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

Gleb Radchenko¹, Elena Hudyakova¹, and Evgeniy Zakharov¹

Supercomputer Simulation Laboratory,
South Ural State University, Chelyabinsk, Russia
E-mail: radchenko@acm.org

The Distributed Virtual Test Bed (DiVTB) technology provides a solution for the integration of problem-oriented systems in distributed computing environments. DiVTB provides a problem-oriented user interface to distributed computing resources within the grid, online launch of virtual experiments, automated search, monitoring and allocation of computing resources for carrying out the virtual experiments. The most important task of the DiVTB system is to provide a user-friendly interface to create and use DiVTB. This paper describes the architecture of Web systems that provide a direct interaction of the DiVTB platform with end users: DiVTB Developer and DiVTB Portal. DiVTB Developer is a Web-based integrated development environment of DiVTB. DiVTB Portal provides a problem-oriented user interface for management and execution of virtual experiments in a distributed computing environment.

1 Introduction

Web-portals increasingly become a major way to provide interfaces to distributed computing systems. Using the Web-based applications one can significantly simplify access to a distributed computing environment for end users. First, the user is not required to install and configure desktop applications to access remote systems. Second, the use of Web technologies such as HTML5 can significantly extend the range of devices from which you can access the grid systems.

Let us consider most widespread Web-portal systems. The Vine Toolkit¹ was developed to provide a Web platform for different scientific portals. The main purpose of the Vine Toolkit is to provide a convenient way to gain access to remote supercomputer resources and grid systems. P-GRADE Portal² is a Web portal focused on management of workflow jobs in grid environments based on Globus Toolkit platform³. It includes the tools necessary to create, execute and monitor workflows. For requirements of EGEE (Enabling Grids for E-Science in Europe)⁴ and the WLCG (The Worldwide LHC Computing Grid)⁵ projects the CIC Portal⁶ (now known as Central Operations Portal) was developed. This portal includes features such as computing environment load monitoring, users management and virtual organizations support, allocation of computing resources among tasks, etc.

After analyzing the existing solutions as well as customer feedback, we identified the following requirements for Web clients for the Distributed Virtual Test Bed System⁷:

- a Web-client should be developed using pure Web technologies (such as HTML5), without the need for additional plug-ins such as Java, Flash, Silverlight etc.;
• an interface for development (IDE) and for using of distributed applications should be separated, providing end users with transparent problem-oriented interface to the grid environment;

• a convenient visual mechanism for workflow descriptions should be provided;

• a convenient mechanism for parameterization of the source files will allow the most seamless integration of classical batch applications in the grid environment;

• automatic generation of the user interface based on the description of the parameters of distributed application should be provided.

The paper is organized as follows: the next section presents an overview of the Distributed Virtual Test Bed (DiVTB) project, including an architecture of DiVTB System software solution. Section 3 describes architecture and features of the DiVTB Developer Web-IDE. In Section 4 the DiVTB Portal Web-application is presented. And finally, in Section 5 possible future extensions and enhancements are discussed.

2 DiVTB System

The principal objective of the Distributed Virtual Test Bed project - is to develop a technology allowing to take into account the specifics of the problem-oriented subject areas while providing the resources of distributed computing environments. This technology aims 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.

The Distributed Virtual Test Bed (DiVTB) technology provides a solution for the integration of problem-oriented systems in distributed computing environments. DiVTB provides a problem-oriented user interface to distributed computing resources within the grid, online launch of virtual experiments, automated search, monitoring and allocation of computing resources for carrying out the virtual experiments.

To implement the DiVTB Technology we created a DiVTB System - a UNICORE-based software solution for DiVTB development and implementation. We selected the UNICORE 6 grid computing middleware as a platform for implementation of grid services. The GridBeans approach included in UNICORE 6 supports the transparent integration of legacy standalone applications as grid services in the grid environment.

Within the DiVTB System (Fig. 1) we distinguish two levels of the components: "driver" level and "client" level. The main representatives of the driver level are DiVTB Server and DiVTB Broker components. They provide execution of virtual experiments in distributed computing environment and usage of UNICORE grid resources.

The "client" layer provides an interaction of developers and end-users with the DiVTB System. We decided to separate the process of development and use of a virtual test bed. Accordingly, we developed two Web applications that provide development and utilization of Distributed Virtual Test Beds:

• DiVTB Developer - a Web IDE that provides the development process of distributed virtual test beds.
DiVTB Portal - a Web application that provides a problem-oriented user interface for management and execution of virtual experiments in a distributed computing environment.

3 DiVTB Developer

The DiVTB Developer Web IDE provides the development of DiVTB, including visual design of workflows, parametrization of the source files, definition and description of parameters of the virtual experiment.

