



CROSSGRID INSTALLATION GUIDE

GRID RESOURCE MANAGEMENT

WP 3.2

| | |
|--------------------------|--|
| Document Filename: | CG-InstallGuide |
| Work package: | WP 3.2-Grid resource management |
| Partner(s): | UAB, DATAMAT, IFIC (CSIC) |
| Lead Partner: | UAB |
| Config ID: | CG3.2-v2.3-InstallationGuide |
| Document classification: | public |

Abstract:

This document is intended for Site Administrator of CrossGrid testbed resources and may also be useful for people that need detailed information about the WP3.2 release

Delivery Slip

| | Name | Partner | Date | Signature |
|--|------|---------|------|-----------|
|--|------|---------|------|-----------|

From

Document Log

| Version | Date | Summary of changes | Author |
|-------------|------------|---|---|
| 1-0-DRAFT-A | 16/1/2003 | Draft Version | Álvaro Fernández |
| 1.1 | 20/2/2003 | Added Tests to see the specific behaviour of the modified RB | Álvaro Fernández |
| 2.0 | 18/3/2004 | Refer to software based on LCG-1 | Enol Fernández, Álvaro Fernández |
| 2.1 | 19/6/2004 | -Added Information and Tests about new interactive features (MPICH-P4 & MPICH-G2 support) -Update package and build information - information and tests about JSS interface | Antonio Hervás, Álvaro Fernández, Marco Sottilaro |
| 2.2 | 2/11/2004 | -Added postprocessing information. -Updated format to CrossGrid Installation Guide Template | Enol Fernández, Álvaro Fernández, Antonio Hervás |
| 2.3 | 15/12/2004 | - Added Glide-In information | Enol Fernández |

Contents

| | | |
|----------|---|-----------|
| 1 | COPYRIGHT NOTICE | 4 |
| 2 | ABOUT THE SOFTWARE | 5 |
| 2.1 | SOFTWARE COMPONENTS..... | 5 |
| 2.2 | DEPENDENCIES | 5 |
| 2.2.1 | <i>Prerequisites for Logging and Bookkeeping services</i> | <i>5</i> |
| 2.2.2 | <i>Prerequisites for “RB node”</i> | <i>6</i> |
| 2.2.3 | <i>Prerequisites for the User Interface</i> | <i>7</i> |
| 2.2.4 | <i>Prerequisites for the JSS Interface</i> | <i>8</i> |
| 3 | INSTALLATION IN THE CROSSGRID TESTBED | 9 |
| 3.1 | RPM LISTS FOR LCFG..... | 9 |
| 3.2 | PROFILE MODIFICATIONS FOR LCFG | 10 |
| 3.3 | MANUAL POST INSTALLATION STEPS..... | 12 |
| 4 | MANUAL INSTALLATION | 13 |
| 4.1 | DOWNLOAD..... | 13 |
| 4.2 | INSTALLATION FROM RPM..... | 13 |
| 4.3 | INSTALLATION FROM SOURCE..... | 13 |
| 4.3.1 | <i>External dependencies.....</i> | <i>13</i> |
| 4.3.2 | <i>Setting environment variables</i> | <i>16</i> |
| 4.3.4 | <i>Compiling the code.....</i> | <i>17</i> |
| 4.4 | CONFIGURATION | 18 |
| 4.4.1 | <i>List of configuration files</i> | <i>18</i> |
| 4.4.2 | <i>Editing the configuration files</i> | <i>19</i> |
| 4.4.3 | <i>Startup scripts</i> | <i>19</i> |
| 4.4.4 | <i>Other requirements.....</i> | <i>20</i> |
| 5 | RUNNING AND TESTING | 22 |
| 5.1 | RUNNING THE RESOURCE BROKER | 22 |
| 5.2 | TESTING THE NEW FEATURES | 22 |
| 5.2.1 | <i>SMF support for MPICH-P4 Jobs.....</i> | <i>22</i> |
| 5.2.2 | <i>Multiple CEs selection.....</i> | <i>22</i> |
| 5.2.3 | <i>Interactive MPICH-P4 and MPICH-G2 Jobs support</i> | <i>27</i> |
| 5.2.4 | <i>Monitoring Plug-in Support</i> | <i>32</i> |
| 5.2.5 | <i>Glide-In Mechanism</i> | <i>33</i> |
| 5.3 | LOG FILES | 36 |
| 6 | THE EDG LICENSE AGREEMENT | 37 |
| 7 | BIBLIOGRAPHY | 39 |

1 COPYRIGHT NOTICE

Copyright (c) 2005 by Álvaro Fernández Casani, Enol Fernández del Castillo, Antonio Hervás Vilchez, Elisa Heymann Pignolo, Anna Morajko, Miquel Angel Senar and Marco Sottilaro on behalf of the EU CrossGrid. All rights reserved.

This research is partly funded by the European Commission IST-2001-32243 Project “CrossGrid”.

Use of this product is subject to the terms and licenses stated in the EDG license agreement. Please refer to Chapter 6 for details.

This software uses code from the following products:

EDG Workload Management. Copyright (c) 2002 CERN and INFN on behalf of the EU DataGrid.

2 ABOUT THE SOFTWARE

Current development of CrossGrid middleware is based on the DataGrid WP1 WMS Software, with the corresponding modifications to add the new functionality required by the CrossGrid applications. The edge release currently used is EDG 2.1.15, and CrossGrid specific release is currently version 2.5.

We name the rpm packages provided with the juxtaposed number version, which makes our current release number **2.1.15.2.5** (lcg-2 compliant)

This guide describes the installation procedure of the CrossGrid middleware that is being developed in Task 3.2. Since the process of installation and configuration is very similar to the DataGrid one, this guide is tightly related with the DataGrid WP1 WMS Software Administrator and User Guide [1], and it will continuously refer to that document.

This document merely gives the basic steps for installation and information about the differences and the new configuration, but the DataGrid document is required for a more deeply understanding of some aspects of the installation and configuration.

2.1 SOFTWARE COMPONENTS

A detailed description of the software is presented in section 3 of [1].

2.2 DEPENDENCIES

The compiled software requires of some external packages to run properly in the installed and configured machines. What comes next is a summary of the presented in section 4 of [1]. Please refer to that guide for a more detailed explanation of the software requirements.

2.2.1 Prerequisites for Logging and Bookkeeping services

From the installation point of view LB services can be split in three main components:

LB local logger and LB APIs: responsible for accepting messages from their sources and forwarding them to the LB server.

For the installation of the LB local-logger and LB APIs the only software required is the Globus Toolkit 2.2 (actually only GSI rpms are needed). Globus 2.2 RPMs are available at <http://DataGrid.in2p3.fr/distribution/globus/vdt-1.1.8/globus/RPMS/>

LB Server: responsible for accepting messages from the LB local logger services, saving them on permanent storage and supporting queries about those messages, generated by the consumer API.

For the installation of the LB server the only software required is the Globus Toolkit 2.2 (actually only GSI RPMs are needed). Globus 2.2 RPMs are available at <http://DataGrid.in2p3.fr/distribution/globus/vdt-1.1.8/globus/RPMS/>

Besides the Globus Toolkit, for the LB server to work properly it is also necessary to install MySQL Distribution 4.0.1 or higher.

Packages and documentation about MySQL can be found at: <http://www.mysql.org>.

Anyway the MySQL RPMs for pc-linux-gnu (i686) is available at <http://DataGrid.in2p3.fr/distribution/external/RPMS/>. At least packages MySQL-4.0.x and MySQL-client-4.0.x have to be installed for creating and configuring the LB database.

The *LB local-logger* services must be installed on all the machines hosting processes pushing information into the LB system, i.e. the “RB node” and the gatekeeper machines of the CEs. An exception is the submitting machine (i.e. the machine running the User Interface) on which this component can be installed but is not mandatory.

The *LB server* services need instead to be installed only on a server machine.

The LB APIs should be installed on the UI machine (C and C++ APIs), “RB node” (C and C++ APIs) and on the CE worker nodes (C and sh APIs).

