The gLite Workload Management System

M. Cecchi, A. Ghiselli, F. Giacomini, A. Gianelle,
A. Guarise, A. Maraschini, S. Monforte,
S. Pellegrini, L. Perini, L. Petronzio, M. Sgaravatto

3rd EGEE User Forum
11-14 February, 2008 - Clermont-Ferrand

www.eu-egee.org

Information Society and Media

Contents

- **Service definition**
- **Mechanisms for parallelism and workflow management**
  - MPI Jobs
  - Collections
  - Directed Acyclic Graphs (DAGs)
  - Petri Net-based workflows
- **WS Interface**

WMS Definition

- **Service responsible for the distribution and management of computational jobs on the resources made available on the Grid**
- **The goal is to provide a generic abstraction of the underlying system**
  - hide heterogeneity of the infrastructure
  - hide complexity of the infrastructure
  - prevent and recover from errors
  - support applications from largely different domains
    - flexible job description language (JDL) based on Condor ClassAds

MPI

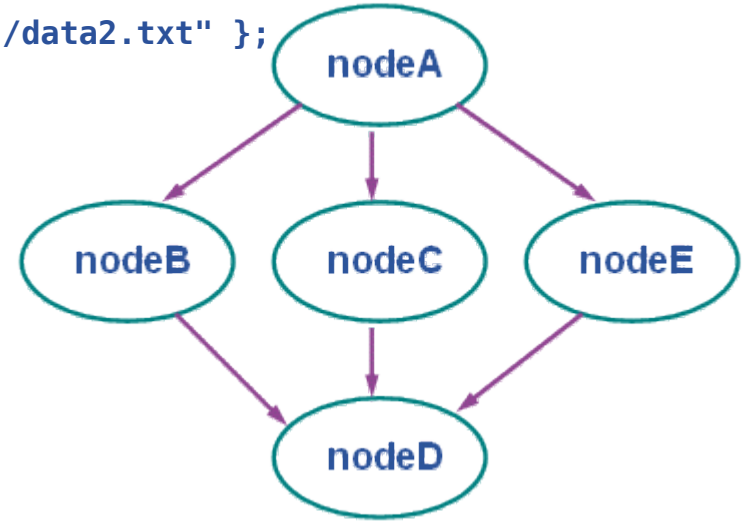
- **Intra-cluster MPI jobs**
  - normal job + specification of the *CPUNumber* attribute
  - the WMS submits the job to the CE asking to reserve the specified number of CPUs
  - the user's job has to determine the batch system type (LSF, PBS, ...) and call *mpirun* itself accordingly

```
[
  Type = "Job";
  CPUNumber = 4;
  Executable = "MPItest.sh";
  Arguments = "cpi 4";
  StdOutput = "test.out";
  StdError = "test.err";
  InputSandbox = {"exe/MPItest.sh", "exe/cpi"};
  OutputSandbox = {"test.err", "test.out", "executable.out"};
  requirements = Member("MPICH", other.GlueHostApplicationSoftwareRunTimeEnvironment)
                 && other.GlueCEInfoTotalCPUs >= 4;
]
```

DAGs

- **Group of jobs with dependencies between them**

```
[
  Type = "dag";
  InputSandbox = {"/home/data/data1.txt", "/home/data/data2.txt" };
  Nodes = [
    nodeA = [
      Description = [
        Executable = "/home/peppe/first.exe";
        InputSandbox = {root.InputSandbox[0]};
        OutputSandbox = {"/home/data/data3.txt"};
        ...
      ];
    ];
    nodeB = [
      Description = [
        Executable = "/home/peppe/second.exe";
        InputSandbox = {root.InputSandbox[1], root.nodes.nodeA.OutputSandbox[0]};
        ...
      ]
    ];
    nodeC = [ ... ];
    nodeD = [ ... ];
    nodeE = [ ... ];
  ];
  Dependencies = {{nodeA, {nodeB, nodeC, nodeE}}, {{nodeB, nodeC, nodeE}, nodeD}}
]
```



DAGs /2

- **Jobs can go to different Computing Elements (CEs) or be collocated on the same CE**
- **Job scheduling is done at run-time**
  - when a node's dependencies have completed successfully
- **The flow control relies on Condor DAGMan**

Collection

- **Group of jobs without dependencies between them**
  - possibly unrelated
  - degenerate DAG

```
[
  Type = "collection";
  Nodes = {
    [
      JobType = "normal";
      Executable = "job1.exe";
      ...
    ],
    [
      JobType = "normal";
      Executable = "job2.exe";
      ...
    ],
    ...
    [
      JobType = "normal";
      Executable = "jobn.exe";
      ...
    ]
  }
]
```

