Problemi di schedulazione distribuita su Grid

Ivan Porro
Università degli Studi di Genova, DIST, Laboratorio BioLab
pivan@unige.it
010-3532789
Riadattato da materiale realizzato da INFN Catania per il progetto “E-science grid facility for Europe and Latin America”
http://indico.eu-eela.eu/conferenceOtherViews.py?view=standard&amp;confId=125
Unit

1 Workload Management System
   • /* Example */
5

Workload Management System
Overview of EGEE gLite Middleware

Access Services
- Grid Access Service
- API

Security Services
- Authorization
- Authentication
- Auditing

Information & Monitoring Services
- Information & Monitoring
- Job Monitoring

Data Services
- Metadata Catalog
- Storage Element
- File & Replica Catalog
- Data Management

Accounting

Job Management Services
- Job Provenance
- Package Manager
- Computing Element
- Workload Management
- Site Proxy

Overview of EGEE gLite Middleware
The **Workload Management System (WMS)** is the gLite 3 component that allows users to submit jobs, and performs all tasks required to execute them, without exposing the user to the complexity of the Grid.

- **It is the responsibility of the user to describe his jobs and their requirements, and to retrieve the output when the jobs are finished.**

In the WLCG/EGEE Grid, two different workload management systems are deployed: the legacy LCG-2 system and the new system from the EGEE project, which is an evolution of the former and therefore has more functionalities.

In the following sections, we will describe the basic concepts of the language used to describe a job and the basic command line interface to submit and manage simple jobs.
Workload Management System

• Workload Management System (WMS) comprises a set of Grid middleware components responsible for distribution and management of tasks across Grid resources.

• Purpose of Workload Manager (WM) is accept and satisfy requests for job management coming from its clients meaning of the submission request is to pass the responsibility of the job to the WM.
  • **WM will pass the job** to an appropriate CE for execution taking into account requirements and the preferences expressed in the job description.

• The decision of which resource should be used is the outcome of a matchmaking process.
Job Description Language

- The Job Description Language (JDL) is a high-level language based on the Classified Advertisement (ClassAd) language, used to describe jobs and aggregates of jobs with arbitrary dependency relations.
  - The JDL is used in WLCG/EGEE to specify the desired job characteristics and constraints, which are taken into account by the WMS to select the best resource to execute the job.

- A job description is a file (called JDL file) consisting of lines having the format:

  attribute = expression;

- Expressions can span several lines, but only the last one must be terminated by a semicolon. Literal strings are enclosed in double quotes. If a string itself contains double quotes, they must be escaped with a backslash:

  Arguments = "\"hello\" 10";
Job Description Language

• The character “‘ ” cannot be used in the JDL.

• Comments must be preceded by a sharp character (#) or a double slash (//) at the beginning if each line.

• Multi-line comments must be enclosed between “/*” and “*/”.

Attention! The JDL is sensitive to blank characters and tabs. No blank characters or tabs should follow the semicolon at the end of a line.
Simple example

Executable = "/bin/hostname";
StdOutput = "std.out";
StdError = "std.err";

The **Executable** attribute specifies the command to be run by the job. If the command is already present on the WN, it must be expressed as an **absolute path**; if it has to be copied from the UI, only the file name must be specified, and the path of the command on the UI should be given in the **InputSandbox** attribute.

Executable = "test.sh";
InputSandbox = {"/home/doe/test.sh"};
StdOutput = "std.out";
StdError = "std.err";
• The Arguments attribute can contain a string value, which is taken as argument list for the executable:
  Arguments = "fileA 10";

• In the **Executable** and in the **Arguments** attributes it may be necessary to use special characters, such as &, \, |, >, <. These characters should be preceded by triple \ in the JDL, or specified inside quoted strings:
  Arguments = "-f file1\\\&file2";

• The attributes **StdOutput** and **StdError** define the name of the files containing the standard output and standard error of the executable, once the job output is retrieved.
• If files have to be copied from the UI to the execution node, they must be listed in the `InputSandbox` attribute:

```plaintext
InputSandbox = {"test.sh", .. , "fileN"};
```

• The files to be transferred back to the UI after the job is finished can be specified using the `OutputSandbox` attribute:

```plaintext
OutputSandbox = {"std.out","std.err"};
```

• The `InputSandbox` cannot contain two files with the same name, even if they have a different absolute path, as when transferred they would overwrite each other.
• The shell environment of the job can be modified using the `Environment` attribute.
  