2.2.2 Prerequisites for “RB node”

The Resource Broker is the component of the DataGrid WMS that has been mainly modified in CrossGrid. The requirements are the same as outlined in section 4.2.1 of [1]:

For the installation of services the Globus Toolkit 2.2 rpms available at <http://DataGrid.in2p3.fr/distribution/globus> under the directory *vdt-1.1.8/RPMS* are required to be installed on the target platform.

Please note that the “RB node” should run a gridftp server (actually a “customized” one), while it should not run a globus gatekeeper.

It is important to recall that the Globus *gridmap file* located in */etc/grid-security* on the RB server machine must be filled with the certificate subjects of all the users allowed to use the Resource Broker functionalities. Users being mapped into the *gridmap file* have to belong to a group having the same name of the user itself. At the same time the dedicated user *edguser* has to belong to *all* these groups.

Moreover on the same platform the following products are expected to be installed:

- **LB local-logger services** (see section 2.2.1)

- **Condor-G**

Condor-G release required is CondorG 6.6.0-2.edg4 for INTEL-LINUX. The Condor-G is available in rpm format (to be installed as root) at:

<http://grid-deployment.web.cern.ch/grid-deployment/download/RpmDir/external>

- **ClassAd library**

The ClassAd release required is a customized classads-0.9.4 release, available in rpm format at:

<http://datagrid.in2p3.fr/distribution/external/RPMS>

The ClassAd library documentation can be found at the following URL:

<http://www.cs.wisc.edu/condor/classad>.

- **Boost library**

The Boost C++ libraries release required is 1.29 (or higher). The boost documentation can be found at the following URL:

<http://www.boost.org>

whilst it is available in rpm format (to be installed as root) at:

<http://datagrid.in2p3.fr/distribution/external/RPMS>

– Replica Manager

The Replica Manager RPMs that must be installed are:

- edg-gsoap-base-1.0.3-1.i386.rpm
- edg-replica-location-client-c++-1.2.8-1.i386.rpm
- edg-replica-optimization-client-c++-1.2.9-1.i386.rpm
- edg-replica-metadata-catalog-client-c++-1.2.8-1.i386.rpm
- edg-replica-manager-client-c++-1.0.6-1.i386.rpm

After the RPM installation, it is then needed to configure the configuration files for the various VOs in <install-dir>/etc/edg-replica-manager.

2.2.3 Prerequisites for the User Interface

The CrossGrid specific UI is backwards compatible with the edg one. With one edg-based the user will be able to contact a CrossGrid Rb, but however some functionality won't be accessible.

This section describes the steps needed to install and configure the User Interface, which is the software module of the WMS allowing the user to access main services made available by the components of the scheduling sub-layer.

The UI software is distributed in 4 different packages:

- The python command line interface
- The C++ API
- The Java API
- The Java GUI

All the above listed packages have a dependency on the Globus Toolkit software. The required release is 2.2 from the VDT distribution. It can be downloaded from <http://datagrid.in2p3.fr/distribution/globus/vdt-1.1.8/globus/RPMS/>. The needed rpms are listed here below:

- vdt_globus_essentials-EDGVDT1.1.8-5.i386.rpm
- vdt_globus_sdk-EDGVDT1.1.8-5.i386.rpm
- vdt_compile_globus_core-EDGVDT1.1.8-1.i386.rpm
- globus-initialization-2.2.4-2.noarch.rpm

Moreover the set of security configuration rpm's for all the Certificate Authorities in Testbed2 available at <http://datagrid.in2p3.fr/distribution/DataGrid/security/RPMS/> have to be installed together with the rpm to be used for renewing your certificate for your CA. This is available at <http://datagrid.in2p3.fr/distribution/DataGrid/security/RPMS/local/>.

The MyProxy package should be installed on the UI node in order to allow users to take advantage of the proxy-renewal feature for long running jobs. The corresponding rpm can be found at <http://datagrid.in2p3.fr/distribution/external/RPMS> and is named as follows:

- myproxy-gcc32dbg-client-0.5.3-1.i386.rpm

The Python interpreter, version 2.2.2 must also be installed on the submitting machine. The rpm for this package is available at <http://datagrid.in2p3.fr/distribution/redhat-7.3/updates/RPMS> as:

- python2-2.2.2-11.7.3.i386.rpm
- tkinter2-2.2.2-11.7.3.i386.rpm

Information about python and the package sources can be found at www.python.org.

Lastly, the following external rpms all available at <http://datagrid.in2p3.fr/distribution/external/RPMS> need to be installed on the UI node. They are the customised Condor classads java library version 1.1:

- classads-jar-1.1-2.i386.rpm

the Java 2 Development Kit version 1.4 (or greater):

- j2sdk-1.4.1_01-fcs.i586.rpm
- j2sdk_profile-1.4.1_01-1.noarch.rpm

the Globus Java CoG Kit version 1.0 alpha:

- cog-jar-1.0-1_alpha.i386.rpm

and the Log4J package version 1.2.6:

- log4j-1.2.6-1jpp.noarch.rpm

2.2.4 Prerequisites for the JSS Interface

The JSS interface (a group of C++ processes for the Job Submission Services –JSS- on Roaming Access Server –RAS- machine) is a layer that is “above” the C++ API: its processes call the methods that are provided by the C++ API.

The CrossGrid JSS running on RAS machine are provided as Web Services and are network accessible through http protocol and standardized SOAP messaging (based on Apache Java Axis Toolkit¹). The services are described using standard Web Services Description Language (WSDL) and published on an Apache TomCat web server².

The Web Services module calls the JSS interface included in the WP3.2 distribution to submit all user requests to RB and/or L&B .

The CrossGrid WorkPackage 3.1 takes care of the Web Services implementation: the full installation and the deployment of the web services need the following packages:

- tomcat4-4.1.18-full.1jpp.noarch.rpm
- axis-1.1.tar.gz
- cg-wp3.1-RAS-services-3.1.1-1.noarch.rpm

¹ <http://ws.apache.org/axis/>

² <http://jakarta.apache.org/tomcat/>

These three packages need to be installed on the RAS machine with the WP3.2 set of RPM's for JSS.

In addition, this package provide the Java client of the Web Services:

–cg-wp3.1-JSS-client-3.1.1-1.noarch.rpm

It can be installed on any machine: in the CrossGrid middleware this interface is used by Migrating Desktop and Web Portal (WP3.1).

More detailed information is available on [4].

3 INSTALLATION IN THE CROSSGRID TESTBED

The CrossGrid testbeds are managed by the LCFG deployment support tool. This tool allows an automatic installation of the software on all required nodes.

3.1 RPM LISTS FOR LCFG

If you have a system running the basic lcg-2 configuration, **the first thing to know is that you will be able to communicate with our enhanced RB, but will not be able to submit or work with mpich-g2 jobs.** You must upgrade the different entities with our new packages in order to work properly.

The needed packages are the following:

- For the **UI**: if you have an user interface you can upgrade it with our new packages in order to be able to communicate with our enhanced RB that includes the detailed functionality reported in this document (supports new mpich-g2 jobs).

The steps to take in this case are to substitute the edg-wl installed packages by the ones provided by our task. The essential ones that have include the new functionality are in **bold (Current CrossGrid version is 2.1.15.2.5)** :

```
cg-wp3.2-bypass_gcc3_2_2-2.5.3-23.i486.rpm
cg-wp3.2-chkpt-api_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-common-api-java-interface_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-common-api-java_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-common-api_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-config_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-logging-api-c_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-logging-api-cpp_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-logging-api-sh_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-services-common_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-ui-api-cpp_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-ui-api-java-interface_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-ui-api-java_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-ui-cli_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-ui-config_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-ui-gui_gcc3_2_2-2.1.15.2.5-1.i486.rpm
```

- For the **RB** node: If you want to have our enhanced RB, you have to upgrade with the following packages. **Don't forget to delete the edg related ones.**

```
cg-wp3.2-bypass_gcc3_2_2-2.5.3-23.i486.rpm
cg-wp3.2-common-api_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-config_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-globus-gridftp_gcc3_2_2-1.5-23.i486.rpm
cg-wp3.2-interactive_gcc3_2_2-2.1.15.2.5-1.i486.rpm
```

cg-wp3.2-logging-api-c_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-logging-api-cpp_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-logging-api-sh_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-proxyrenewal_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-services-common_gcc3_2_2-2.1.15.2.5-1.i486.rpm

- The latest packages support the usage of the monitoring plugin integrated with the Resource broker, and this is indicated in the rpms with the string “monplugin” in the Release number:

cg-wp3.2-wm_gcc3_2_2-2.1.15.2.5-1monplugin.i486.rpm

The monitoring plugin is released in its own rpm, with support the monitoring tools already integrated, currently the postprocessing tool. Please note the specific tool plugins may require additional rpms.