Collection /2

- **Allows *bulk submission***
  - many jobs submitted with one operation
  - reduces submission time
- **Originally implemented like a DAG**
  - source of instability, due to a potentially large number of nodes being scheduled at the same time
- **Now dedicated implementation**
  - allows *bulk matchmaking*: one match for many *equivalent* jobs
    - two jobs are equivalent if the values of their *SignificantAttributes* are literally the same

```
SignificantAttributes = {“requirements”, “rank”};
Requirements = Member("MPICH",other.GlueHostApplicationSoftwareRunTimeEnvironment);
Rank = -other.GlueCEStateEstimatedResponseTime;
```

Parametric

- **A group of almost equal jobs without dependencies**
  - the difference is in the value of some attributes

```
[
  Type = "job";
  JobType = "parametric";
  Executable = "sim.exe";
  StdInput = "input_PARAM_.txt";
  StdOutput = "output_PARAM_.txt";
  Parameters = 10;
  ParameterStart = 1;
  ParameterStep = 1;
  InputSandbox = {"file:///home/user/sim.exe", "file:///home/user/data/input_PARAM_.txt";
  OutputSandbox = {"output_PARAM_.txt"};
]

[
  Type = "job";
  JobType = "parametric";
  ...
  Parameters = {DC1, DC14, DC14bis, HRfixed};
  InputSandbox = {"gsiftp://neo.datamt.it:3344/home/cms/_PARAM_"};
]
```

- **Transformed in a collection**

Workflow Management

- **The goal is to provide a fully generic workflow management engine based on the (High-Level) Petri Net formalism**
  - turing-complete
  - can model any workflow (DAGs, loops, branches, ...)
- **Aim at interoperability between different engines/languages**
  - reference language is GWorkflowDL (with variations)

Workflow Management /2

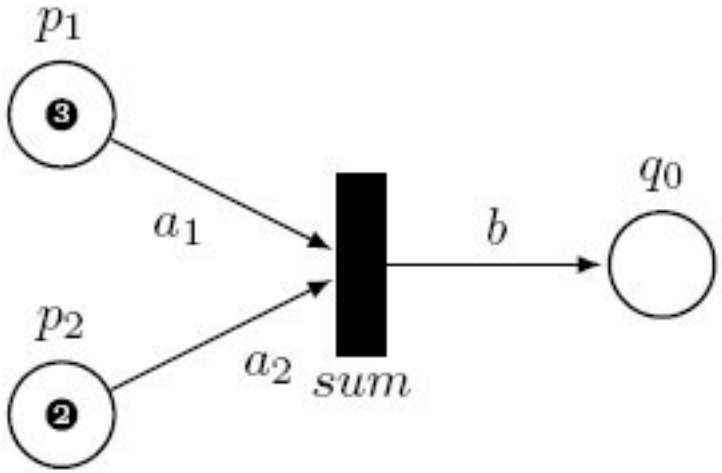
- **Developed within the CoreGrid project**
- **Independent of the underlying Grid infrastructure**
- **Work in progress but experimental prototype available**
  - implemented in C++ and python
  - tested on the EGEE infrastructure
  - translator from JDL available, e.g. for a DAG
- **More information available at**

https://twiki.cnaf.infn.it/cgi-bin/twiki/view/EgeeJra1It/WorkflowManagementSystem

WfMS: basic idea

1. Describe an operation with an abstract PN in GWorkflowDL

```
<workflow>
   <place ID="p1">
     <token><data><t1 xsd:type="xs:int">3</t1></data></token>
   </place>
   <place ID="p2">
     <token><data><t2 xsd:type="xs:int">2</t2></data></token>
   </place>
   <place ID="q0" />
   <transition ID="sum">
     <inputPlace placeID="p1" edgeExpression="a1"/>
     <inputPlace placeID="p2" edgeExpression="a2"/>
     <outputPlace placeID="q0" edgeExpression="b"/>
     <operation /> <!-- generic operation -->
   </transition>
</workflow>
```



WfMS: basic idea /2

2. Map the operation to a concrete one

- web service invocation

```
<operation>
    <wsOperation wsdl="http://localhost/plus?wsdl" operationName="plus">
        <in>n1</in>
        <in>n2</in>
        <out>q1</out>
    </wsOperation>
</operation>
```

- local operation (via python)

```
<operation>
    <pyOperation operation="b = a1 + a2" />
</operation>
```

- sub-workflow

```
<operation>
    <swOperation name="Sum" />
</operation>
```

WMS Interface

- **The current WMS interface is a Web Service**
  - the legacy *Network Server* interface is not supported anymore
- **Changes**
  - decoupling job registration from job start
  - improved performance
  - file perusal
- **Compliant with WS-I Basic Profile**
- **The complementary Logging & Bookkeeping Service also provides a WS interface**
- **For example the WfMS uses the WS interfaces of both WMS and LB**

Conclusions

- **The WMS aims at providing a reliable access point to the Grid for users running computational jobs, hiding the heterogeneity and complexity of the infrastructure**
- **The WMS supports several mechanisms for parallelism and workflow management**
  - intra-cluster MPI jobs, DAGs, collections, parametric jobs
- **A generic workflow management system is under development**
- **The WMS and the LB both provide a WS interface for an easier integration and interoperability with other Grid services**