Enabling Grids for E-sciencE

www.eu-egee.org

Job Description Language – How to control your Job

Nadav Grossaug
IsraGrid
nadav@isragrid.org.il

Israel Academic Grid

EGEE is a project funded by the European Union under contract IST-2003-508833

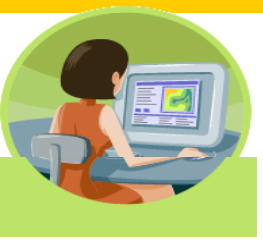
Outline

- Introduction
- Job submission services – what is really going on inside...
- JDL syntax

The use of jobs for running applications

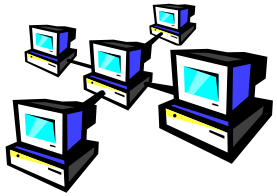
- Jobs are the way users execute applications on the grid.
- Information to be specified when a job has to be submitted:
  - Job characteristics
  - Job requirements and preferences on the computing resources
    - Also including software dependencies
  - Job data requirements
- Information specified using a Job Description Language (JDL)
  - Based upon Condor's *CLASSified ADvertisement language (ClassAd)*
    - Fully extensible language
    - A ClassAd is a sequence of attributes separated by semi-colon (;).

How does it work?
Main components

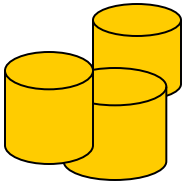


User Interface (UI): The place where users logon to the Grid

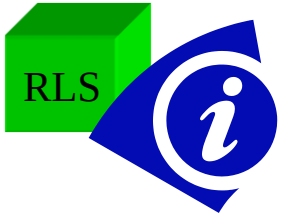
Resource Broker (RB): Matches the user requirements with the available resources on the Grid



Computing Element (CE): A batch queue on a farm of computers where the user Job gets executed



Storage Element (SE): A storage server where Grid files are stored (read/write/copy) or replicated.

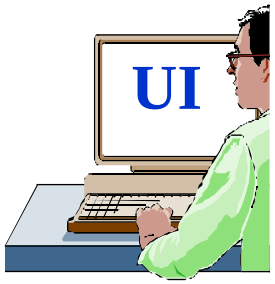


Catalogues(MDS/RLS): A server aiding in finding Grid files.

EGEE/LCG Workload Management System

- The user interacts with Grid via a **Workload Management System (WMS)**
- The Goal of WMS is the **distributed scheduling and resource management in a Grid environment**.
- What does it allow Grid 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

Job Submission



Network Server

Workload Manager

Job Contr. - CondorG

RB node

RLS

Inform. Service

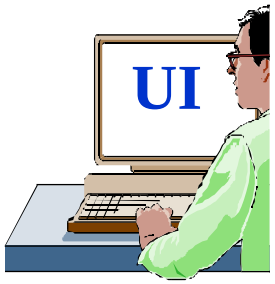
CE characts & status

SE characts & status

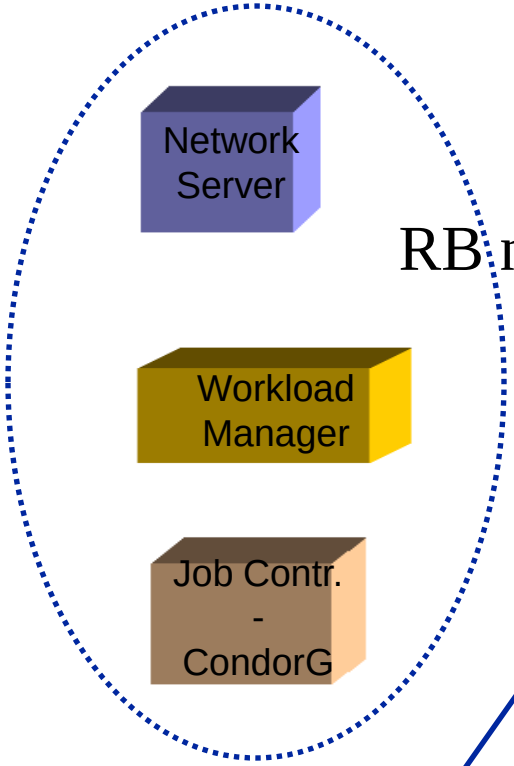
Computing Element

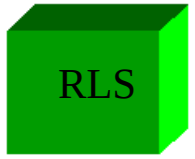
Storage Element

Job Submission



UI: allows users to access the functionalities of the WMS
(via command line, GUI, C++ and Java APIs)