  ```
  Environment = {
  "CMS_PATH=$HOME/cms",
  "CMS_DB=$CMS_PATH/cmdb"
  };
  ```

• The `VirtualOrganisation` attribute can be used to explicitly specify the VO of the user:
  
  ```
  VirtualOrganisation = "gilda";
  ```
• JobType
  
  - *Normal* (simple, sequential job), *Interactive*, *MPICH*, *Checkpointable*, *Partitionable*, *Parametric*

  - Or combination of them
    - Checkpointable, Interactive
    - Checkpointable, MPI
    - Interactive, MPI *NO!!!*

  JobType = “Interactive”;
  JobType = {“Interactive”, “Checkpointable”}
• The **Requirements** attribute can be used to express constraints on the resources where the job should run.
  
  – Its value is a Boolean expression that must evaluate to true for a job to run on that specific CE.

**Note:** *Only one Requirements attribute can be specified* (if there are more than one, only the last one is considered). If **several conditions** must be applied to the job, then **they all must be combined in a single Requirements attribute**.

• For example, let us suppose that the user wants to run on a CE using LSF as batch system (local scheduler), and whose Worker Nodes have at least two CPUs. He will write then in the job description file:

```
Requirements = other.GlueCEInfoLRMSType == "LSF" &&
other.GlueCEInfoTotalCPUs > 1;
```
• The WMS can be also asked to send a job to a particular queue in a CE with the following expression (here, PBS is another local scheduler, and the queue name is “jobmanager-pbs-short”):

\[
\text{Requirements} = \text{other.GlueCEUniqueID} == "lxshare0286.cern.ch:2119/jobmanager-pbs-short";
\]

• It is also possible to use regular expressions when expressing a requirement.

  – Let us suppose for example that the user wants all his jobs to run on any CE in the domain cern.ch This can be achieved in the JDL file using the following expression:

\[
\text{Requirements} = \text{RegExp("cern.ch", other.GlueCEUniqueID)};
\]

• The opposite can be required by using:

\[
\text{Requirements} = (!\text{RegExp("cern.ch", other.GlueCEUniqueID))};
\]
• If the job duration is significant, it is strongly advised to put a requirement on the maximum CPU time, or the wallclock time (expressed in minutes), needed for the job to complete.

  – For example, to express the fact that the job needs at least 8 CPU hours and 20 wallclock hours:

\[
\text{Requirements} = \text{other.GlueCEPolicyMaxCPUPTime} > 480 \quad \&\& \quad \text{other.GlueCEPolicyMaxWallClockTime} > 720;
\]

• It is possible to have the WMS automatically resubmitting jobs which, for some reason, are aborted by the Grid. The user can limit the number of times the WMS should resubmit a job by using the JDL attributes \texttt{RetryCount}.

\[
\text{RetryCount} = 7;
\]
\[
\text{RetryCount} = 0;
\]
The choice of the CE where to execute the job, among all the ones satisfying the requirements, is based on the rank of the CE, a quantity expressed as a floating-point number. The CE with the highest rank is the one selected.

- By default, the rank is equal to 
  `other.GlueCEStateEstimatedResponseTime`, where the estimated response time is an estimation of the time interval between the job submission and the beginning of the job execution.

- Rank = `other.GlueCEStateFreeCPUs`;
  which will rank best the CE with the most free CPUs.
Where are the information?