The monitoring plugin of the resource broker is released in:

cg-wp3.2-monplugin_gcc3_2_2-2.1.15.2.5-1.i486.rpm

See section [Monitoring Plug-in Support] for more information.

- additionally if the RB node works as LB node, upgrade with:

cg-wp3.2-lbserver_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-locallogger_gcc3_2_2-2.1.15.2.5-1.i486.rpm

- For the **RAS/JSS** node:

For this node all the UI packages are needed (look rpm list before), and some additional ones:

cg-wp3.2-interactive_gcc3_2_2-2.1.15.2.5-1.i486.rpm
cg-wp3.2-jss-cpp-srv_gcc3_2_2-2.1.15.2.5-1.i486.rpm

3.2 PROFILE MODIFICATIONS FOR LCFG

The profiles used for the installation of software are the same as the ones used by edg's RB. Additionally there are some specific CrossGrid parameters for the workload config which are the following:

- **G2LauncherLogDir** specifies the pathname of the MPICH-G2 Launcher log directory. This path is used during job submission to store the different logs produced during MPICH-G2 job execution.

E.g:

```
G2LauncherLogDir = "${EDG_WL_TMP}/jobcontrol/g2launcher";
```

- **CheckinTimeout** represents the time (in seconds) to abort a MPICH-G2 jobs if all the subjobs are not executed. It defaults to 300 seconds if not defined.

E.g.:

```
CheckinTimeout = 700;
```

- **G2MultipleSubmit** specifies the way G2 subjobs are launched. If True, only one rsl is submitted to each site for execution. If False, every subjob is submitted as an independent rsl. It defaults to true.

E.g.:

```
CG2MultipleSubmit = True;
```

- **CondorStatus** specifies the condor_status command.

E.g.:

```
CondorStatus = ${CONDORG_INSTALL_PATH}/bin/condor_status;
```

- **CondorConfigVal** specifies the condor_config_val command.

E.g.:

```
CondorConfigVal =  
${CONDORG_INSTALL_PATH}/bin/condor_config_val;
```

- **GlideInIdleTime** specifies the time (in minutes) the remote glide-in will wait for a job. Every time a job finishes the remote glide-in will wait this time and exit normally if no more jobs are sent to the machine, setting the machine free for PBS again. It defaults to 20 minutes.

E.g.:

```
GlideInIdleTime = 10;
```

Condor configuration is also affected in the RB version which incorporates the Glide-In mechanism. The following parameters need to be set:

- **NEGOTIATOR_HOST** refers to the hostname of the machine where the condor_negotiator is running for your pool. Normally it is defined with the \$(CONDOR_HOST) macro. Also, you can change the port that the negotiator runs on. By default, the negotiator uses port 9614, but you can set the port with a ":port". This port should be in the GLOBUS_TCP_PORT_RANGE to allow communication between RB machine and WorkerNodes

E.g.:

```
NEGOTIATOR_HOST = $(CONDOR_HOST):22000
```

- **COLLECTOR_HOST** refers to the hostname of the machine where the condor_collector is running for your pool. Normally it is defined with the \$(CONDOR_HOST) macro. Also, you can change the port that the collector runs on. By default, the collector uses port 9618, but you can set the port with a ":port". This port should be in the GLOBUS_TCP_PORT_RANGE to allow communication between RB machine and WorkerNodes

E.g.:

```
COLLECTOR_HOST = $(CONDOR_HOST):22001
```

- **DAEMON_LIST** refers to condor daemons that will run in the RB machine. master, schedd, collector and negotiator are needed for glide-In.

E.g.:

```
DAEMON_LIST = MASTER, SCHEDD, COLLECTOR, NEGOTIATOR
```

- **LOWPORT and HIGHPORT** refers to the range of ports that Condor will use. This may be useful if you are behind a firewall. This range should be inside the `GLOBUS_TCP_PORT_RANGE`

E.g.:

```
LOWPORT = 21000
HIGHPORT = 24000
```

- **HOSTALLOW_WRITE** refers to all hosts that can initialize a glidein process, all hosts must be separated with commas. This option should contain all WorkerNodes of testbed.

E.g.:

```
HOSTALLOW_WRITE = *
```

- **GLIDEIN_SERVER_NAME** refers to the hostname of the server which stores the glidein binaries. This should be the RB machine

E.g.:

```
GLIDEIN_SERVER_NAME = aorbgrid.uab.es
```

- **GLIDEIN_SERVER_DIR** refers to the directory in the server which stores the glidein binaries. In this directory must be created a tarball with the condor binaries needed to execute the Glide-In, in next section is an example of how to create this tarball.

E.g.:

```
GLIDEIN_SERVER_DIR = /opt/condor/glidein
```

3.3 MANUAL POST INSTALLATION STEPS

Glide-In mechanism uses a package of Condor binaries which the WorkerNodes download every time they are needed. This package is not created automatically, but the following script will do it:

```
#!/bin/sh
# create glide-in package for download

condor_v=`condor_version | head -1 | cut -d" " -f2`
machine=`uname -m`
system=`uname -s | tr 'A-Z' 'a-z'`
release=`uname -r | cut -d"." -f1,2`

file=/opt/condor/glidein/$condor_v-$machine-$system-$release.tar.gz
mkdir /opt/condor/glidein
tar -czf $file -C /opt/condor/glidein condor_master condor_startd condor_starter
condor_starter.std
```

4 MANUAL INSTALLATION

For manual installation consult instructions for installing manually a RB at <http://lcgdeploy.cvs.cern.ch/cgi-bin/lcgdeploy.cgi/lcg2/manual-install/RB/>. The only differences between CrossGrid are the packages needed which can be found in section 3, and the configuration parameters specified in section 4.4.1

4.1 DOWNLOAD

The easiest way to obtain the software is to access the CVS repository at FZK (gridportal.fzk.de) and to get the source code for the task3.2.

Rpm files with the compiled code are expected to reside at the central software repository (<http://gridportal.fzk.de/distribution/CrossGrid/releases/allfiles/7.3/cg/wp3/>); where the official Complete CrossGrid releases are available to download.

Releases can be obtained also from our task directly (http://gridportal.fzk.de/distribution/CrossGrid/CrossGrid/wp3/wp3_2-scheduling/)

4.2 INSTALLATION FROM RPM

The complete list of rpms can be found in the manual install guide and section 3 of this document.

4.3 INSTALLATION FROM SOURCE

Since CrossGrid middleware is based on DataGrid, basic requirements for building the software are the same.

In Section 3 of [1] some requirements are outlined for the installation.

The EDG WP1 and consequently CrossGrid middleware software runs and has been tested on platforms running *Globus Toolkit 2.2* on top of *Linux RedHat 7.3*.