The application programmer designs the virtual test bed by describing the workflow. A workflow consists of actions, which define stages of virtual experiment. Each action is implemented by computing services, located in the UNICORE grid environment. DiVTB Developer receives information about the available computing services via DiVTB Info component, which periodically polls the available grid nodes and receives information about the interfaces of UNICORE Applications installed on them.

The developer can import and export the DiVTB Project as an archive which consists of XML-based project files and a set of additional source files. In addition, the developer can export the DiVTB Project to DiVTB Server component, which provides storage and execution of DiVTB.

3.1 DiVTB Developer Client

The DiVTB Developer IDE is implemented as a client-server Web-based application (Fig. 2). The server part provides storage and handling of DiVTB projects. The client part provides a visual interface for workflow creation and customization.

The project management component processes and stores DiVTB projects in the project database. It interacts also with the DiVTB Info service to get the information about services available in the UNICORE Grid, including parameters of each service. The DiVTB Server interaction component allows the developer to upload the DiVTB project to the DiVTB Server, which provides execution of the virtual experiment in the UNICORE grid environment.
3.2 DiVTB Developer Interface

The user interface of the DiVTB Developer IDE can be divided into three logical parts: workflow description module, action description module, source file editing and parameterization module.

The workflow description module provides the DiVTB workflow design (Fig. 3). The developer can build a workflow using several node types (action nodes and control nodes), connected by edges which describe the control flow.

In table 1 you can find available types of workflow nodes, including their graphical representation and semantics.

In the action description module developer can specify the following parameters of the node:

- a name and a description of the semantics of action node;
- a basic service of an action (developer can choose a service from the list of existing services);
- values of system parameters (start-up options for a service);
- a list of problem parameters in terms of the relevant problem domain;
- names or name masks of input source files that are to be uploaded/downloaded from the grid service before/after the calculations execution;
- a list of input files for a service.

The source file editing and parameterization module provides an interface for editing DiVTB input files (Fig. 4). These files describe the simulated problem in a format that is
appropriate to the end computing services. The developer can edit any source file and insert a special template code into the source code using problem or system parameters of the DiVTB as variables. The parametrized source files together with values of DiVTB parameters would be sent to the DiVTB Server during virtual experiment initialization. There, the parametrized templates would be transformed to the source files for the corresponding UNICORE applications.

We use Java Minimal Template Engine (JMTE) as a template engine for DiVTB. JMTE is a template system that allows developer to embed variables in the template text, using a simple syntax that supports basic operations with variables (like loops, conditions etc.). Thus, creating a problem-oriented file once for a specific task, the developer can describe a DiVTB for a whole class of similar problems.

4 DiVTB Portal

4.1 DiVTB Portal Architecture

The main purpose of the DiVTB Portal is to provide a problem-oriented user interface for management and execution of virtual experiments in a distributed computing environment. The engineer interacts with a Web-interface of the DiVTB Portal. It provides a list of test beds available to the user and allows the user to setup and run the virtual experiment. It also responsible for authentication and management of DiVTB System users.

The DiVTB Portal consists of five components (Fig. 5):

- Test Beds Manager
- Web Forms Generator
<table>
<thead>
<tr>
<th>Name</th>
<th>Graphical representation</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial node</td>
<td><img src="initial_node.png" alt="Initial Node" /></td>
<td>A control node that starts the workflow.</td>
</tr>
<tr>
<td>Final node</td>
<td><img src="final_node.png" alt="Final Node" /></td>
<td>A control node that indicates the end of the workflow</td>
</tr>
<tr>
<td>Action node</td>
<td><img src="action_node.png" alt="Action Node" /></td>
<td>The Action node provides a specific action of a virtual experiment on the basis of particular service, available in the UNICORE Grid.</td>
</tr>
<tr>
<td>Decision node</td>
<td><img src="decision_node.png" alt="Decision Node" /></td>
<td>A control node that provides a choice of a route of a control flow depending on the value of a Boolean expression (the WHEN field of a control flow branch).</td>
</tr>
<tr>
<td>Merge node</td>
<td><img src="merge_node.png" alt="Merge Node" /></td>
<td>A control node, which merges two or more alternative control flow branches.</td>
</tr>
<tr>
<td>Fork node</td>
<td><img src="fork_node.png" alt="Fork Node" /></td>
<td>A control node that initiates the execution of multiple concurrent control flows.</td>
</tr>
<tr>
<td>Join node</td>
<td><img src="join_node.png" alt="Join Node" /></td>
<td>A control node that synchronizes several parallel control flows.</td>
</tr>
</tbody>
</table>

Table 1. Workflow node types

- Client for DiVTB Server
- Virutal Experiments Manager
- Local Storage

*Test Beds Manager* provides management of test beds, including import of test beds from the DiVTB Server and distribution of test beds between the users of the DiVTB System.