- **GLUE Schema**
- (Grid Laboratory for a Uniform Environment)
The WMProxy

• The WMProxy is the service responsible to provide access to the WMS functionality through a Web Service Interface

• The gLite WMProxy Server can be either accessed directly through the published WSDL, the C++ command line interface, or the API

• has been designed to efficiently handle a large number of requests for job submission and control to the WMS

  – it provides additional features such as bulk submission and the support for shared and compressed sandboxes for compound jobs.
gLite WMS Architecture
gLite WMS Architecture

Job management requests (submission, cancellation) expressed via a Job Description Language (JDL)
Finds an appropriate CE for each submission request, taking into account job requests and preferences, Grid status, utilization policies on resources.
Keeps submission requests

Requests are kept for a while if no resources are immediately available

gLite WMS Architecture
Repository of resource information available to matchmaker

Updated via notifications and/or active polling on resources
gLite WMS Architecture

Performs the actual job submission and monitoring
The Information Supermarket

• ISM represents one of the most notable improvements in the WM (not present in previous non-SOA WMS)

• The ISM basically consists of a repository of resource information that is available in read only mode to the matchmaking engine

• ISM update is the result of
  – the arrival of notifications
  – active polling of resources
  – some arbitrary combination of both
The Task Queue

- The Task Queue represents the second most notable improvement in the WM internal design
  - possibility to keep a submission request for a while if no resources are immediately available that match the job requirements
    - technique used by the AliEn and Condor systems, two other low level middlewares

- Non-matching requests
  - will be retried either periodically
    - eager scheduling approach
  - or as soon as notifications of available resources appear in the ISM
    - lazy scheduling approach
Job Submission Services

WMS components responsible to handle the job during its lifetime and performs the submission

• Job Adapter (JA)
  – is responsible for
    ▪ making the final touches to the JDL expression for a job, **before it is passed to CondorC for the actual submission**
    ▪ creating the job wrapper script that creates the appropriate execution environment in the CE worker node
      • transfer of the input and of the output sandboxes

• CondorC
  – responsible for
    ▪ performing the actual job management operations
      • job submission, job removal

• DAGMan (DAG → Direct Acyclic Graphs)
  – meta-scheduler
    ▪ purpose is to navigate the graph
    ▪ determine which nodes are free of dependencies
    ▪ follow the execution of the corresponding jobs
Job Submission Services

- Log Monitor (LM)
  - is responsible for
    - watching the CondorC log file
    - intercepting interesting events concerning active jobs

- Proxy Renewal Service
  - is responsible to assure that,
    - for all the lifetime of a job, a valid user proxy exists within the WMS
      - MyProxy Server is contacted in order to renew the user's credential

- Logging & Bookkeeping (LB)
  - is responsible to
    - Stores events generated by the various components of the WMS
    - Querying the LB user can retrieve information about the status of the job
Submitted job is entered by the user to the User Interface
Waiting job accepted and waiting for Workload Manager processing.
Jobs State Machine (3/9)

Ready job processed by WM but not yet transferred to the CE (local batch system queue).
Jobs State Machine (4/9)

Scheduled job waiting in the queue on the CE.
Running job is running.
**Done** job exited or considered to be in a terminal state by CondorC (e.g., submission to CE has failed in an unrecoverable way).
Aborted job processing was aborted by WMS (waiting in the WM queue or CE for too long, expiration of user credentials).
**Cancelled** job has been successfully canceled on user request.
Jobs State Machine (9/9)

*Cleared* output sandbox was transferred to the user or removed due to the timeout.
### Status Definition

<table>
<thead>
<tr>
<th>Status</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBMITTED</td>
<td>The job has been submitted by the user but not yet processed by the Network Server</td>
</tr>
<tr>
<td>WAITING</td>
<td>The job has been accepted by the Network Server but not yet processed by the Workload Manager</td>
</tr>
<tr>
<td>READY</td>
<td>The job has been assigned to a Computing Element but not yet transferred to it</td>
</tr>
<tr>
<td>SCHEDULED</td>
<td>The job is waiting in the Computing Element’s queue</td>
</tr>
<tr>
<td>RUNNING</td>
<td>The job is running</td>
</tr>
<tr>
<td>DONE</td>
<td>The job has finished</td>
</tr>
<tr>
<td>ABORTED</td>
<td>The job has been aborted by the WMS (e.g. because it took too long, or the proxy certificate expired, etc.)</td>
</tr>
<tr>
<td>CANCELLED</td>
<td>The job has been cancelled by the user</td>
</tr>
<tr>
<td>CLEARED</td>
<td>The Output Sandbox has been transferred to the User Interface</td>
</tr>
</tbody>
</table>
Delegating a proxy to WMProxy