4.3.1 External dependencies

We use a “standard” RB machine to compile our prototypes, so you can use this machine as a base to compile our release. Any rpm package installed there is in principle needed. This is a long list of packages so you can refer to the list of packages used by the lcfg to install that machine. In addition to that, some packages are needed to compile properly:

PACKAGES

rpm packages to compile crossgrid 2.1.15.2.2 based code
updated to June 10th, 2004

VOMS

voms-api_gcc3_2_2-1.2.0.14-1_RH7.3
voms-api-gcc32dbgpthr-gcc3_2_2-1.2.0.14-1_RH7.3

GACL

gacl-0.9.2-1_gcc3_2_2
gacl-devel-0.9.2-1_gcc3_2_2

RPM

rpm-4.0.4-7x.18
rpm-python-4.0.4-7x.18
rpm-devel-4.0.4-7x.18
rpm-build-4.0.4-7x.18

SECURITY_J

edg-java-security-1.5.9-1
bouncycastle-jdk14-1.19-2

COG

cog-jar-1.1-1

CONDOR

condor-6.6.0-2.edg4
on lcg2 machines (with condor-6.6.0-2.edg4) use condor-gcc-3.2.2-1.2.1-2
otherwise use condor-gcc-3.2.2-1.2.1-1 (compilation on lcg1 machines)

CLASSADS

classads-jar-1.1-2
classads-g3-0.9.4-vh8

BOOST

boost-g3-1.29.1-vh6

VDT/GLOBUS

vdt_compile_globus_core-VDT1.1.13-b5

REPLICA

edg-replica-manager-client-c++_gcc3_2_2-devel-2.2.9-1 (did not find v 2.2.10)

RGMA

edg-rgma-api-java-3.2.4-1
edg-rgma-common-3.2.4-1
netlogger-jar-1.0.0-1

SWIG

swig-1.3.19-1

ARES

ares-devel-1.1.1-2

EXPAT

expat-1.95.2-2

expat-devel-1.95.2-2

XERCES

xerces-c-1.7.0-3_gcc3_2_2

xerces-j1-1.4.4-12jpp

MYPROXY

edg-lcfg-myproxy-1.1.3-edg1

myproxy-VDTALT1.1.8-14.edg4

myproxy-config-1.1.8-13.edg1

PYTHON 2

python2-2.2.2-11.7.3.i386.rpm

tkinter2-2.2.2-11.7.3.i386.rpm

MYSQL

MySQL-4.0.x for pc-linux-gnu (i686)

LOG4j

log4j-1.2.6-1jpp

EDG JAVA SECURITY

edg-java-security-client-1.5.9

PERL

Perl 5

OTHERS

gcc version 3.2.2

GNU make version 3.79.1 or higher

GNU autoconf version 2.13

GNU libtool 1.4.2

GNU automake 1.4

GNU m4 1.4 or higher

ENVIRONMENT VARIABLES

EDG_RPM_RELEASE_VERSION=_gcc3_2_2 (in order to generate rpm names correctly)

Packages should be installed in order to compile properly. In addition some changes are needed in the machine that builds the software:

- Change file permissions due to wrong file permissions of the installed packages:
 1. `chmod -R +rx /opt/condor`
 2. `chmod +r /opt/globus/etc/globus_packages/globus_opendap`
- Execute `/opt/globus/setup/globus/setup-globus-common` to create some needed files like `globus-sh-tools-vars.sh`
- Create symbolic links to the gcc-3.2.2 compiler, in the directory `/opt/gcc-3.2.2/bin/` :
 1. `ln -s c++-3.2.2 c++`
 2. `ln -s g++-3.2.2 g++`
 3. `ln -s gcc-3.2.2 gcc`

You have also to use this compiler by default, putting this directory at the beginning of the PATH. These changes are needed because of the configure scripts that are not able to work properly with the gcc-3.2.2 compiler if you don't make the links.

Another option is to set the environment variables CC and CXX to the respective compilers, but additional changes are needed to some config files that we are not going to describe here

- Rename the file `/opt/gcc-3.2.2/lib/libstdc++.la` to `/opt/gcc-3.2.2/lib/libstdc++.la.SAVE`.
This is needed in order to not get an error compiling the `common/utilities/directory` and is a known DataGrid issue. Maybe in the future they will correct the package `gcc3-3.2.2-edg1`
- add `/opt/gcc-3.2.2/lib` to the `LD_LIBRARY_PATH`

Following the guidelines described at section 3 of [1] we present a build procedure for all the source tree.

4.3.2 Setting environment variables

Before starting the compilation, some environment variables related to the WMS components can be set or configured by means of the *configure* script. This is needed only if package defaults are not suitable. Involved variables are listed below:

- `GLOBUS_LOCATION` base directory of the Globus installation
The default path is `/opt/globus`.
- `MYSQL_INSTALL_PATH` base directory of the MySQL installation
The default path is `/usr`.
- `EXPAT_INSTALL_PATH` base directory of the Expat installation.
The default path is `/usr`.
- `CLASSAD_INSTALL_PATH` base directory of the Classad library installation. The default path is `/opt/classads`.

- CONDORG_INSTALL_PATH base directory of the Condor installation.
The default path is */opt/CondorG*.
- PYTHON_INSTALL_PATH base directory of the Python installation.
The default path is */usr*.
- MYPROXY_INSTALL_PATH base directory of the MyProxy installation .
The default path is */usr/local*.

In order to build the *whole* WP3.2 package, all the environment variables in the previous list must be set.

To build the *User Interface* module, the environment variables that need to be set are the following:

- GLOBUS_LOCATION
- CLASSAD_INSTALL_PATH
- PYTHON_INSTALL_PATH
- EXPAT_INSTALL_PATH

If you plan to build the “rb node” services, variables to be set are:

- GLOBUS_LOCATION
- MYSQL_INSTALL_PATH
- EXPAT_INSTALL_PATH
- CLASSAD_INSTALL_PATH
- CONDORG_INSTALL_PATH

The *LB server* and *Local Logger* modules need the following environment variables to be built:

- GLOBUS_LOCATION
- MYSQL_INSTALL_PATH
- EXPAT_INSTALL_PATH

Finally, the *LB APIs* module needs:

- GLOBUS_LOCATION
- EXPAT_INSTALL_PATH

4.3.3 Compiling the code

The first thing to do to compile the software is to download the source tree, and change into the *workload* directory. The next steps are:

Configure the machine-dependent variables for the compilation:

Run the following commands:

```
./autogen.sh
```

At this point the *configure* command can be run. The configure script has to be invoked as follows:

```
./configure <options>
```

For compiling all the code with the standard configuration you can type the command without options. The list of options, including options for modifying the installation path and defining the path for locating the required software, can be found with `-help` option. During the configure step, the different Makefiles for compiling the software are created.

Compile the code:

Once the configure script has terminated its execution, check that the *make* from the GNU distribution is in your path and then always in the workload source code directory run:

```
make
```

then:

```
make apidoc
```

and then:

```
make check
```

to build the test code. If the two previous steps complete successfully, the installation of the software can be performed. In order to install the package in the installation directory specified either by the `--prefix` option of the *configure* script or by the default value (i.e. `/opt/edg`), you can now issue the command:

```
make install
```

It is possible to run "make clean" to remove object files, executable files, library files and all the other files that are created during "make" and "make check". The command:

```
make -i dist
```

can be used to produce in the workload's base directory, a binary gzipped tar ball of the Workload distribution. This tar ball can be both transferred on other platforms and used as source for the RPM creation.

For creating the RPMs for workload 1.0 (according to the configure options you have used) make sure that your `PATH` is set in such a way that the *GNU autotools*, *make* and the *gcc* compiler can be used and edit the file `$(HOME)/rpmmacros` (if this file does not exist in your home directory, then you have to create it) to set the following entry:

```
%_topdir      <your home dir>/rpm/redhat
```

Then you can issue the command:

```
make rpm
```

that generates the RPMs in `$(HOME)/rpm/redhat/RPMS`

4.4 CONFIGURATION

4.4.1 List of configuration files

Once the rpm has been installed, the "rb node" services must be properly configured. This is done as in EDG, editing the file `$(EDG_WL_CONFIG_DIR)/edg_wl.conf` file. If `$(EDG_WL_CONFIG_DIR)` hasn't been defined, the `edg_wl.conf` file is looked for first in `/opt/edg/etc`, then in `/etc`, and then in `/usr/local/etc`.

