Parallel jobs in UNICORE 6

Dr. Bernd Schuller
Jülich Supercomputing Centre
Outline

• UNICORE 6 overview and architecture

• Parallel jobs in UNICORE 6
  – Batch job generation
  – Job description
  – Execution environments
  – User side
UNICORE 6

Overview

• Integrated, complete Grid middleware stack including graphical & commandline clients
• Focus on ease of use (both end users and admins)
• Lightweight: only Java + Perl
• Supports many resource management systems and operating systems (Linux/Unix, Mac OS X, Windows)
• Strong support for applications and workflows
UNICORE 6
Architecture

- Portal e.g. GridSphero
- UCC command-line client
- URC Eclipse-based Rich client
- HILA Programming API

scientific clients and applications
UNICORE 6
Architecture

Portal
- e.g. GridSphere

UCC
- command-line client

URC
- Eclipse-based
  Rich client

HiLA
- Programming
  API

Gateway – Site 1

Gateway

Gateway – Site 2

scientific clients
and applications

authentication
UNICORE 6 Architecture

Gateway - Site 1
- UNICORE Atomic Services
- OGSA
- XNJS - Site 1
- UNICORE WS-RF hosting environment
- XACML entity
- XUUDB
- Target System Interface - Site 1
  - Local RMS (e.g. Torque, LL, LSF, etc.)
- External Storage
  - USpace
  - Storage

Gateway
- Service Registry
- Workflow Engine
- Service Orchestrator
- XACML entity
- XUUDB
- IVOS VO Service

Gateway - Site 2
- UNICORE Atomic Services
- OGSA
- XNJS - Site 2
- UNICORE WS-RF hosting environment
- XACML entity
- XUUDB
- Target System Interface - Site 2
  - Local RMS (e.g. Torque, LL, LSF, etc.)
- External Storage
  - USpace
  - Storage

Scientific clients and applications

Authentication
Emerging standard interfaces
Grid services hosting
Job incarnation & authorisation
Parallel scientific jobs of multiple end-users on target systems
Parallel jobs in UNICORE 6: A bit of history...

- UNICORE was initiated in 1997 as a means for simplifying and unifying access to German supercomputer centers (in Jülich, Stuttgart, Munich, ...) 
- Back then, almost every job was a parallel job (excluding e.g. compile jobs), so it was necessary to specially mark serial jobs :-)
- Nowadays, systems and jobs are much more heterogeneous, and UNICORE's parallel job support has changed to reflect that
UNICORE 6
Parallel jobs

• Batch job generation process
• Job description
• Execution environments
• Clients
UNICORE 6

Batch job generation process

1. Submit job description

2. The XNJS generates a TSI script from the job description and local configuration stored in IDB file

3. Send TSI script

4. Submit TSI script ("qsub")
UNICORE 6

Job description

• JSDL 1.0 (OGF standard)
  • Application name / version (mapped to executable by UNICORE) or executable
  • Arguments, environment variables, in/out/err redirects
  • Data staging specification
  • Resources requested (number of nodes, number of CPUs, etc)

• UNICORE-specific extension
  – Execution environment

• “Standardized” JSDL extension: SPMD
  – Covers parallel applications
  – Mapped to execution environments
Execution environments: motivation

- Many different parallel environments
- (MPI, OpenMP, Hybrid, ...) and implementations exist
- Different execution modes (debug, testing, production, ...)
- Additional configuration parameters
- Needed: abstraction, nice user interface
UNICORE 6

Execution environments

• Administrator
  – Knows available tools and how to set them up
  – Configures available options

• User
  – Selects execution environment
  – Chooses parameters and options
  – Customises via pre/post commands
Simple example: IDB entry

```xml
<ee:ExecutionEnvironment xmlns:ee="...">
  <ee:Name>OpenMPI</ee:Name>
  <ee:Version>1.0</ee:Version>
  <ee:ExecutableName>/vsgc/software/openmpi/bin/mpiexec</ee:ExecutableName>

  <ee:Argument>
    <ee:Name>Processes</ee:Name>
    <ee:IncarnatedValue>-np</ee:IncarnatedValue>
    <ee:ArgumentMetadata>
      <ee:Description>The number of processes</ee:Description>
      <ee:Type>double</ee:Type>
      <ee:ValidValue>[1,8192]</ee:ValidValue>
    </ee:ArgumentMetadata>
  </ee:Argument>

  <ee:Option>
    <ee:Name>VERBOSE</ee:Name>
    <ee:IncarnatedValue>-v</ee:IncarnatedValue>
  </ee:Option>
</ee:ExecutionEnvironment>
```
Example: commandline client job file

```json
{
    Executable: "./hello.mpi",

    Imports: [
        {From: "/myfiles/hello.mpi", To: "hello.mpi" },
    ],

    Resources:{ CPUsPerNode: 2, Nodes: 2, },

    Execution environment: {
        Name: OpenMPI,
        Arguments: { Processes: 12, },
    },
}
```
**UNICORE 6**

**Example: rich client**

---

**Job Properties:**

<table>
<thead>
<tr>
<th>Use</th>
<th>Property</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total number of CPUs</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>Number of nodes</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>CPUs per node</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RAM per node</td>
<td>256</td>
<td>MBytes</td>
</tr>
<tr>
<td></td>
<td>Wall time</td>
<td>60</td>
<td>minute</td>
</tr>
<tr>
<td></td>
<td>OS</td>
<td>linux</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU Architecture</td>
<td>x86</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Remote login</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Notification email</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Execution_Environments</td>
<td>OpenMPI</td>
<td></td>
</tr>
</tbody>
</table>

**Execution Environment Settings**

- **Execution Environment**: OpenMPI
- **Run an openmpi application**
  - **OpenMPI**
    - **Processes**: 12.0
    - **ExportEnvironmentVariable**: PATH
    - **VERBOSE**: true
    - **TEST**: true
    - **User Precommand**: 
    - **User Postcommand**: 

---

[Image of UNICORE 6 interface with job properties and execution environment settings]
Example: generated TSI script (sent to TSI and executed by the batch system)

```bash
#!/bin/sh
#TSI_SUBMIT
#TSI_TIME 3600
#TSI_MEMORY 256
#TSI_NODES 2
#TSI_PROCESSORS_PER_NODE 2
#TSI>Total_PROCESSORS 4
#TSI_HOST_NAME none
#TSI_QUEUE batch
#
/vsgc/software/openmpi/bin/mpiexec -np 12 ./hello.mpi
```
UNICORE 6

Example: generated TSI script

```bash
#!/bin/sh
#TSI_SUBMIT
#TSI_TIME 3600
#TSI_MEMORY 256
#TSI_NODES 2
#TSI_PROCESSORS_PER_NODE 2
#TSI_TOTAL_PROCESSORS 4
#TSI_HOST_NAME none
#TSI_QUEUE batch
#

/vsgc/software/openmpi/bin/mpiexec -np 12 ./hello.mpi
```
Summary

• UNICORE 6 execution environments provide user-friendly way to run parallel jobs without the need to know site specific details

• Behind the scenes
  – Administrator configures site-specific details and UNICORE publishes them to Grid clients
  – Batch job generation is customised through execution environments
EMI is partially funded by the European Commission under Grant Agreement RI-261611

Thank you!