• Each job submitted to WMProxy must be associated to a proxy credential previously delegated by the owner of the job to the WMProxy server.
  – This proxy is then used any time WMProxy needs to interact with other services for job related operations (e.g. submission to the CE, a GridFTP file transfer etc.)

  – There are two possible mechanisms to ask for a delegation of the user credentials:

    ▪ asking the “automatic” delegation of the credentials during the submission operation

    ▪ asking for an “explicit“ delegation
To explicitly delegate a user proxy to WMProxy, the command to use is:

```bash
glite-wms-job-delegate-proxy -d <delegID>
```

where `<delegID>` is a string chosen by the user.

For example, to delegate a proxy:

```
$ glite-wms-job-delegate-proxy -d mydelegID
```

Connecting to the service

https://rb102.cern.ch:7443/glite_wms_wmproxy_server

====== glite-wms-job-delegate-proxy Success ======

Your proxy has been successfully delegated to the WMProxy:

https://rb102.cern.ch:7443/glite_wms_wmproxy_server

with the delegation identifier: mydelegID

============================================
Submitting a simple job

• Starting from a simple JDL file, we can submit it via WMProxy by doing:

```
$ glite-wms-job-submit -d mydelegID test.jdl
Connecting to the service
https://rb102.cern.ch:7443/glite_wms_wmproxy_server

======== glite-wms-job-submit Success ========
The job has been successfully submitted to the WMProxy
Your job identifier is:
https://rb102.cern.ch:9000/vZKKk3gdBla6RySximq_vQ
```

==============================================
• The command returns to the user the job identifier (jobID), which uniquely defines the job and can be used to perform further operations on the job, like interrogating the system about its status, or canceling it.

• The format of the jobID is:
  https://<LB_hostname>[:<port>]/<unique_string>

• where <LB hostname> is the host name of the Logging and Bookkeeping (LB) server for the job, which usually sits on the WMS used to submit the job.
• To submit jobs via WMProxy, it is required to have a valid VOMS proxy (a certificate initialized with VO extensions), otherwise the submission will fail with an error like the following:

Error - Operation failed
Unable to delegate the credential to the endpoint: https://rb102.cern.ch:7443/glite_wms_wmproxy_server
User not authorized:
unable to check credential permission
(/opt/glite/etc/glite_wms_wmproxy.gacl)
(credential entry not found)
credential type: person
input dn: /C=CH/O=CERN/OU=GRID/CN=John Doe
The `-o <file path>` option allows users to specify a file to which the jobID of the submitted job will be appended. This file can be given to other job management commands to perform operations on more than one job with a single command, and it is a convenient way to keep trace of one’s jobs.

The `-r <CEId>` option is used to directly send a job to a particular CE. If used, the match making will not be carried out.
- The drawback is that the BrokerInfo file, which provides information about the evolution of the job, will not be created, and therefore the use of this option is discouraged.
Computing Element

• A CE is identified by <CEId>, which is a string with the following format:

  \(<\text{CE hostname}>:<\text{port}>/\text{jobmanager-}<\text{service}>-\<\text{queue}>\)

• where <CE hostname> and <port> are the host name of the machine and the port where the Grid Gate is running

• <queue> is the name of one of the corresponding LRMS queue

• <service> is the LRMS type, such as LSF, PBS, condor.