Once the rpm has been installed, the "rb node" services must be properly configured. This is done as in EDG, editing the file `$(EDG_WL_CONFIG_DIR)/edg_wl.conf` file. If `$(EDG_WL_CONFIG_DIR)` hasn't been defined, the `edg_wl.conf` file is looked for first in `/opt/edg/etc`, then in `/etc`, and then in `/usr/local/etc`.

4.4.2 Editing the configuration files

The configuration file is composed of 5 parts:

- one for the "common" (i.e. "used" by all services) attributes
- one for the configuration of the NS
- one for the configuration of the WM
- one for the configuration of the JC
- one for the configuration of the LM

The JC part is the only one which differs from the EDG ones, see Section 4.2.2 of [1] to

Configuration of the Job Controller is accomplished editing the configuration file and setting opportunely the attributes in the:

```
JobController = [
```

```
...  
...  
/;
```

section.

These attributes are listed in Section 4.2.2.4 of [1]. For CrossGrid, there are 2 new attributes listed hereafter:

- **G2LauncherLogDir** specifies the pathname of the MPICH-G2 Launcher log directory. This path is used during job submission to store the different logs produced during MPICH-G2 job execution.

E.g:

```
G2LauncherLogDir = "${EDG_WL_TMP}/jobcontrol/g2launcher";
```

- **CheckinTimeout** represents the time (in seconds) to abort a MPICH-G2 jobs if all the subjobs are not executed. It defaults to 300 seconds if not defined.

E.g.:

```
CheckinTimeout = 700;
```

- **G2MultipleSubmit** specifies the way G2 subjobs are launched. If True, only one rsl is submitted to each site for execution. If False, every subjob si submitted as an independent rsl. It defaults to true.

E.g.:

```
CG2MultipleSubmit = True;
```

- **CondorStatus** pecifies the condor_status command.

E.g.:

```
CondorStatus = ${CONDORG_INSTALL_PATH}/bin/condor_status;
```

- **CondorConfigVal** specifies the condor_config_val command.

E.g.:

```
CondorConfigVal =  
${CONDORG_INSTALL_PATH}/bin/condor_config_val;
```

- **GlideInIdleTime** specifies the time (in minutes) the remote glide-in will wait for a job. Every time a job finishes the remote glide-in will wait this time and exit normally if no more jobs are sent to the machine, setting the machine free for PBS again. It defaults to 20 minutes.

E.g.:

```
GlideInIdleTime = 10;
```

4.4.3 Startup scripts

The software can be started by using the following scripts in /etc/init.d:

```
edg-wl-wm
```

edg-wl-ns
edg-wl-jc
edg-wl-lm
edg-wl-locallogger
edg-wl-lbserver

This services should be enabled in the init runlevel 5 or 3 of the “RB node”

4.4.4 Other requirements

4.4.4.1 Environment

Environment variables that have to be set (or can be set) for the NS, WM, JC and LB services are listed hereafter:

- **EDG_WL_LOG_DESTINATION**
The Logging library i.e. the library providing APIs for logging job events to the LB reads its immediate logging destination from the environment variable `EDG_WL_LOG_DESTINATION`.
- **CONDOR_CONFIG**
This variable has to refer to the CondorG configuration file, usually `/opt/condor/etc/condor_config`
- **EDG_WL_CONFIG_DIR**
As explained in section , this variable refers to the directory where the configuration file for the WMS services running on the “RB node” (`edg_wl.conf`) is available.
- **GRIDMAP**
This variable must refer to the grid-mapfile (usually `/etc/grid-security/grid-mapfile`)
- **LD_LIBRARY_PATH**
Should include `$GLOBUS_LOCATION/lib`, the Boost lib directory and the gcc 3.2 lib directory
- **EDG_LOCATION**
Should refer to the EDG software installation directory (usually `/opt/edg`): needed for the WP2 services used by the RB

Then of course, if some environment variables are used in the NS/WM/JC/LM configuration sections, they have of course to be set as well.

Anyway, all variables that must be defined for the proper execution of the WMS services, are set by the relevant start-up scripts.

4.4.4.2 Users

The RB services uses an special user specified by the environment variable EDG_WL_USER. By default this user is named “edguser” and has to have access to the software installation path.

4.4.4.3 Ports

The different ports needed by the RB node components are configurable in the `edg_wl.conf` file. This ports should be open both inbound and outbound for TCP connections. Additionally the `GLOBUS_PORT_RANGE` (usually ports between 20000 and 25000) must be open for inbound and outbound connections.

4.4.4.4 Certificates

The RB node needs a certificate in order to run properly.

4.4.4.5 Folders

The different startup scripts create folders under `/var/edgwl` automatically upon start up of services.

5 RUNNING AND TESTING

5.1 RUNNING THE RESOURCE BROKER

The Resource Broker is started using the startup services described in section 4.4.3. They should be added to the default machine init runlevel and not started manually.

5.2 TESTING THE NEW FEATURES

5.2.1 SMP support for MPICH-P4 Jobs

The new scheduler supports the improved execution of mpich-p4 jobs on testbeds with SMP machines. This feature is enabled automatically whenever the testbed selected to execute the job has this kind of computers, so there is no need to specify any kind of new options in the JDL file describing the job.

5.2.2 Multiple CEs selection

One of the new features is the selection of multiple CEs for submitting a mpi job to the selected resources.

1.- What we need first is a JDL file defining the mpi job that we want to submit to the resource broker. This jdl file shall contain the specifications and requirements of the job, and also the new fields that we have established to correctly define the mpi jobs.

This new fields are:

| | |
|----------------|--|
| JobType | Field that defines that is a mpi job. Possible values are: |
| | mpich - defines a mpich-p4 job |
| | mpich-g2 - defines a mpich-g2 job |
| | normal - (default) common sequential job |

NodeNumber Field that defines the required number of cpus to execute the mpi job

For testing purposes we show one jdl file, for example:

```
VirtualOrganisation = "cg";
Executable          = "mpi_app";
JobType             = "mpich-g2";
NodeNumber         = 10;
Arguments          = "-n";
StdOutput          = "std.out";
StdError           = "std.err";
Requirements       =
other.GlueCEInfoLRMSType=="pbs";
Rank               = other.GlueHostBenchmarkSI00;
InputSandbox       = {"mpi_app"};
OutputSandbox      = {"std.out", "std.err"};
```

We can see that in this description we are looking for groups of CEs whose queue type is PBS and that have at least 10 free CPUs counting all the involved CEs in the group, to run our mpi job.

2 .-Next we will submit the jdl file to the modified RB that supports this new syntax in the jdl file. The command to get the available CEs would be, as usual:

edg-job-list-match file.jdl

With this command we are sending the jdl file to the RB to obtain the list of available CE. The output of the command will give the following results:

CROSSGRID INSTALLATION GUIDE
Grid Resource Management

Connecting to host cg07.ific.uv.es, port 7772

GROUPS OF CE IDs LIST

The following groups of CE(s) matching your job requirements have been found:

| *Groups with 1 CEs* | *TotalCPUs* | *FreeCPUs* |
|---|-------------|------------|
| [Rank=650] | | |
| ce001.grid.ucy.ac.cy:2119/jobmanager-pbs-infinite | 10 | 10 |
| [Rank=650] | | |
| ce001.grid.ucy.ac.cy:2119/jobmanager-pbs-long | 10 | 10 |
| [Rank=650] | | |
| ce001.grid.ucy.ac.cy:2119/jobmanager-pbs-short | 10 | 10 |
| [Rank=630] | | |
| cluster.ui.sav.sk:2119/jobmanager-pbs-workq | 16 | 16 |
| [Rank=400] | | |
| zeus24.cyf-kr.edu.pl:2119/jobmanager-pbs-infinite | 58 | 57 |
| [Rank=400] | | |
| zeus24.cyf-kr.edu.pl:2119/jobmanager-pbs-long | 58 | 57 |
| [Rank=400] | | |
| zeus24.cyf-kr.edu.pl:2119/jobmanager-pbs-short | 58 | 57 |

