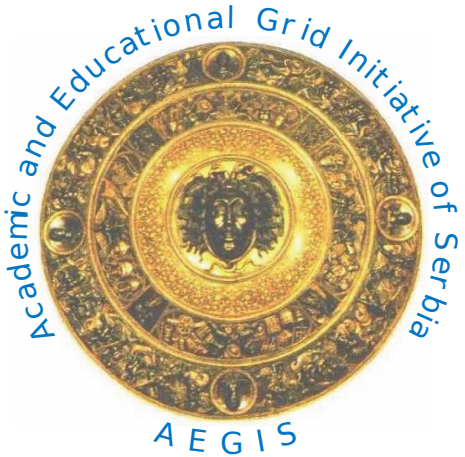
How to use computing resources at Grid

Nikola Grkic

ngrkic@ipb.ac.rs

Scientific Computing Laboratory

Institute of Physics Belgrade, Serbia



SEE-GRID-SCI

SEE-GRID eInfrastructure for regional eScience

Oct. 07, 2009

www.eu-egee.org

- **JDL**

  The Job Description Language (JDL) is a high-level language used to describe jobs and aggregates of jobs with arbitrary dependency relations

- **Simple example**

```
[
Type = "Job";
Executable = "/bin/hostname";
Arguments = "";
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox =   {"std.out","std.err"};
Requirments = “”;
]
```

- **Additional attributes**

```
InputSandbox = {"test.sh", "fileA", "fileB", ...}
InputSandbox = {
"gsiftp://lxb0707.cern.ch/cms/doe/data/fileA”,"fileB"};
VirtualOrganisation = "cms”;
RetryCount = 0;
MyProxyServer = "myproxy.ipb.bg.ac.rs"
```

- **Requirements**

```
Requirements =RegExp("ce64.ipb.bg.ac.rs*”,other.GlueCEUniqueID);
Requirements = Member("VO-cms-CMSSW_2_0_0",
other.GlueHostApplicationSoftwareRunTimeEnvironment);
Requirements = (other.GlueHostArchitecturePlatformType ==
"x86_64”);
```

- **Single Job Submission**

```
glite-wms-job-list-match -a <jdl file>
glite-wms-job-delegate-proxy -d <delegID>
glite-wms-job-submit -a <jdl file>
glite-wms-job-status <jobID>
glite-wms-job-cancel <jobID>
glite-wms-job-output <jobID>
glite-wms-job-logging-info <jobID>
```

Simple job example

```
Executable = "test.sh";
Arguments = "fileA  fileB";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {"test.sh", "fileA", "fileB"};
OutputSandbox = {"std.out", "std.err"};
```

http://wiki.ipb.ac.rs/index.php/Submiting_jobs

- **Job collection**

Job1.jdl

```
Executable = "/bin/hostname";
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out","std.err"};
```

Job2.jdl

```
Executable = "/bin/date";
StdOutput = "std2.out";
StdError = "std2.err";
OutputSandbox = {"std2.out","std2.err"};
```

- **Parametric Job**

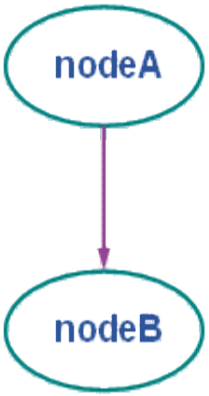
```
[
 JobType = "Parametric";
 Executable = "/bin/sh";
 Arguments = "message_PARAM_.sh";
 InputSandbox = "message_PARAM_.sh";
 Parameters= 6;
 ParameterStep = 2;
 ParameterStart = 0;
 StdOutput = "myoutput_PARAM_.txt";
 StdError = "myerror_PARAM_.txt";
 OutputSandbox  = {"myoutput_PARAM_.txt",
 "myerror_PARAM.txt"};
 ShallowRetryCount = 1;]
]
```

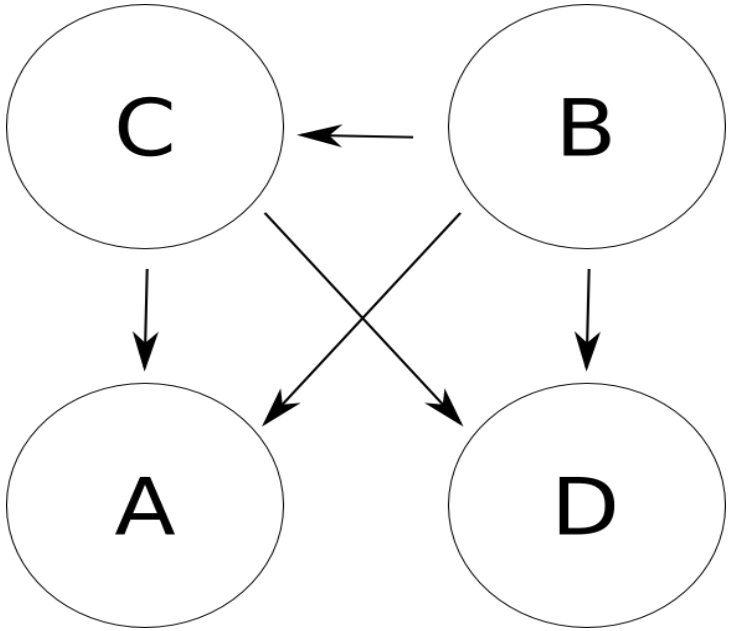
- **MPI job**

```
Type = "Job";
JobType = "MPICH";
CpuNumber = 2;
Executable = "test-mpi.sh";
Arguments = "test-mpi";
StdOutput = "test-mpi.out";
StdError = "test-mpi.err";
InputSandbox = {"test-mpi.sh","test-mpi.c"};
OutputSandbox = {"test-mpi.err","test-mpi.out",
"mpiexec.out"};
Requirements = Member("MPICH");
```

DAG (directed acyclic graph) Jobs

Dependencies = {{nodeA,nodeB},{nodeA,nodeC},{{nodeB,nodeC},nodeD}





Interactive Job

```
JobType = "Interactive" ;
Executable = "interactive.sh" ;
InputSandbox ={"interactive.sh"} ;
```

```
#!/bin/sh
 echo "Welcome!"
 echo "Please tell me your name: "
 read name
 echo "That is all, $name."
 echo "Bye bye."
 read name2
 echo "Go away, $name2."
 exit 0
```

Advanced job examples