Providing a Web Portal for Development and Utilization of Distributed Virtual Test Beds on the basis of UNICORE grid infrastructure

Gleb Radchenko, Elena Hudyakova, Evgeniy Zakharov
South Ural State University (SUSU)
Supercomputer Simulation Laboratory (SSL)

18 June, 2013, UNICORE Summit 2013, Leipzig
Distributed Problem-Oriented Environments
Goal:

- The principal objective is to develop a technology allowing to take into account the specifics of the problem-oriented subject areas while providing the recourses of distributed computing environments.
- This technology aimed to create “intelligent” middleware providing users with easy, transparent and secure access to distributed computing resources and allowing them to solve specific classes of applied problems.
A Problem-Oriented Environment

Define Parameters of Simulation

Get Results of Simulation

Run the Simulation

Development

User

DiVTB

Application programmer

Grid
Distributed Virtual Test Bed

Features:

- Providing a transparent web-access to the resources of a distributed computing environment
- User do not need to know about the distributed nature of the computing environment
- User do not need desktop application to perform simulation
Distributed Virtual Test Bed (DiVTB) includes:

- an **interface** for a CAE-problem statement;
- a **driver** (a set of software tools enabling the use of grid resources for virtual experiment);
- a set of **grid services** (a set of supercomputers in a distributed computing environment, with the installed software components)
DiVTB System
DiVTB Developer
Distributed Virtual Test Bed Development

- Workflow description
- Source files

Workflow
description
Source files

Upload test bed data:
- Source files
- Project.xml
- Parameters.xml

Parameters.xml

DiVTB Developer

DiVTB Server

DiVTB Portal

UNICORE/X

DiVTB Info

Services catalogue

Service run parameters

Application programmer

- Workflow
- Parameters
- Parameterization of source files

2

3

4

5
DiVTB Developer: Features

- Description of the DiVTB workflow in UML-like notation
- Parameterization of DiVTB by making templates of source project files
- Export and import of DiVTB project as XML to DiVTB Server
DiVTB Developer: Used Technologies

- Python/Django
- Javascript/JQuery/Backbone
- Bootstrap
- Raphael.js: workflow designer
- JMTE: Template notation
DiVTB Developer: Creating Project
DiVTB Developer: Adding Nodes

- Conditional Execution
  - Make decision
  - Merge decision

- Parallel Execution
  - Make fork
  - Join fork

- Calculations
  - Add activity

NodeName (ANSYS)
DiVTB Developer: Creating Workflow
DiVTB Developer: Nodes Control

- **Select node:** double click

- **Delete node:** select node and press Delete key

- **Create edge between two nodes:** select source node and double click on target node
DiVTB Developer: Activity Node Settings

Activity Settings

Name: Solver1

The name of activity.

Service: LS-Dyna (1.0)

Description:

Services from DiVTB Info
DiVTB Developer: Parameters

System Parameters

User

Problem Parameters

Activity Settings

Main  System Parameters  Problem Parameters  Files  Input/Output

process number  12  Constant value

(proc_num)

Activity Settings

Main  System Parameters  Problem Parameters  Files  Input/Output

tube_length (Length of main tube)  mm  Constant value

Add problem parameter

Save  Delete node

Save  Delete node
DiVTB Developer: Input/Output Data

- **Input data**
  Files with specified names will be uploaded to computing node during execution of Test Bed

- **Output data**
  Files with specified names will be downloaded from compute node during execution of Test Bed
DiVTB Developer: Source Files

- **Service** need **source files** to do calculations
- Source files automatically become input data

![Activity Settings]

For edit and parameterization of source file
DiVTB Developer: Source Files Templates

Source File Template

Problem Parameters

System Parameters

Compilation

Specific Source File

DiVTB Server

values

SERVICE

Engineer
DiVTB Portal
DiVTB Portal: Overview

- No need to install any applications
- Requires web-browser only
- Problem-oriented user interface
- Works on different devices (PC, Tablet, Smartphone, etc.)
- Provides management functions for:
  - Users (accounts, roles, resources)
  - Testbeds (import, distributing between users)
  - Jobs (submitting, monitoring, getting results)
DiVTB Portal: Architecture
DiVTB Portal: Testbeds import

1. Import testbeds from DiVTB Server

DiVTB Portal

1.1 Get testbed list

1.2 Testbed list

1.3 Save testbeds

Portal manager

DiVTB Server

Testbed manager

Job manager
DiVTB Portal: Create and submit a job

1. Send input files and specify parameters for job submission

2. Create job from received data
   - 2.1 Create job from received data
   - 2.2 Save job in local storage

3. Run job
   - 3.1 Upload job data
   - 3.2 Run job

DiVTB Portal

Testbed manager

Job manager

DiVTB Server
DiVTB Portal: Submit job

1. Engineer
2. Submit job
3. DiVTB Portal
   - TestBed manager
4. Upload job data
5. Job manager
6. Run job execution
7. DiVTB Server
DiVTB Portal: Test beds