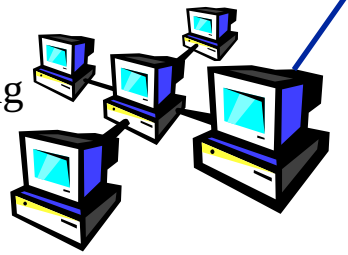


Job Status

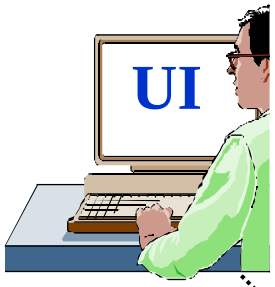
submitted

CE characts & status

SE characts & status



Job Submission



Network Server

RB node

Workload

RLS

Job Status

submitted

Inform. Service

glite-wms-job-submit –a myjob.jdl

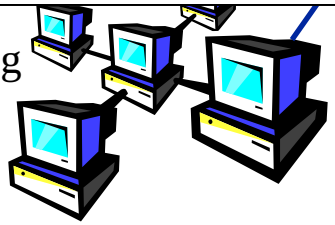
Myjob.jdl

```
JobType = "Normal";
Executable     = "$(CMS)/exe/sum.exe";
InputSandbox = {"/home/user/WP1testC","/home/file*", "/home/user/DATA/*"};
OutputSandbox = {"sim.err", "test.out", "sim.log"};
Requirements   = other. GlueHostOperatingSystemName == "linux" &&
other.GlueCEPolicyMaxWallClockTime > 10000;
Rank  = other.GlueCEStateFreeCPUs;
```

Job Description Language (JDL) to specify job characteristics and requirements

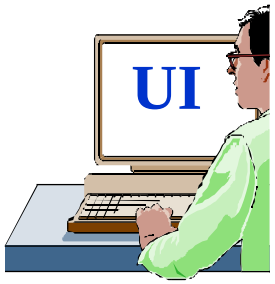
& status

Computing Element



Storage Element

Job Submission



Job

Input Sandbox files

Network Server

RB node

Workload Manager

RB storage

Job Contr. - CondorG

NS: network daemon responsible for accepting incoming requests

Inform. Service

Job Status

submitted

waiting

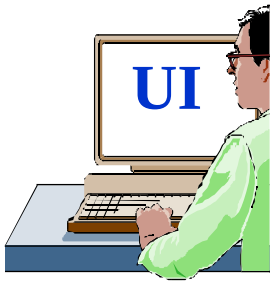
CE characts & status

SE characts & status

Computing Element

Storage Element

Job Submission



Network Server

Workload Manager

RB storage

Job Contr. - CondorG

RB node

RLS

Inform. Service

Job Status

submitted

waiting

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

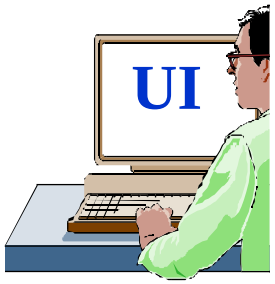
CE characts & status

SE characts & status

Computing Element

Storage Element

Job Submission



RB node

Network Server

Match-Maker/ Broker

Workload Manager

RB storage

Job Contr. - CondorG

Where must this job be executed ?

RLS

Inform. Service

Job Status

submitted

waiting

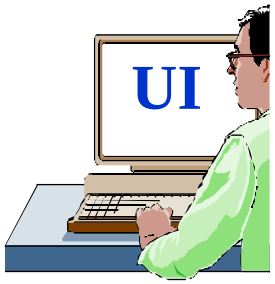
CE characts & status

SE characts & status

Computing Element

Storage Element

Job Submission



RB node

Network Server

Match-Maker/ Broker

Workload Man

RB storage

Job Con - CondorG

RLS

Inform. Service

Job Status

submitted

waiting

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

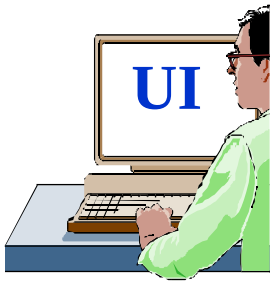
CE charact & status

SE charact & status

Computing Element

Storage Element

Job Submission



RB node

Network Server

Match-Maker/ Broker

Workload Manager

RB storage

Job Contr. - CondorG

RLS

Inform. Service

Job Status

submitted

waiting

What is the status of the Grid ?

CE characts & status

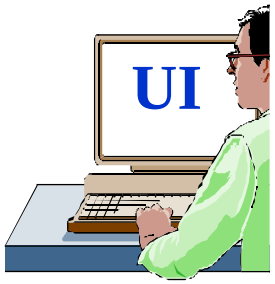
SE characts & status

Computing Element

Storage Element

Where are (which SEs) the needed data ?