• E.g.:

  \(\text{adc0015.cern.ch:2119/jobmanager-lcgpbs-infinite}\)
Listing CE(s) matching a job

• It is possible to see which CEs are able to run a job described by a given JDL using:

```
$ glite-wms-job-list-match -d mydelegID --rank test.jdl
Connecting to the service https://rb102.cern.ch:7443/glite_wms_wmproxy_server

COMPUTING ELEMENT IDs LIST
The following CE(s) matching your job requirements have been found:

*CEId*                        *Rank*
- CE.pakgrid.org.pk:2119/jobmanager-lcgpbs-cms 0
- grid-ce0.desy.de:2119/jobmanager-lcgpbs-cms -10
- gw-2.ccc.ucl.ac.uk:2119/jobmanager-sge-default -56
- grid-ce2.desy.de:2119/jobmanager-lcgpbs-cms -107
```
Retrieving the status of a job

$ glite-wms-job-status
  https://rb102.cern.ch:9000/fNdD4FW_Xxkt2s2aZJeoeg
************************************************************************
BOOKKEEPING INFORMATION:
Status info for the Job :
  https://rb102.cern.ch:9000/fNdD4FW_Xxkt2s2aZJeoeg
Current Status: Done (Success)
Exit code: 0
Status Reason: Job terminated successfully
Destination: cel.inrne.bas.bg:2119/jobmanager-lcgpbs-cms
Submitted: Mon Dec 4 15:05:43 2006 CET
************************************************************************

• The verbosity level controls the amount of information provided. The value of the -v option ranges from 0 to 3.

• The commands to get the job status can have several jobIDs as arguments, i.e.: glite-wms-job-status <jobID1> ... or, more conveniently, the -i <file path> option can be used to
• The **--noint** option suppresses the interactivity and all the jobs are considered.
• If the **--all** option is used instead, the status of all the jobs owned by the user submitting the command is retrieved.
• The **--from / --to [MM:DD:]hh:mm[:[CC]YY]** options make the command query for jobs that were submitted after/before the specified date and time.
• The **--status <status>** option makes the command retrieve only the jobs that are in the specified status.
• The **--exclude <status>** option makes it retrieve jobs that are not in the specified status.
• The option **--o <file path>** redirect the command output to a file.
Cancelling a job

```bash
glite-wms-job-cancel
    https://rb102.cern.ch:9000/P1c60RFsrIZ9mnBALa7yZA
Are you sure you want to remove specified job(s) [y/n]? y
Connecting to the service
https://128.142.160.93:7443/glite_wms_wmproxy_server
========== glite-wms-job-cancel Success ===========
The cancellation request has been successfully submitted for the following job(s):
- https://rb102.cern.ch:9000/P1c60RFsrIZ9mnBALa7yZA
==============================================

If the cancellation is successful, the job will terminate in status CANCELLED
Retrieving the output(s)

$ glite-wms-job-output
   https://rb102.cern.ch:9000/yabp72aERhofLA6W2-LrJw

Connecting to the service
https://128.142.160.93:7443/glite_wms_wmproxy_server

JOB GET OUTPUT OUTCOME
Output sandbox files for the job:
https://rb102.cern.ch:9000/yabp72aERhofLA6W2-LrJw
have been successfully retrieved and stored in the directory:
/tmp/doe_yabp72aERhofLA6W2-LrJw

The default location for storing the outputs (normally /tmp) is defined in the UI configuration, but it is possible to specify in which directory to save the output using the --dir <path name> option.
‘Scattered’ Input Sandboxes

- A new feature introduced by the gLite WMS is the possibility to indicate input sandbox files stored not on the UI, but on a GridFTP server, and, similarly, to specify that output files should be transferred to a GridFTP server when the job finishes.

```java
InputSandbox = {"gsiftp://lxb0707.cern.ch/cms/fileA", "fileB"};
```

- It is also possible to specify a base GridFTP URI with the attribute `InputSandboxBaseURI`
  - files expressed as simple file names or as relative paths will be looked for under that base URI.