<table>
<thead>
<tr>
<th>Testbed name</th>
<th>Version</th>
<th>Author</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS-Dyna Bullet</td>
<td>2.0</td>
<td>Eugene Zakharov</td>
<td></td>
</tr>
<tr>
<td>Magnetization of ultrathin film</td>
<td>1.0</td>
<td>Yury Kirienko</td>
<td></td>
</tr>
<tr>
<td>The construction of the diagnostic scales</td>
<td>1.0</td>
<td>Poluyanov A.N.</td>
<td></td>
</tr>
</tbody>
</table>

- Provision of test beds to engineers
- Generation of virtual experiment from the test bed
### Testbed: LS-Dyna Bullet

**Job identification parameters:**

- **Job name for identification in portal:**
  - <job name "first_category" data="first_category_title"/>
  - <text data="vx_parameter_T" />
  - Default value: **LS-Dyna Bullet**

**Bullet parameters:**

- **Velocity X:**
  - <parameter name="vx" type="Float" visible="true">28.7 m/s</parameter>
  - Default value: **28.7 m/s**

- **Velocity Y:**
  - <parameter name="vy" type="Float" visible="true">0 m/s</parameter>

- **Velocity Z:**
  - <parameter name="vz" type="Float" visible="true">0 m/s</parameter>

**Submit**  **Back**
# DiVTB Portal: Interface & parameters

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>Interface item</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>LS-Dyna Bullet&lt;br&gt;Default value: LS-Dyna Bullet</td>
</tr>
<tr>
<td>String with constraints (enum)</td>
<td>LS-Dyna Test&lt;br&gt;Default value: LS-Dyna Test</td>
</tr>
<tr>
<td>Integer, Float, Double, ...</td>
<td>28.7 m/s&lt;br&gt;Default value: 28.7 m/s</td>
</tr>
<tr>
<td>Integer, Float and Double with constraints (enum)</td>
<td>39.2&lt;br&gt;Default value: 28.7 m/s</td>
</tr>
<tr>
<td>Input file (select)</td>
<td>Select file</td>
</tr>
<tr>
<td>Input file (uploading)</td>
<td>Uploading source_parameter.dat</td>
</tr>
<tr>
<td>Input file (uploaded)</td>
<td>source_parameter.dat</td>
</tr>
</tbody>
</table>
DiVTB Portal: Job submission

1: Async upload of input file
1.2: UploadState
2: Submit
2.3: return

1.1: input file processing
2.1: Validation of the parameters
2.2: createInstance
2.2.1: new instance()
2.2.2: instanceId
2.3: Pack input files into instanceId.zip
2.4: uploadInputFiles(instanceId, instanceId.zip)
2.4.1: unzip(instanceId.zip)
2.4.2: return
2.5: submitJob(instanceId, parameters)
2.5.1: execute(instanceId)
2.5.2: return
DiVTB Portal: Jobs

<table>
<thead>
<tr>
<th>Job Name</th>
<th>Start Time</th>
<th>Finish Time</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS-Dyna Bullet (Vacuum)</td>
<td>2013-02-23</td>
<td>???:???:??</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17:10:35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS-Dyna Bullet (Water)</td>
<td>2013-02-23</td>
<td>???:???:??</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17:10:12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS-Dyna Bullet 2</td>
<td>2013-02-23</td>
<td>2013-02-23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17:03:08</td>
<td>17:08:43</td>
<td></td>
</tr>
<tr>
<td>LS-Dyna Bullet 1</td>
<td>2013-02-23</td>
<td>2013-02-23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16:53:18</td>
<td>16:53:21</td>
<td></td>
</tr>
</tbody>
</table>

- Update job state in the grid environment
- Get execution results
- Stop execution and/or remove finished jobs
DiVTB Portal: Fetching results

1. Engineer requests job results.
2. DiVTB Portal retrieves results.
3. Results are stored in a database.
5. Engineer receives job execution results in the caebeanId.zip archive.

Steps:
- Request for job results
- Get job results
- Sending parts of caebeanId.zip (contains job execution results)
Deployment and testing
DiVTB Portal: Role in job execution

1. Engineer specifies values:
   - Parameter values
   - Input files

2. DiVTB Portal uploads job data:
   - Input files
   - ParameterValues.xml

3. Request for TSI:
   - Workflow
   - Resource parameters

4. DiVTB Broker

5. DiVTB Server compiles job files from file templates

6. Execution results

7. UNICORE/X runs service:
   - Job files
   - System parameters

UNICORE/X
DiVTB System Deployment
Testing speed (1)

Uploading source files into DiVTB system

Time to upload, min.

File size, GB

1

2

5

SSH
DiVTB Portal
UNICORE Rich Client
Testing speed (2)

Download results of virtual experiment

Time to download, min.

File size, GB

- SSH
- DiVTB Portal
- UNICORE Rich Client
What’s next?

- Validation and visualization of workflows execution in DiVTB Portal and DiVTB Developer
- Development of interactive remote visualization system for DiVTB Portal
- Integration of authorization and authentication with UNICORE services
Questions?

http://supercomputer.susu.ac.ru/en/