Advanced use of Workload Management System

Emidio Giorgio
INFN Catania

Bologna, 13 aprile 2007

Tutorial CYCLOPS

Outline

- **MPI Jobs**
- **WMProxy Overview**
  - **Special Jobs**
    - DAG jobs
    - Job collections
    - Parametric jobs

MPI Overview

- **Execution of parallel jobs is an essential issue for modern informatics and applications.**
- **Most used library for parallel jobs support is MPI (Message Passing Interface)**
- **At the state of the art, parallel jobs can run inside single Computing Elements (CE) only;**
  - several projects are involved into studies concerning the possibility of executing parallel jobs on Worker Nodes (WNs) belonging to different CEs.
- The source code must have been compiled with mpicc libraries

MPI JDL

- > $ cat mpi.jdl

```
[
 Type = "Job";
#Mandatory
 JobType = "MPICH";
#Bash script containing invocation to MPI
#interpreter, mpirun
 Executable = "MPIscript.sh";
#The number of CPU that will be used
 NodeNumber = 2;
 StdOutput = "cpi.out";
 StdError = "cpi.err";
 InputSandbox = {"cpi","MPIscript.sh"};
 OutputSandbox = {"cpi.err","cpi.out"};
 RetryCount = 3;
]
```

MPI jobs @ work

- Login to the UI :
  - **ssh bolognaXX@glite-tutor.ct.infn.it**
- initialize a proxy if you haven't it
  - **voms-proxy-init --voms gilda**
- **cd examples**
- **edg-job-submit mpi.jdl**
- Monitor job status and finally get the output using the job id given by the submit command

CYCLOPS

Workload Manager Proxy

Information Society and Media

- **WMProxy (Workload Manager Proxy)**
  - is a new service providing access to the gLite Workload Management System (WMS) functionality through a simple Web Services based interface.
  - has been designed to handle a large number of requests for job submission
    - gLite 3.0=> ~180 secs for 500 jobs
    - goal is to get in the short term to ~60 secs for 1000 jobs
  - it provides additional features such as *bulk submission* and the support for *shared and compressed* sandboxes for *compound jobs*.
  - It's the natural replacement of the NS in the passage to the *SOA approach*.

New request types

- **Support for new types strongly relies on newly developed JDL converters and on the DAG submission support**
  - all JDL conversions are performed on the server
  - a single submission for several jobs
- **All new request types can be monitored and controlled through a single request id**
  - each sub-jobs can be however followed-up and controlled independently through its own id
- **“Smarter” WMS client commands/API**
  - allow submission of DAGs, collections and parametric jobs exploiting the concept of “shared sandbox”
  - allow automatic generation and submission of collections and DAGs from sets of JDL files located in user specified directories on the UI

CYCLOPS

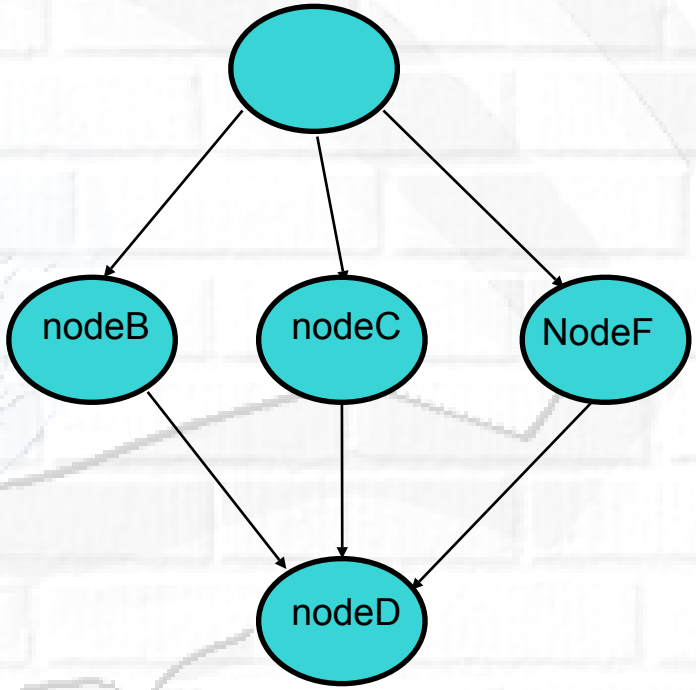
Special Jobs

- DAG
- Job Collection
- Parametric jobs

Information Society and Media

DAG job

- A DAG job is a set of jobs where input, output, or execution of one or more jobs can depend on other jobs
- Dependencies are represented through **D**irected **A**cyclic **G**raphs, where the nodes are jobs, and the edges identify the dependencies



DAG -- JDL structure

- Type = "DAG" → *Mandatory*
- VirtualOrganisation = "yourVO" → *Mandatory*
- Max_Nodes_Running = int >0 → *Optional*
- MyProxyServer = "…" → *Optional*
- Requirements = "…" → *Optional*
- Rank = "…" → *Optional*
- InputSandbox =  more later! → *Optional*
- ~~OutSandbox = "…"~~
- Nodes = nodeX  more later! → *Mandatory*
  Dependencies  more later! → *Mandatory*

Attribute: Nodes

CYCLOPS

The *Nodes* attribute is the core of the DAG description;

```
....
Nodes = [  nodefilename1 = [...]
           nodefilename2 = [...]
           .......

           dependencies = …

]
```

```
Nodefilename1 = [ file = "foo.jdl"; ]
Nodefilename2 =
  [   file = "/home/vardizzo/test.jdl";
      retry = 2;          ]
```

```
Nodefilename1 = [
    description = [ JobType = "Normal";
                    Executable = "abc.exe";
                    Arguments = "1 2 3";
                    OutputSandbox = […];
                    InputSandbox = […];
                    ….. ]
    retry = 2;
                ]
```