*Web Forms Generator* provides automatic generation of DiVTB Web-interface on the basis of parameters of DiVTB. Using this interface, a user can specify values of parameters of DiVTB and submit the virtual experiment into the UNICORE grid environment.

*Client for DiVTB Server* provides interaction with DiVTB Server service, including download of available DiVTB, virtual experiments submission and download of simulation results.

*Virtual Experiments Manager* provides functions for virtual experiments execution, including updates of status of workflow execution and retrieval of results.
4.2 Management of DiVTB

All the available test beds are stored in the project storage of the DiVTB Server. Each test bed has a unique identifier (the UID). Based on the UID, DiVTB Portal can determine presence or absence of the certain test bed in its local database. To maintain the database of distributed virtual test beds, DiVTB Portal uses Virtual Experiments Manager which provides import of test beds from the DiVTB Server storage.
During the test beds import process, DiVTB Portal requests from the DiVTB Server a list of the UIDs of all available DiVTB using the getProjectIDs method. Next, DiVTB portal checks presence or absence for each identifier in its own database.

If the DiVTB is absent in the DiVTB Portal’s database, it requests the description of DiVTB interface and parameters of the virtual experiment by calling getProblemCaebean method from the DiVTB Server. After retrieval, the DiVTB can be provided for the users of DiVTB Portal.

In case if the test bed is presented in the DiVTB Portal database but unavailable in DiVTB Server, its status changes to "Not Active". The user of the DiVTB Portal can not create a virtual experiment based on the test bed with such status. However, he can view parameters and download results of previous experiments based on such test bed.

All test beds in the DiVTB Portal are divided into two groups: test beds, imported into DiVTB Portal, and test beds that are available to the user of the portal. Each user can ask from an administrator of a DiVTB System for a subscription to any available test bed.

A user can create a virtual experiment based on any DiVTB on which he signed. A user also can unsubscribe from a DiVTB or view a list of his/her virtual experiments.

4.3 Generation and Execution of Virtual Experiment

Each DiVTB consists of groups of parameters. Each group is a union of semantically related parameters of the experiment. Parameter names should be unique within the group. For each group, the Web Forms Generator creates a container.

For each parameter inside the group, the Web Forms Generator creates an HTML-form, corresponding to the type of the parameter (table 2). Using DiVTB Developer, an application programmer can limit the variety of definitions of any parameter (except the "File" parameter) by listing the possible values of the parameter. In this case Web Forms Generator will provide to the user a drop down list with the possible values of the parameter.

Web Forms Generator of the DiVTB Portal allows users to set parameters of the test beds, by specifying their values in a simple HTML-form. The parameters in this case may have a different type (e.g. String, Integer, Float, File and etc.). If the parameter type is "File", the user needs to select the desired file through the file selection dialog. Selected files are loaded asynchronously into the DiVTB Portal and temporarily stored before the job execution. Figure 6 shows an example of the generated interface for specifying parameter values.

Before submission, the values of all parameters (including files) are validated by the DiVTB Portal. If the value of a parameter does not match the domain of its possible values, the user will be prompted to fix the value of the corresponding parameter. After successful validation, DiVTB Portal calls a CreateInstance method of the DiVTB Server and creates an instance of an experiment. Next, all source files are uploaded from DiVTB Portal to DiVTB Server by the uploadSourceFiles method. Finally, the submitJob method provides a transfer of the parameters of the virtual experiment to the DiVTB Server in a context of a previously created instance. Figure 7 shows the process of setting up and successful completion of the virtual experiment.

There are four possible states of virtual experiments:

- **NOT_STARTED** - the initial state. The corresponding job is not running.
Table 2. Mapping DiVTB parameters and corresponding Web forms

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>Interface item</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>![Image](LS-Dyna Bullet) Default value: LS-Dyna Bullet</td>
</tr>
<tr>
<td>Integer, Float, Double, …</td>
<td>![Image](28.7 m/s) Default value: 28.7 m/s</td>
</tr>
<tr>
<td>Integer, Float and Double with constraints (enum)</td>
<td>![Image](30.2 m/s) Default value: 28.7 m/s</td>
</tr>
<tr>
<td>Input file (select)</td>
<td>![Image](Select file)</td>
</tr>
<tr>
<td>Input file (uploading)</td>
<td>![Image](Uploading source_parameter.dat)</td>
</tr>
<tr>
<td>Input file (uploaded)</td>
<td><img src="source_parameter.dat" alt="Image" /></td>
</tr>
</tbody>
</table>

Testbed: LS-Dyna Bullet

**Job identification parameters:**

- **Job name**: LS-Dyna Bullet  Default value: LS-Dyna Bullet

**Bullet parameters:**

- **Velocity X**: 28.7 m/s  Default value: 28.7 m/s
- **Velocity Y**: 0 m/s  Default value: 0 m/s
- **Velocity Z**: 0 m/s  Default value: 0 m/s