| *Groups with 2 CEs* | *TotalCPUs* | *FreeCPUs* |
|--|-------------|------------|
| [Rank=440 TotalCPUs=12 FreeCPUs=12] | | |
| cagnode45.cs.tcd.ie:2119/jobmanager-pbs-infinite | 4 | 4 |
| ce100.fzk.de:2119/jobmanager-pbs-long | 8 | 8 |
| [Rank=498 TotalCPUs=10 FreeCPUs=10] | | |
| ce01.lip.pt:2119/jobmanager-pbs-infinite | 2 | 2 |
| ce100.fzk.de:2119/jobmanager-pbs-long | 8 | 8 |
| [Rank=433.6 TotalCPUs=10 FreeCPUs=10] | | |
| ce100.fzk.de:2119/jobmanager-pbs-long | 8 | 8 |
| cg01.ific.uv.es:2119/jobmanager-pbs-infinite | 2 | 2 |
| [Rank=448 TotalCPUs=10 FreeCPUs=10] | | |
| ce100.fzk.de:2119/jobmanager-pbs-long | 8 | 8 |
| cgnode00.di.uoa.gr:2119/jobmanager-pbs-infinite | 2 | 2 |
| [Rank=498 TotalCPUs=12 FreeCPUs=10] | | |
| ce100.fzk.de:2119/jobmanager-pbs-long | 8 | 8 |
| cms.fuw.edu.pl:2119/jobmanager-pbs-infinite | 4 | 2 |
| [Rank=566.667 TotalCPUs=12 FreeCPUs=12] | | |
| cagnode45.cs.tcd.ie:2119/jobmanager-pbs-infinite | 4 | 4 |
| xgrid.icm.edu.pl:2119/jobmanager-pbs-infinite | 8 | 8 |
| [Rank=650 TotalCPUs=10 FreeCPUs=10] | | |
| ce01.lip.pt:2119/jobmanager-pbs-infinite | 2 | 2 |
| xgrid.icm.edu.pl:2119/jobmanager-pbs-infinite | 8 | 8 |
| [Rank=555 TotalCPUs=16 FreeCPUs=16] | | |
| ce100.fzk.de:2119/jobmanager-pbs-long | 8 | 8 |
| xgrid.icm.edu.pl:2119/jobmanager-pbs-infinite | 8 | 8 |
| [Rank=585.6 TotalCPUs=10 FreeCPUs=10] | | |
| cg01.ific.uv.es:2119/jobmanager-pbs-infinite | 2 | 2 |
| xgrid.icm.edu.pl:2119/jobmanager-pbs-infinite | 8 | 8 |
| [Rank=600 TotalCPUs=10 FreeCPUs=10] | | |
| cgnode00.di.uoa.gr:2119/jobmanager-pbs-infinite | 2 | 2 |
| xgrid.icm.edu.pl:2119/jobmanager-pbs-infinite | 8 | 8 |
| [Rank=650 TotalCPUs=12 FreeCPUs=10] | | |
| cms.fuw.edu.pl:2119/jobmanager-pbs-infinite | 4 | 2 |
| xgrid.icm.edu.pl:2119/jobmanager-pbs-infinite | 8 | 8 |

The obtained command output contains a list of resources or groups of resources where to execute the parallel job. Such list is composed of:

- a) Groups of elements that contain only 1 CE, so the job could be submitted to just one CE or cluster. This is the best desirable situation. We have 7 groups of 1 CE, being the first best resource is:

```
*Groups with 1 CEs*                *TotalCPUs* *FreeCPUs*

[Rank=650]
ce001.grid.ucy.ac.cy:2119/jobmanager-pbs-infinite    10        10
```

The CE **ce001.grid.ucy.ac.cy:2119/jobmanager-pbs-infinite** has all 10 free CPUs and a global rank (based on the SI00 of that CE) of 650. This resource will be the first selected by the Application Scheduler.

As it can be seen, the next resources that would be selected would be:

```
ce001.grid.ucy.ac.cy:2119/jobmanager-pbs-long
ce001.grid.ucy.ac.cy:2119/jobmanager-pbs-short
cluster.ui.sav.sk:2119/jobmanager-pbs-workq
zeus24.cyf-kr.edu.pl:2119/jobmanager-pbs-infinite
zeus24.cyf-kr.edu.pl:2119/jobmanager-pbs-long
zeus24.cyf-kr.edu.pl:2119/jobmanager-pbs-short
```

- b) After single CEs, groups of CEs that fulfill the requirements are formed. In this case we find 11 groups with 2 CEs suitable for executing our job. The best one, according with the rank is:

```
*Groups with 2 CEs*                *TotalCPUs* *FreeCPUs*

[Rank=650 TotalCPUs=10 FreeCPUs=10]
ce01.lip.pt:2119/jobmanager-pbs-infinite            2          2
xgrid.icm.edu.pl:2119/jobmanager-pbs-infinite       8          8
```

This groups is the one with more computed Rank from the groups with 2 CEs. It has also its 10 needed cpus free (from a total of 10) and a computed rank of 650. It is again calculated with the weighted rank of every CE of the group (**ce01.lip.pt:2119/jobmanager-pbs-infinite** with a rank of 650, and **xgrid.icm.edu.pl:2119/jobmanager-pbs-infinite** also with rank 650 makes the weighted rank of the same 650, no matter how many cpus are contributing with).

We can see another meaningful example in the group:

```
[Rank=440 TotalCPUs=12 FreeCPUs=12]
cagnode45.cs.tcd.ie:2119/jobmanager-pbs-infinite    4          4
ce100.fzk.de:2119/jobmanager-pbs-long              8          8
```

This group formed by 2 CEs from **tcd** and **fzk** have a total number of 12 CPUS, from where all of them are free (without running jobs). This groups fulfils the requirements, having also at least the cpus wanted.

The computed rank of 440 is not the average of the ranks of the components (400 and 460, that would be 430) but the weighted rank calculated considering the number of free cpus of each component.

c) As we can see there are no possible groups with 3 CEs that group the required number of cpus that we are asking for, and that are not taken into account in the previous groups (with 1, or 2 CEs). However we can find 4 groups with 4 CEs each one, and this is the biggest that we can from with the CEs in the testbed and the situation in the moment that we are asking. The best group would be:

| *Groups with 4 CEs* | *TotalCPUs* | *FreeCPUs* |
|--|-------------|------------|
| [Rank=500 TotalCPUs=12 FreeCPUs=10] | | |
| cagnode45.cs.tcd.ie:2119/jobmanager-pbs-infinite | 4 | 4 |
| ce01.lip.pt:2119/jobmanager-pbs-infinite | 2 | 2 |
| cgnode00.di.uoa.gr:2119/jobmanager-pbs-infinite | 2 | 2 |
| cms.fuw.edu.pl:2119/jobmanager-pbs-infinite | 4 | 2 |

This group if formed by CEs from **tcd**, **lip**, **uoa** and **fuw**, collecting 10 free cpus from a total of 12. The original Ranks from every site are (not shown in the logs):

| | |
|--|------------------------|
| cagnode45.cs.tcd.ie:2119/jobmanager-pbs-infinite | 400 (contributes 4/10) |
| ce01.lip.pt:2119/jobmanager-pbs-infinite | 650 (contributes 2/10) |
| cgnode00.di.uoa.gr:2119/jobmanager-pbs-infinite | 400 (contributes 2/10) |
| cms.fuw.edu.pl:2119/jobmanager-pbs-infinite | 650 (contributes 2/10) |

Giving the weighted computed rank of 500.

For a more detailed explanation of how the process of resource selection is carried, document [2] can be consulted.