```java
InputSandbox = {"fileA", "data/fileB", "file:///home/doe/fileC"};
```

```java
InputSandboxBaseURI = "gsiftp://lxb0707.cern.ch/cms/doe";
```
Storing output files in a GridFTP Server

• In order to store the output sandbox files to a GridFTP server, the `OutputSandboxDestURI` attribute must be used together with the usual `OutputSandbox` attribute.
  - `OutputSandbox` is used to list the output files created by the job in the WN to be transferred.
  - `OutputSandboxDestURI` is used to express where the output files are to be transferred.

```
OutputSandbox = {"fileA", "data/fileB", "fileC"};
```

```
OutputSandboxDestURI =
  {"gsiftp://lxb0707.cern.ch/cms/doe/fileA",
   "gsiftp://lxb0707.cern.ch/cms/doe/fileB","fileC"};
```

– where the first two files have to be copied to a GridFTP server, while the third file will be copied back to the WMS with the usual mechanism. Clearly, glite-wms-job-output will retrieve only the third file.
Another possibility is to use the `OutputSandboxBaseDestURI` attribute to specify a base URI on a GridFTP server where the files listed in `OutputSandbox` will be copied.

```
OutputSandbox = {"fileA", "fileB"};
```

```
OutputSandboxBaseDestURI =
   "gsiftp://lxb0707.cern.ch/cms/doe/";
```

will copy both files under the specified GridFTP URI.

**Note:** the directory on the GridFTP where the files have to be copied must already exist.
‘Compressed’ Sandboxes

• A compressed archive is created with the input sandboxes files using **libtar** and **zlib** libraries
  – This is done automatically by WMProxy client commands
  – this mechanism can be enabled/disabled by the user through the JDL (**AllowZippedISB** attribute)

• The archive is transferred (instead of single files) to the WMS
  – If the sandbox is composed by several files, besides the gain brought by compression, this method allows saving the overhead for several calls to **globus-url-copy**

• WMProxy service untars the files in the jobs directories when the job is ‘started’ and removes the archive
Real Time Output Retrieval

- The user can enable the job perusal by setting the attribute `PerusalFileEnable` to true in the job JDL.
  - This makes the WN to upload, at regular time intervals (defined by the `PerusalTimeInterval` attribute and expressed in seconds), a copy of the output files specified using the `glite-wms-job-perusal` command to the WMS machine (by default), or to a GridFTP server specified by the attribute `PerusalFilesDestURI`.

```plaintext
Executable = "job.sh";
StdOutput = "stdout.log";
StdError = "stderr.log";
InputSandbox = {"job.sh"};
OutputSandbox = {"stdout.log","stderr.log","testfile.txt"};
PerusalFileEnable = true;
PerusalTimeInterval = 30;
RetryCount = 0;
```
After the job has been submitted with glite-wms-job-submit, the user can choose which output files should be inspected:

```bash
$ glite-wms-job-perusal --set -f stdout.log -f stderr.log -f testfile.txt
https://wms104.cern.ch:9000/B02xR3EQg9ZHHoRc-1nJkQ
```

Connecting to the service

https://128.142.160.93:7443/glite_wms_wmproxy_server

Connecting to the service

https://128.142.160.93:7443/glite_wms_wmproxy_server

=============== glite-wms-job-perusal Success ================
Files perusal has been successfully enabled for the job:
https://wms104.cern.ch:9000/B02xR3EQg9ZHHoRc-1nJkQ

============================================
.. and, when the job starts, the user can see one output file:

```
$ glite-wms-job-perusal --get -f testfile.txt \
https://wms104.cern.ch:9000/B02xR3EQg9ZHHoRc-1nJkQ

Connecting to the service
https://137.138.45.79:7443/glite_wms_wmproxy_server

Connecting to the service
https://137.138.45.79:7443/glite_wms_wmproxy_server

========== glite-wms-job-perusal Success ===========
The retrieved files have been successfully stored in:
/tmp/doe_OoDVmWCAhx_HiSPvASGsg

====================================================================
https://grid.ct.infn.it/twiki/bin/view/GILDA/SimpleJobSubmission
WMProxy User’s guide

JDL Attributes Specification

gLite User’s guide
- https://edms.cern.ch/file/722398/1.2/gLite-3-UserGuide.pdf