Figure 6. Example of interface for virtual experiment definition for the "LS-Dyna Bullet" virtual test bed

- **RUNNING** - intermediate state. The virtual experiment was started by the `submitJob` method.
When a user requests the results of a virtual experiment being in the intermediate state, the DiVTB Portal uses the `getStatus` method to get currently available intermediate results. When the experiments status change from an initial or intermediate to the final state, DiVTB Portal obtains the final results by means of `getStatus` method and calls the `getExecTime` method to get the duration of the conducted virtual experiment.

When the user deletes a virtual experiment in SUCCESSFULL state, the DiVTB Portal sends a `removeResults` request to the DiVTB Server. This query deletes all the results of the virtual experiment stored on the DiVTB Server. If the user wants to stop and delete a virtual experiment in RUNNING, HELD or FAILED state, the DiVTB Portal calls `removeWorkDir` method, that stops the execution of all active actions of the workflow.
and deletes all intermediate results from the DiVTB Server.

4.4 DiVTB Portal File Transfer Tests

To test DiVTB Portal file transfer speed and compare it with alternative ways of files transfer (such as SSH and UNICORE Rich Client file transfer) we deployed DiVTB System on computing resources of Supercomputing Simulation Laboratory of South-Ural State University. For testing, we used the client PC from the external network. We have prepared three test files, size of 1, 2, and 5 GB respectively. Each file has been downloaded and uploaded from the node, where the DiVTB Portal was located by means of SSH, UNICORE Rich Client and DiVTB Portal systems. To take into account temporal anomalies in the network, each download and upload procedure was carried out for 5 times. Average values of measurements made are shown in Tables 3 and 4. As can be seen from the test results, the rate of data transfer to/from the end user through the DiVTB Portal corresponds to the existing common methods of data transmission.

![Table 3. Comparison of download time and speed (mean values)](image)

<table>
<thead>
<tr>
<th>File Size, MB</th>
<th>SSH</th>
<th>DiVTB Portal</th>
<th>UNICORE Rich Client</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time, s.</td>
<td>Speed, Mbit/s</td>
<td>Time, s.</td>
</tr>
<tr>
<td>1024</td>
<td>459</td>
<td>17.85</td>
<td>456</td>
</tr>
<tr>
<td>2048</td>
<td>918</td>
<td>17.85</td>
<td>949</td>
</tr>
<tr>
<td>5120</td>
<td>2293</td>
<td>17.86</td>
<td>2323</td>
</tr>
</tbody>
</table>

Table 4. Comparison of upload time and speed (mean values)

<table>
<thead>
<tr>
<th>File Size, MB</th>
<th>SSH</th>
<th>DiVTB Portal</th>
<th>UNICORE Rich Client</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time, s.</td>
<td>Speed, Mbit/s</td>
<td>Time, s.</td>
</tr>
<tr>
<td>1024</td>
<td>156</td>
<td>52.51</td>
<td>300</td>
</tr>
<tr>
<td>2048</td>
<td>306</td>
<td>53.54</td>
<td>684</td>
</tr>
<tr>
<td>5120</td>
<td>720</td>
<td>56.89</td>
<td>1512</td>
</tr>
</tbody>
</table>

5 Conclusion and Future Work

We presented an architecture and peculiarities of the DiVTB System’s user-oriented Web-applications: DiVTB Developer and DiVTB Portal. The DiVTB Developer provides a Web-based IDE for the development of distributed virtual test beds, including design of
a workflow of a DiVTB, description of problem-oriented parameters of simulation and parametrization of source files by means of templating engine.

The DiVTB Portal provides a problem-oriented user interface for management and execution of virtual experiments in a distributed computing environment, supporting automatic generation of Web-forms for virtual experiments definition on the basis of DiVTB parameters description and transfer of input and output files of virtual experiment from the DiVTB Server.

As future work, there are several areas in which we would like to continue the development of the DiVTB Web-applications. First, the experience of using the DiVTB Developer and DiVTB Portal systems has identified the need of workflows validation and visualization of workflows execution.

Second, users asked us to provide a possibility to visualize the results of their virtual experiments without the need of downloading them. A possible solution to this is to provide a remote interactive visualization system for the DiVTB Portal.

In addition, we would like to provide an integration of our authorization and authentication services with UNICORE services. This will significantly enhance the security of the DiVTB system and greatly expand its applicability to other UNICORE-based grids.

Acknowledgments

The reported study was partially supported by RFBR, research project No. 11-07-00478-a and by the Ministry of Education and Science of Russia, task No. 8.3786.2011

References