Finally, the command used to submit such file is:

```
edg-job-submit file.jdl
```

The output of the command will give the following results:

```
Connecting to host aow5grid.uab.es, port 7772
Logging to host aow5grid.uab.es, port 9002
```

```
*****
```

JOB SUBMIT OUTCOME

The job has been successfully submitted to the Network Server.
Use `edg-job-status` command to check job current status. Your job identifier (`edg_jobId`) is:

- `https://aow5grid.uab.es:9000/jR0hjTzOlyFkRkpP_i1R8Q`

This output indicates that the job has been sent to the RB and now the user must check its status using the `edg-job-status` command. The job will pass through three different states: Waiting, Running and Done. If there is any kind of error during the execution of the job and the application launcher is not able to get the checkins from all the subjobs, the job will be aborted and the `edg-job-status` command will show an output like this:

BOOKKEEPING INFORMATION:

```
Printing status info for the Job :
https://aorbgrid.uab.es:9000/PQBDca1SUaDCAEJooBtWag
Current Status:      Aborted
Status Reason:      Could not receive all checkins from subjobs
Destination:        ce.grid.cesga.es:2119/jobmanager-pbs-infinite
reached on:         Thu Mar 25 12:12:36 2004
```

When the job has finished, the user can get the output using the command `edg-job-get-output`, this command output is depicted next:

Retrieving files from host `aow5grid.uab.es`

JOB GET OUTPUT OUTCOME

```
Output sandbox files for the job:
- https://aow5grid.uab.es:9000/jR0hjTzOlyFkRkpP_i1R8Q
have been successfully retrieved and stored in the directory:
/tmp/jobOutput/jR0hjTzOlyFkRkpP_i1R8Q
```

In the directory specified by the `edg-job-get-output` can be found the output and error files of each subjob of the application

5.2.3 Interactive MPICH-P4 and MPICH-G2 Jobs support

Another new feature is the support of interactive `mpich-p4` and `mpich-g2` jobs. The source code of interactive programs for the `mpich-p4` and `mpich-g2` testing can be downloaded from cvs at:

- http://savannah.fzk.de/cgi-bin/viewcvs.cgi/CrossGrid/CrossGrid/wp3/wp3_2-scheduling/etc/tests/interactive-tests/

1. First of all, we need to define this feature in the job descriptor file, to this purpose we will use the JobType field. In order to do this we will write one of the next attributes:

| | | |
|----------------|-------------------------------|---------------------------------------|
| JobType | { "interactive", "mpich" } | - Defines an interactive mpich-p4 job |
| | { "interactive", "mpich-g2" } | - Defines an interactive mpich-g2 job |

We can see that this field uses the same attributes than the interactive jobs and mpich-p4/mpich-g2 jobs, the difference resides that now we use them simultaneously.

Note#1: In the JDL file, we must specify both the fields which define an mpich-p4/g2 job and the ones dealing with interactivity. e.g.: ListenerPort for an interactive Job and NodeNumber for an mpich job.

Note#2: As in interactive jobs we must not define OutputSandbox, StdOutput and StdError attributes.

For testing this new feature we show one jdl file, for example:

```
VirtualOrganisation = "cg";
Type                = "Job";
Executable          = "interactive_mpich-g2_app";
JobType             = { "interactive", "mpich-g2" };
NodeNumber         = 10;
ListenerPort       = 24100;
Arguments           = "-n";
FuzzyRank          = true;
InputSandBox       = {"interactive_mpich-g2_app"};
Requirements       = -
Other.GlueCEStateEstimatedResponseTime;
Rank               = other.GlueCEStateStatus ==
"Production";
```

2. Next, we submit the jdl file using the modified UI to the modified RB that supports this new feature.

edg-job-submit file.jdl

The output of the command will give the following results:

```
Selected Virtual Organisation name (from JDL): cg
Connecting to host aorbgrid.uab.es, port 7772
Logging to host aorbgrid.uab.es, port 9002

*****

                        JOB SUBMIT OUTCOME

The job has been successfully submitted to the Network Server.
Use edg-job-status command to check job current status. Your job
identifier (edg_jobId) is:

- https://aorbgrid.uab.es:9000/IAYUQS7E6J4aySd3bjImVQ

---

The Interactive Session Listener has been successfully launched
with the following parameters:

Host:                aouigrid.uab.es
Port:                24501

*****

*****

Interactive Job console started for
https://aorbgrid.uab.es:9000/IAYUQS7E6J4aySd3bjImVQ
Please press ^C to exit from the session
*****
```

This output indicates that the job has been sent to the RB and now the user must wait for the beginning of the job execution.

When the job is running, if it is an interactive mpich-g2 job, the output should be similar to the following:


```
Job Status: Aborted (Some subjobs got an error
while in the CondorG queue)
*****
Interactive Session was aborted.
Removing Listener and input/output streams...
Done
Press <enter> to go to prompt
*****
```

However, if the user cancelled the job, by pressing ctrl+c, appears the next message in the console:

```
*****
Interactive Session ended by user.
Removing Listener and input/output streams...
Done
Press <enter> to go to prompt
*****
```

If the job was cancelled, by the user, the job must be removed using the command *edg-job-cancel*, but if the job was not executing in the remote host still, it can be restored with *the edg-job-attach* command:

```
edg-job-attach https://aorbgrid.uab.es:9000/ta84GWScp0qnuTH3g2N8yQ
*****
JOB ATTACHED:
The Interactive Session Listener has been successfully launched
with the following parameters:
---
Host:                aouigrid.uab.es
Port:                24501
*****

*****
Interactive Job console started for
    https://aorbgrid.uab.es:9000/ta84GWScp0qnuTH3g2N8yQ
Please press ^C to exit from the session
*****
```

From then on, the job will continue its execution exactly as if the *edg-job-submit* command had been used.

5.2.4 Monitoring Plug-in Support

We have developed an interface for integrating the monitoring tools developed in the Crossgrid project to be used by the final user. The basic ideas for the APIs and the working mode can be found in the document [3].

Currently this API has been implemented by the Postprocessing tool and has been integrated in the latest release of our packages.

For installing the support for this, the related rpm packages that are modified are the following.

- The latest packages support the usage of the monitoring plugin integrated with the Resource broker, and this is indicated in the rpms with the string “monplugin” in the Release number:

cg-wp3.2-wm_gcc3_2_2-2.1.15.2.5-1monplugin.i486.rpm

- The monitoring plugin support itself is released in its own rpm, with the requirements of the packages monitoring tools already integrated, currently the postprocessing tool. The monitoring plugin of the resource broker is released in:

cg-wp3.2-monplugin_gcc3_2_2-2.1.15.2.5-1.i486.rpm

Additionally it is required the specific monitoring tools and the implemented monitoring plugin that we want to use, currently the postprocessing tools. The particular rpms can be found in the particular installation guide of the ppt tool, but for reference this are the rpms tested:

cg-wp3_2_2-gmdat-client-libs-cvs2004101419-1.i386.rpm

For the final user, this monitoring tools can be tested using the provided functions in the Rank sections of the JDL of their jobs. The final specification of this functions, and the related parameter and metrics used, will be provided by the particular tool plugin.

As a simple example this JDL can be provided to test the integration of this tool:

```
VirtualOrganisation = "cg";
Type                = "Job";
Executable          = "interactive_mpich-g2_app";
JobType             = { "interactive", "mpich-g2" };
NodeNumber         = 10;
ListenerPort       = 24100;
Arguments          = "-n";
FuzzyRank          = true;
InputSandBox       = {"interactive_mpich-g2_app"};
Requirements       = - other.GlueCEStateEstimatedResponseTime;
Rank               =
  ppt_getClusterParameter("max", "idle_bogomips", other.GlueCeUniqueId, 0) *
  ppt_getNetworkParameter("avg", "bandwidth", other.groupGlueCeUniqueId, 0);
```

Additional examples may be found in the ppt user guide.

5.2.5 Glide-In Mechanism

To provide support for high priority jobs (i.e. interactive jobs), the CrossGrid Resource Broker has introduced the use of Glide-In mechanism from Condor. Using this feature, the RB can control every WorkerNode where a batch job is been executing and send high priority jobs to those machine while lowering the priority of the batch job. To achieve this, every “normal” job is considered to be a batch job and, instead of being submitted as a stand-alone job, it is submitted using a Condor Glide-In. This involves a two step submission:

1. Submit a Condor Glide-In to the selected machine. The RB submits a Condor Glide-In to the selected machine using Globus. Once started, the Condor Glide-In will create two virtual machines on the WorkerNode, one with low priority for batch jobs (vm2) and another one with high priority for interactive jobs (vm1).
2. The actual job is submitted to the selected WorkerNode using Condor. The job will run as normally but it can be preempted to allow the execution of a high priority application.

