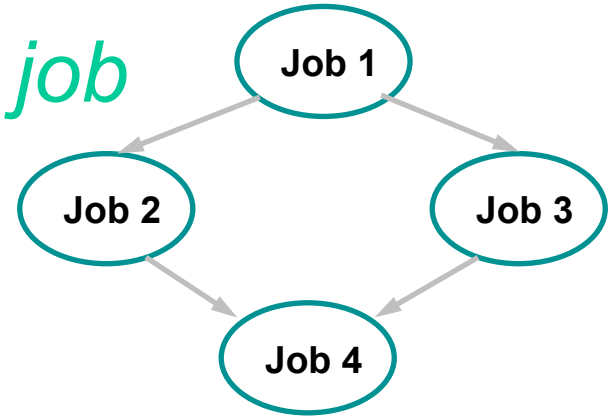
JDL of a DAG

```
[ 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



DAG: inheritance of sandboxes

```
[ Type = “DAG";
…
    job4 = [
          descpription = [
                JobType = "Normal";
                InputSandbox = {
root.nodes.job1.description.OutputSandbox[0],
root.nodes.job2.description.OutputSandbox,
                              ...};
                         ...;];
    ];
    . . .
  ]
```

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 1

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

   ParameterStart = 0;
   ParameterStep = 2;
   Parameters = 6;    → _PARAM_: 0, 2, 4, 6, 8, 10

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

JDL of a parametric job 2

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

    Parameters = {alpha, beta, gama};

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

Status of WMProxy

- **For simple jobs: glite-wms-… is *the* recommended way to use the WMS**
- **History:**
  - Before the **glite-wms-** commands we had **glite-** commands
    - used the WMS without WMProxy
  - Before the glite- commands we had
    - **edg-** commands (edg-job-submit….)
      - *European Data Grid – project before EGEE*
    - Used the “resource broker”
    - Still very widely used
  - You might see these commands still in use.
- **Status**
  - Complex jobs with WMProxy: first stable version just released. Not yet in routine production use
  - Watch for news!

SCIENTIFIC COMPUTING LABORATORY

Practicals on advanced job submission

- Create and submit a JDL file with different requirements and rankings
- Create and submit a JDL file for a collection of jobs
- Create and submit a JDL file for a parametric job
- Create and submit a JDL file for a DAG job