Job Submission



RB node

Network Server

Match-Maker/ Broker

Workload Manager

CE choice

RB storage

Job Contr. - CondorG

RLS

Inform. Service

Job Status

submitted

waiting

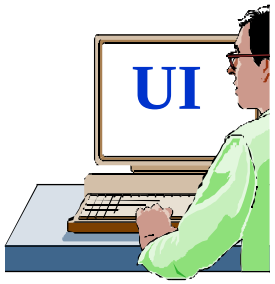
CE characts & status

SE characts & status

Computing Element

Storage Element

Job Submission



RB node

Network Server

Workload Manager

Job Adapter

Job Contr. -

RB storage

RLS

Inform. Service

Job Status

submitted

waiting

CE characts

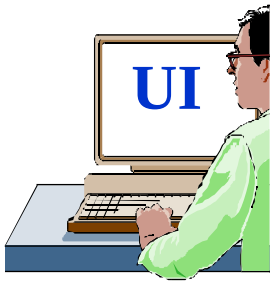
SE characts & status

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

Computing Element

Storage Element

Job Submission



RB node

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

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

Computing Element

RLS

Inform. Service

CE characts & status

SE characts & status

Storage Element

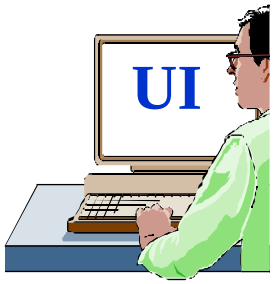
Job Status

submitted

waiting

ready

Job Submission



RB node

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

Input Sandbox files

Job

Computing Element

RLS

Inform. Service

CE characts & status

SE characts & status

Storage Element

Job Status

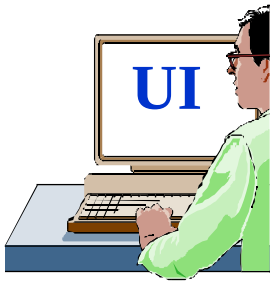
submitted

waiting

ready

scheduled

Job Submission



RB node

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

RLS

Inform. Service

Job Status

submitted

waiting

ready

scheduled

running

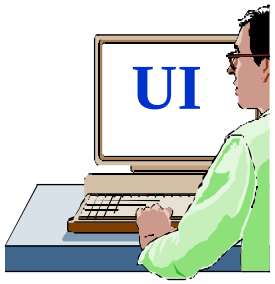
Computing Element

Job

"Grid enabled" data transfers/ accesses

Storage Element

Job Submission



RB node

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

Output Sandbox files

Computing Element

RLS

Inform. Service

Storage Element

