An Experimental Implementation of OGSA *Advanced* Execution Management Services in NAREGI

March 2005

NAREGI Project, Japan
Objective

- Evaluation of *advanced* features in OGSA-EMS architecture, for complex grid applications such as an “across-sites” job using GridMPI
An example of EM Services interaction in OGSA v1.0

From Figure 5 “Interaction of EMS services to execute a legacy BLASH job” in OGSA Version 1.0:
NAREGI OGSA-EM Services Structure

WP3: WFT

WP1: UNICORE NJS (as a WF engine)
- SuperScheduler
  - Execution Planning Services
  - Candidate Set Generator
- Job Manager
  - Discover & Select
  - Submit

WP3: PSE
- Application Contents Service (ACS)
- Deployment

Distributed Information Services
- Register / Query
- Register
- Update
- Query

GridVM
- Resource Usage Services
- Information Services

UNICORE NJS (as a security framework)
- Reserve
- Deploy
- Deploy

OGSA-BES
- Service Container
NAREGI Super-Scheduler consists of grid services based on GGF OGSA-EMS:

→ **EPS (Execution Planning Services)**
   An EPS is a service that builds mappings called “schedules” between jobs and resources. An EPS will typically attempt to optimize some objective function such as execution time, cost, reliability, etc.

→ **CSG (Candidate Set Generator)**
   A CSG determines the set of resources on which a task can execute. A CSG generates a set of containers (really their Resource Handles) in which it is possible to run a job.

→ **RS (Reservation Service)**
   A RS presents a common interface to all varieties of reservable resources on the grid.
→ NAREGI EPS createAgreement() accepts *an abstract JSDL document* which describes the job requirements, then returns the *Agreement* for the abstract JSDL document.

→ NAREGI EPS calls a CSG to get a set of resources, produces *the concrete JSDL documents* for the resources, and calls a RS to co-schedule the resources.

→ NAREGI EPS supports the following portTypes:

<table>
<thead>
<tr>
<th>portType</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsag:AgreementFactory</td>
<td>wsag:createAgreement</td>
</tr>
<tr>
<td></td>
<td>wsrp:getResourceProperty</td>
</tr>
<tr>
<td>wsag:Agreement</td>
<td>wsag:terminate</td>
</tr>
<tr>
<td></td>
<td>wsrp:getResourceProperty</td>
</tr>
<tr>
<td></td>
<td>wsrl:destroy / wsrl:setTerminationTime</td>
</tr>
<tr>
<td></td>
<td>wsnt:subscribe</td>
</tr>
</tbody>
</table>
NAREGI CSG accepts **an incomplete JSDL document**, produces and issues **the query expressions** to information services from the JSDL, and returns a set of resources (their location information such as EPR).

NAREGI CSG is implemented as a Web Service rather than a Grid Service.

NAREGI CSG supports the following **portType**:

<table>
<thead>
<tr>
<th>portType</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>naregi:CSG</td>
<td>naregi:generateCandidateSet</td>
</tr>
</tbody>
</table>

Note that NAREGI CSG directly takes the job submission description (JSDL) as an input parameter due to lack of job resource registry, while OGSA-EMS compliant CSG will access the specified job resource and get the job submission description (JSDL). Also, NAREGI CSG returns the EPR (End Point Reference) of Reservation Service on each candidate container.


→ NAREGI RS aggregates the operations to resource-level reservation services.
→ NAREGI RS provides synchronous reservations to co-schedule a parallel job across sites.
→ NAREGI RS supports the following portTypes:

<table>
<thead>
<tr>
<th>portType</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsag:AgreementFactory</td>
<td>wsag:createAgreement</td>
</tr>
<tr>
<td></td>
<td>wsrp:getResourceProperty</td>
</tr>
<tr>
<td>wsag:Agreement</td>
<td>wsag:terminate</td>
</tr>
<tr>
<td></td>
<td>wsrp:getResourceProperty</td>
</tr>
<tr>
<td></td>
<td>wsrp:getResourceProperty</td>
</tr>
<tr>
<td></td>
<td>wsrl:destroy / wsrl:setTerminationTime</td>
</tr>
<tr>
<td></td>
<td>wsnt:subscribe</td>
</tr>
</tbody>
</table>
NAREGI EM Services Interaction

1. Abstract JSDL (256)
2. Creation of Agreement Instance
   - Local RS 1: EPR (192)
   - Local RS 2: EPR (64)
3. Candidate Set Generator
   - Candidates: Local RS 1 EPR (192), Local RS 2 EPR (64)
4. Concrete JSDL (192)
5. (1) 14:00-3:00
   - (2) Concrete JSDL (192) + Local RS 2 (EPR)
6. 15:00-18:00
   - Concrete JSDL (192)
7. Cluster (Site) 1
   - Local RS 1
8. 15:00-18:00
   - Concrete JSDL (192)
9. Cluster (Site) 2
   - Local RS 2

Abstract JSDL
Concrete JSDL
Reservation Service
Execution Planning Services
Candidate Set Generator
Local RS #: Local Reservation Service #
1. Our services need to dynamically produce the workflow according to the brokering results.

From our experience

abstract job

 MPI job (256)

concrete job (workflow)

MPI job (64)

MPI job (192)

MPI job (256)
2. Our services needs to reflect the brokering results into the JSDL document.
3. Our services needs to reflect the execution result of other workflow task into the JSDL document.

```xml
<jsdl:Environment>
  <Variable name="IMPI_SERVER">
    X.X.X.X
  </Variable>
  <Variable name="IMPI_PORT">
    56043
  </Variable>
</jsdl:Environment>
```

listen addr/port information : “X.X.X.X:56043”

abstract job

→

concrete job (workflow)

MPI job (256)

GetFile

Does File Exist?

StdOutFile

nothing

mpi-server 2

StdOutFile

2

MPI job (64)

MPI job (192)
4. Our services need to renegotiate the Agreement in order to reflect the runtime information.

From our experience

abstract job

MPI job (256)

concrete job (workflow)

 MPI job (64)
 MPI job (192)

Does File Exist?

StdOutFile

GetFile

mpi-server 2

> StdOutFile

listen addr/port information: “X.X.X.X:56043”
There are some large gaps between basic EM Services and advanced EM Services:

1. Advanced EM Services may handle the workflow to refine the abstract job to the concrete jobs.
2. Advanced EM Services may manage the brokering flow and the execution flow in the single workflow.
3. Advanced EM Services may process the runtime information of applications or their tools.
Thank you !