The virtual machines created can be monitorized using the `condor_status` command in the RB machine. In the following figure it is depicted one example of the `condor_status` command:

```
$ condor_status
```

| Name | OpSys | Arch | State | Activity | LoadAv | Mem | ActvtyTime |
|---------------|-------|-------|---------|----------|--------|------------|------------|
| vm1@17181@cgw | LINUX | INTEL | Owner | Idle | 0.000 | 313[?????] | |
| vm2@17181@cgw | LINUX | INTEL | Claimed | Busy | 0.000 | 313[?????] | |

| | Machines | Owner | Claimed | Unclaimed | Matched | Preempting |
|-------------|----------|-------|---------|-----------|---------|------------|
| INTEL/LINUX | 2 | 1 | 1 | 0 | 0 | 0 |
| Total | 2 | 1 | 1 | 0 | 0 | 0 |

In the example two virtual machines are available (only one job has been submitted) and one of them, vm2, is busy executing the job. For every WorkerNode there will be a pair of virtual machines: vm1 for high priority jobs and vm2 for low priority jobs.

Interactive jobs will be treated in a different way. The Resource Broker will always try to allocate a free WorkerNode (or CPU) to execute that job. If there is one CPU available which meets the job requirements, the job will be submitted there without any Glide-In to allow a fast start-up and execution of the application. Otherwise, if there isn't any CPU available which meets the job requirements, the Resource Broker will search for glide-in machines with the high-priority virtual machine available. The job will be sent to one of the high priority virtual machine which meets the job requirements immediately, causing the batch job that is executing in the other virtual machine to lower its priority to benefit the interactive job.

This Glide-In mechanism will be used only in the case of not having any free CPU available to run the job, so a full testbed is needed to test it. Following there is an example of how to simulate the behaviour of a full testbed and activate the Glide-In mechanism for an interactive job.

First of all, select a site with few CPUs to avoid overloading too many resources. One apt site is `cgnode00.di.uoa.gr` which only has one CPU. Then submit as many long jobs as needed to fill this site up. In this case the job will just sleep for 10 minutes to keep the machine busy.

```
VirtualOrganisation = "cg";
Type                = "Job";
JobType             = "Normal";
Executable          = "/bin/sleep";
Arguments           = "600";
Requirements        = other.GlueCEStateStatus == "Production";
Rank                = -other.GlueCEStateEstimatedResponseTime;
```

In this case only one job is needed, and it can be submitted using the following command:

```
edg-job-submit -r cgnode00.di.uoa.gr:2119/jobmanager-pbs-long sleep.jdl
```

When the job starts running, an interactive job requiring this site can be submitted to use the high priority virtual machine created in the WorkerNode. The following jdl shows an example:

```
VirtualOrganisation = "cg";
Type                = "Job";
JobType             = "Interactive";
Executable          = "bc.sh";
InputSandbox        = {"bc.sh"};
ListenerPort        = 24100;
Requirements        = other.GlueCEUniqueID ==
                    "cgnode00.di.uoa.gr:2119/jobmanager-pbs-long";
Rank                = -other.GlueCEStateEstimatedResponseTime;
```

The bc.sh shell script is depicted here:

```
#!/bin/sh
# bc doesn't work properly if launched directly, wrap it with a little sh
echo "This is a CrossGrid Interactive calculator using bc"
echo "-----"
/usr/bin/bc -l
echo "-----"
echo "bye"
```

The job is submitted as usual, when the RB looks for machines will only find a high priority virtual machine which meets all the requirements, since there isn't any other machines with free CPUs available the job will be submitted to the Glide-In. In the next figure is shown an execution example:

```
[enol@aorbgrid jobs]$ edg-job-submit bc.jdl
```

```
Selected Virtual Organisation name (from JDL): cg
Connecting to host aorbgrid.uab.es, port 7772
Logging to host aorbgrid.uab.es, port 9002
```

```
*****
```

JOB SUBMIT OUTCOME

```
The job has been successfully submitted to the Network Server.
Use edg-job-status command to check job current status. Your job
identifier (edg_jobId) is:
```

```
- https://aorbgrid.uab.es:9000/1VcKgRU_jstD9x1A_ISPBA
```


5.3 LOG FILES

Each component of CrossGrid Resource Broker generates log files describing the actions performed. This files, as well as the log level for each, can be specified in the workload configuration file.

6 THE EDG LICENSE AGREEMENT

Copyright (c) 2005 CrossGrid. All rights reserved.

This software includes voluntary contributions made to the CrossGrid Project. For more information on CrossGrid, please see <http://www.eu-CrossGrid.org>.

Installation, use, reproduction, display, modification and redistribution of this software, with or without modification, in source and binary forms, are permitted. Any exercise of rights under this license by you or your sub-licensees is subject to the following conditions:

1. Redistributions of this software, with or without modification, must reproduce the above copyright notice and the above license statement as well as this list of conditions, in the software, the user documentation and any other materials provided with the software.

2. The user documentation, if any, included with a redistribution, must include the following notice:
“This product includes software developed by the CrossGrid Project (<http://www.eu-CrossGrid.org>).”

Alternatively, if that is where third-party acknowledgments normally appear, this acknowledgment must be reproduced in the software itself.

3. The names “CrossGrid” and “CG” may not be used to endorse or promote software, or products derived therefrom, except with prior written permission by cgooffice@cyfronet.krakow.pl.

4. You are under no obligation to provide anyone with any bug fixes, patches, upgrades or other modifications, enhancements or derivatives of the features, functionality or performance of this software that you may develop. However, if you publish or distribute your modifications, enhancements or derivative works without contemporaneously requiring users to enter into a separate written license agreement, then you are deemed to have granted participants in the CrossGrid Project a worldwide, non-exclusive, royalty-free, perpetual license to install, use, reproduce, display, modify, redistribute and sub-license your modifications, enhancements or derivative works, whether in binary or source code form, under the license conditions stated in this list of conditions.

5. DISCLAIMER

THIS SOFTWARE IS PROVIDED BY THE CROSSGRID PROJECT AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, OF SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE OR USE ARE DISCLAIMED. THE CROSSGRID PROJECT AND CONTRIBUTORS MAKE NO REPRESENTATION THAT THE SOFTWARE, MODIFICATIONS, ENHANCEMENTS OR DERIVATIVE WORKS THEREOF, WILL NOT INFRINGE ANY PATENT, COPYRIGHT, TRADE SECRET OR OTHER PROPRIETARY RIGHT.

6. LIMITATION OF LIABILITY

THE CROSSGRID PROJECT AND CONTRIBUTORS SHALL HAVE NO LIABILITY TO LICENSEE OR OTHER PERSONS FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, OR PUNITIVE DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF USE, DATA OR PROFITS, OR BUSINESS INTERRUPTION, HOWEVER CAUSED AND ON ANY THEORY OF CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7 BIBLIOGRAPHY

[1] DATAGRID WP1 – WMS Software administrator and User Guide (DataGrid-01-TEN-0118-1_1.doc) download at <http://server11.infn.it/workload-grid/documents.html>

[2] The EU-CrossGrid Approach for Grid Application Scheduling (Elisa Heymann, Álvaro Fernandez, Miquel A. Senar, Jose Salt). To be published at post-proceedings in the Springer Lecture Notes in Computer Science. Presented at Across Grids Conference and available at <http://ific.uv.es/grid/CROSSGRID-IFIC/ACrossGrid-scheduling-final.pdf>

[3] Crossgrid WP3.2 Monitoring Tools Integration. Internal document
http://savannah.fzk.de/distribution/crossgrid/crossgrid/wp3/wp3_2-scheduling/docs/CG3.2-v1.2-MonitoringPlugin.pdf