Job Status

submitted

waiting

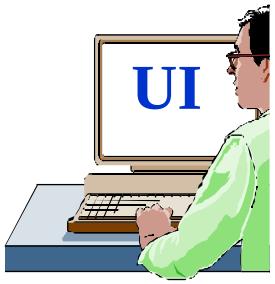
ready

scheduled

running

done

Job Submission



glite-wms-job-output <job-id>

RB node

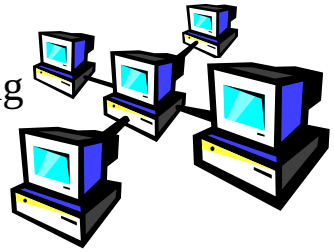
Network Server

Workload Manager

RB storage

Job Contr. - CondorG

Computing Element



RLS

Inform. Service

Storage Element

Job Status

submitted

waiting

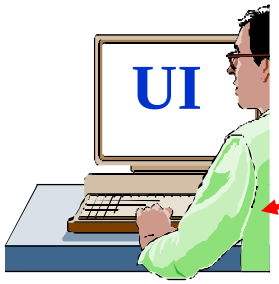
ready

scheduled

running

done

Job Submission



Output Sandbox files

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

RB node

Computing Element

RLS

Inform. Service

Storage Element

Job Status

submitted

waiting

ready

scheduled

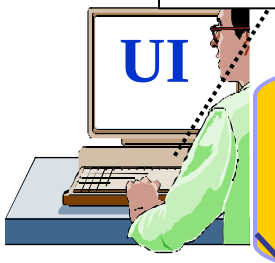
running

done

cleared

Job monitoring

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



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

Job status

Network Server

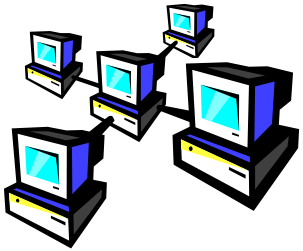
Workload Manager

RB node

Logging & Bookkeeping

Job Contr. - CondorG

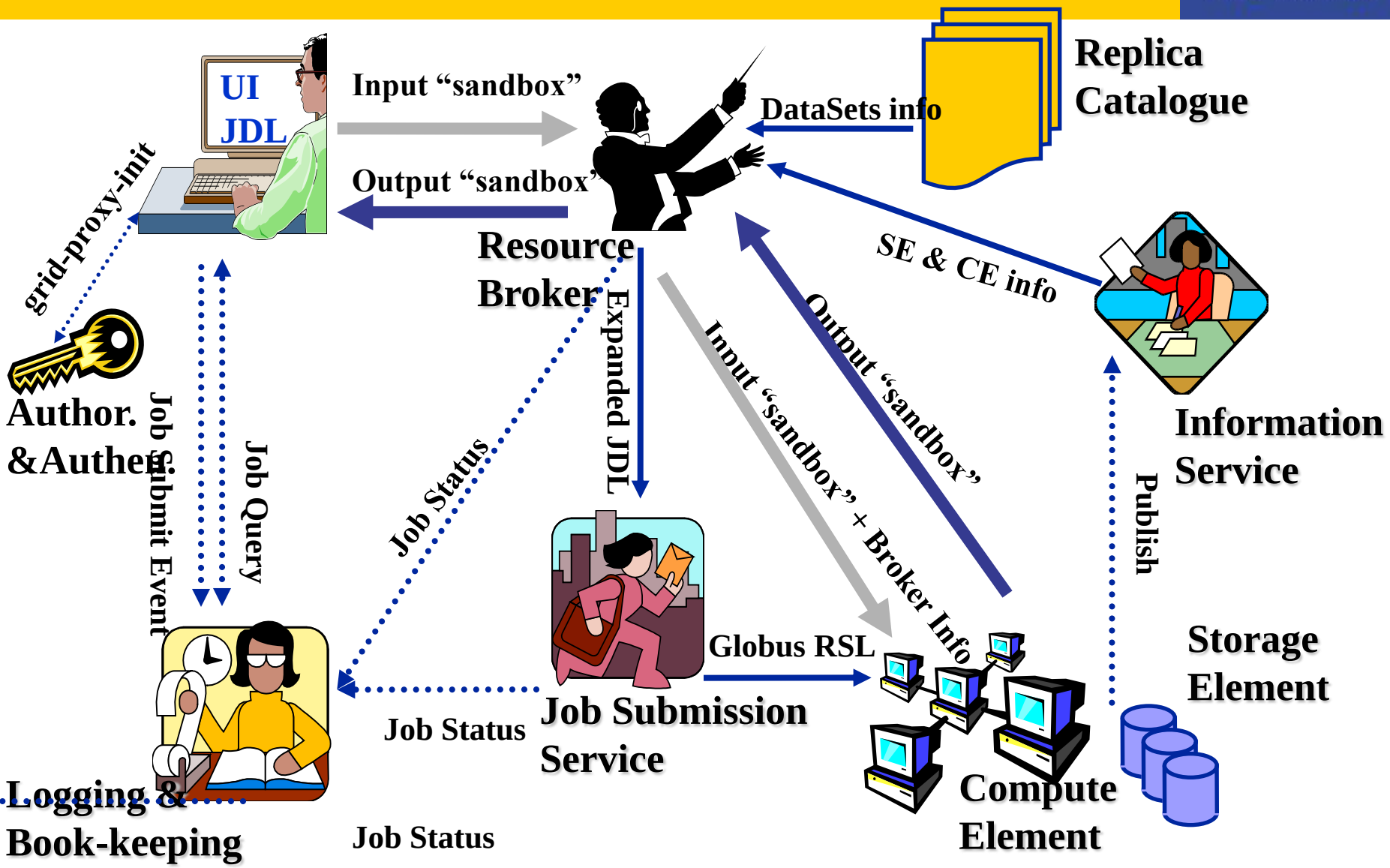
Computing Element



Log Monitor

LM: parses CondorG log file (where CondorG logs info about jobs) and notifies LB

A typical job workflow



Essential 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 (// or /* */)
- The JDL is sensitive to blank characters and tabs
  - they should not follow the semicolon (;) at the end of a line

Essential JDL

- The supported attributes are grouped in two categories:
  - **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 (see talk Job Services With Data Requirements)*
      - Input data to process, SE where to store output data, protocols spoken by application when accessing SEs

Essential JDL (contd.)