Information Society and Media

Attribute: Dependencies

- It is a list of lists representing the dependencies between the nodes of the DAG.

```
....
Nodes = [  nodefilename1 = [...]
           nodefilename2 = [...]
           .......

           dependencies = ...

]
```

```
dependencies =
        {nodefilename1, nodefilename2}
```

MANDATORY : YES!

```
dependencies = {};
```

```
{ nodefilename1, nodefilename2 }
```

```
{ { nodefilename1, nodefilename2 }, nodefilename3 }
```

```
{ { { nodefilename1, nodefilename2}, nodefilename3}, nodefilename4 }
```

DAG jdl

```
[
  type = "dag";
  max_nodes_running = 4;
  nodes = [
    nodeA = [
      file ="nodes/nodeA.jdl" ;
    ];
    nodeB = [
      file ="nodes/nodeB.jdl" ;
    ];
    nodeC = [
      file ="nodes/nodeC.jdl"
    ];
    nodeD = [
      file ="nodes/nodeD.jdl";
    ];
   dependencies = {
      {nodeA, nodeB},
      {nodeA, nodeC},
      { {nodeB,nodeC}, nodeD }
    }
  ];
]
```

Node description could also be done here, instead of using separate files

Job Collection

- A job collection is a set of independent jobs that user wants to submit and monitor via a single request
- Jobs of a collection are submitted as DAG nodes without dependencies
- JDL is a list of classad, which describes the subjobs

```
[
        Type = "collection";
        VirtualOrganisation = “gilda";
        nodes = {
                [  <job descr 1 >],
                [  <job descr 2 >],
                 …
               };
]
```

‘Scattered’ Input Sandboxes

- **Input Sandbox can contain**
  - file paths on the UI machine (i.e. the usual way)
  - pointer to other files within the DAG/collection
  - URI pointing to files on a remote gridFTP/HTTPS server

```
InputSandbox = {
    "gsiftp://neo.datamat.it:2811/var/prg/sim.exe",
     root.nodes.nodeA.description.OutputSandbox[0],
    "file:///home/pacio/myconf“ };
```

- **Only local files (`file://`) are uploaded to the WMS node**
- **File pointed by URIs are directly downloaded on the WN by the JobWrapper just before the job is started**

‘Scattered’ Output Sandboxes

- JDL is enriched with attributes, specifying the destinations for the files listed within OutputSandbox attribute list

```
OutputSandbox = {      "jobOutput","run1/event1",
                                    "jobError"                    };
OutputSandboxDestURI = {
                "gsiftp://matrix.datamat.it/var/jobOutput",
                "https://grid003.ct.infn.it:8443/home/cms/event1",
                "gsiftp://matrix.datamat.it/var/jobError"         };
```

- A base URI to be applied to all sandbox files can also be specified

```
OutputSandboxBaseDestURI = "gsiftp://neo.datamat.it/home/run1/";
```

- Files are copied when the job has completed execution by the JobWrapper to the specified destination without transiting on the WMS node

Job collection example

```
[
  type = "collection";
  InputSandbox = {"date.sh"};
  RetryCount = 3;
  nodes = {
     [
       file ="jobs/job1.jdl" ;
     ],
     [
        [
           Executable = "/bin/sh";
           Arguments = "date.sh";
           StdOutput = "date.out";
           StdError =  "date.err";
           OutputSandbox ={"date.out", "date.err"};
         ]
     ],
     [
       file ="jobs/job3.jdl" ;
     ]
  };
]
```

All nodes will share this Input Sandbox

Parametric Job

- A parametric job is a job where one or more of its attributes are parameterized
- Values of attributes vary according to a parameter

```
[
  JobType = "Parametric";
  Executable = "/bin/sh";
  Arguments = "md5.sh input_PARAM_.txt";
  InputSandbox = {"md5.sh", "input_PARAM_.txt"};
  StdOutput = "out_PARAM_.txt";
  StdError = "err_PARAM_.txt";
  Parameters = 4;
  ParameterStart = 1;
  ParameterStep = 1;
  OutputSandbox = {"out_PARAM_.txt", "err_PARAM_.txt"};
]
```

- Job monitoring / managing is always done through a unique jobID, as if the job was single (see submissio of collection

Parametric job / 2

- Parameter can be also a list of string
- InputSandbox (if present) has to be coherent with parameters

```
[ui-test] /home/giorgio/param > cat param2.jdl
[
      JobType = "Parametric";
              Executable = "/bin/cat";
               Arguments = "input_PARAM_.txt";
              InputSandbox = "input_PARAM_.txt";
              StdOutput = "myoutput_PARAM_.txt";
              StdError = "myerror_PARAM_.txt";
              Parameters = {earth,moon,mars};
              OutputSandbox  = {"myoutput_PARAM_.txt"};
]

[ui-test] /home/giorgio/param > ls
inputEARTH.txt  inputMARS.txt  inputMOON.txt param2.jdl
```

References

- gLite 3.0 User Guide
  - https://edms.cern.ch/file/722398/1.1/gLite-3-UserGuide.pdf
- JDL attributes specification for WM proxy
  - https://edms.cern.ch/document/590869/1
- WMProxy quickstart
  - http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/wmproxy_client_quickstart.shtml
- WMS user guides
  - https://edms.cern.ch/document/572489/1
- MPI
  - http://www-unix.mcs.anl.gov/mpi/