- **At least one has to specify the following attributes:**
  - the name of the executable
  - the files where to write the standard output and standard error of the job (recommended, not mandatory)
  - the arguments to the executable, if needed
  - the files that must be transferred from UI to WN and viceversa

```
[
Executable = "ls -al";
StdError = "stderr.log";
StdOutput = "stdout.log";
OutputSandbox = {"stderr.log", "stdout.log"};
]
```

Job Description Language: relevant attributes

- **Type** = Job (default) / DAG / Collection
- **JobType** = *Normal* (default) / *Interactive* / *MPICH* / *Parameteric*
- **Executable** (mandatory)
  - The command name (can reside on the UI or on the CE)
- **Arguments** (optional)
  - Job command line arguments
- **Environment (optional)**
  - List of environment settings
- **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. Can use regular expressions but cannot contain two files with the same name.
- **OutputSandbox** (optional)
  - List of files, generated by the job, which have to be retrieved (no RE)
- **VirtualOrganisation** (optional)
  - Specify the VO of the user

Job Description Language: relevant attributes

- **Requirements**
  - Some possible requirements values are below reported:
    - *other.GlueCEInfoLRMSType == "PBS" && other.GlueCEInfoTotalCPUs > 1* (the resource has to use PBS as the LRMS and whose WNs have at least two CPUs)
    - *Member("CMSIM-133", other.GlueHostApplicationSoftwareRunTimeEnvironment)* (a particular experiment software has to run on the resource and this information is published on the resource environment)
      - The *Member* operator tests if its first argument is a member of its second argument
    - *other.GlueCEPolicyMaxWallClockTime > 86000* (the job has to run for more than 86000 seconds, affects CE and queue selection)

Job Description Language: relevant attributes

- **Rank**
  - Expresses preference (how to rank resources that have already met the Requirements expression)
  - It is expressed as a floating-point number
  - The CE with the highest rank is the one selected
  - If not specified, default value defined in the UI configuration file is considered
    - Default: - *other.GlueCEStateEstimatedResponseTime* (the lowest estimated traversal time)
    - Default: *other.GlueCEStateFreeCPUs* (the highest number of free CPUs)
  - Other possible rank value is below reported:
    - *(other.GlueCEStateWaitingJobs == 0 ? other.GlueCEStateFreeCPUs : -other.GlueCEStateWaitingJobs)* (the number of waiting jobs is used if this number is not null and the rank decreases as the number of waiting jobs gets higher; if there are not waiting jobs, the number of free CPUs is used)

Other useful attributes

- **Perusal…** - a set of attributes enabling inspection of job output in run-time
- **DataRequirements, OutputSE** – used to specify the location of large input/output files for choosing CEs. Discussed further in data management
- **RetryCount, ShallowRetryCount** – the number of deep and shallow retries if a job fails.

Example of JDL file

```
[
JobType = “Normal”;
Executable = "$(CMS)/exe/sum.exe";
InputSandbox = {"/home/user/WP1testC","/home/file*”,
   "/home/user/DATA/*"};
OutputSandbox = {“sim.err”, “test.out”, “sim.log"};
Requirements  = (other. GlueHostOperatingSystemName ==
   “linux") && (other.GlueCEPolicyMaxWallClockTime >
   10000);
Rank = other.GlueCEStateFreeCPUs;
]
```

Job Submission

glite-wms-job-submit -a/-d delegID

[-r *<res_id>*] [--vo *<VO>*] [-o *<output file>*] *<job.jdl>*

-a/-d delegID – choose which way the job delegates the proxy (automatic on job submission/before job submission)

-r the job is submitted directly to the computing element identified by <*res_id*> (***not recommended***)

--vo the Virtual Organisation (overrides the one in jdl)

-o the generated jobId is written in the <*output file*>

Useful for batch job manipulations, e.g.:

glite-wms-job-status -i *<input file>*

Gets the status for all jobs specified in file

Possible job states

Flag	Meaning
SUBMITTED	submission logged in the LB
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 WN of the selected CE queue
DONE	job terminated without grid errors
CLEARED	job output retrieved
ABORT	job aborted by middleware, check ***reason***

Other (most relevant) UI commands

- **`glite-wms-job-list-match -a /-d delegID <job.jdl>`**
  - Lists resources matching a job description
  - Performs the matchmaking without submitting the job
- **`glite-wms-job-cancel <jobid>`**
  - Cancels a given job
- **`glite-wms-job-status <jobid>`**
  - Displays the status of the job
- **`glite-wms-job-output <jobid>`**
  - Returns the job-output (the OutputSandbox files) to the user
- **`glite-wms-job-logging-info <jobid>`**
  - Displays logging information about submitted jobs (all the events “pushed” by the various components of the WMS)
  - Very useful for debug purposes

Let's try it out!
(But questions first…)