Kepler Workflow on GARUDA Grid

User Guide

ver 1.0

June 2010

Submitted to

Zeus Numerix Pvt. Ltd.

&

EADS (Bangalore, India)

CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING

C-DAC Knowledge Park,

No. 1, Old Madras Road, Byappanahalli,

Bangalore – 560 038.

Tel : +91-80-2534 1909

Fax: + 91-80-2524 7724

www.cdacb.in

Table of Contents

Chapter 1 - Introduction

1.0 Kepler

Kepler is free and open source, scientific workflow designed to help scientists, analysts, and computer programmers create, execute, and share models and analyses across a broad range of scientific and engineering disciplines. The Kepler software is developed and maintained by the cross-project Kepler collaboration, which is led by a team consisting of several key institutions that originated the project: UC Davis, UC Santa Barbara, and UC San Diego. Kepler is a java-based application that is maintained for the Windows, OSX, and Linux operating systems. The Kepler Project supports the official code-base for Kepler development, as well as provides user manuals, mechanism to report bugs, suggest enhancements, discussion forum, etc.

Kepler allows the user to create a workflow and execute the same from GUI. Once created a workflow can also be run from command line as it allows saving a workflow in xml or kar format and executed via command line.

1.1 Kepler Features

- Kepler provides a graphical user interface and a run-time engine that can execute workflows either from within the graphical interface or from a command line.
- Kepler uses a director/actor metaphor to visually represent the various components of a workflow. A director controls (or directs) the execution of a workflow. The actors take their execution instructions from the director. In other words, actors specify *what* processing occurs while the director specifies *when* it occurs.
- Actors are ready-to-use processing components that can be easily customized, connected and then run to perform complex applications efficiently. Kepler has a searchable library containing nearly 350 actors.
- Kepler workflows can be nested, allowing complex tasks to be composed from simpler components.
- Kepler workflows can leverage the computational power of grid technologies (e.g., Globus, SRB, etc). It supports job submission actor for Globus, MyProxy actor for authentication, and GridFTP actor for file transfer.
- Kepler workflows and customized components can be saved, reused, and shared with colleagues using the **K**epler **AR**chive format (KAR).

1.2 GARUDA Grid

GARUDA is a nation wide grid spread across 17 cities of the country, comprising of collaboration of science researchers, computational nodes, mass storage and scientific instruments, with an aim to provide the technological advances required to enable data and compute intensive science for the 21st century.

GARUDA is based on the Globus Toolkit 4.0.7. The Globus Alliance has adopted the Open Grid Standards Infrastructure (OGSI) standards based on convergence of Web

Services and Grid Computing technology. The various technology components of GARUDA include a Portal for access and job submission, program development tools, scheduler, reservation manger, and storage resource manager, and tools for monitoring and management of the system. Access to GARUDA is through certificates obtained from the Indian Grid Certification Authority (IGCA).

The GARUDA grid has a grid head node to which various compute resources are connected. The cluster under consideration for this project is composed of - Xeon Quad Core Dual CPU Processor X5460 nodes, having 16GB RAM, connected by Infiniband with RHEL 5.1 on Rocks v5.0. The GARUDA grid can be accessed from Internet through the gateway machine.

1.3 Kepler on GARUDA Grid

Kepler development version (2.0) has been identified based on discussion with the Kepler Forum, as it provides Job Submission Actors for Globus. The Kepler source code was compiled on the grid head node (referred to as Gridfs – IP as 203.200.36.236). The Kepler actor packages are deployed on Gridfs at /usr/local/EADS_AAOW.

The job submission actor requires inputs:

- Job script content - which is an RSL script,
- Cluster name,
- Globus path, and
- Scheduler type.

The executable name needs to be mentioned in the rsl script tag as **`<executable>executablename</executable>.`**

For Kepler to be visible from remote machine, a visual sharing tool like XManager needs to be used. The X11 on gridfs machine has been enabled for desktop sharing.

1.4 Pre-requisite software

- JAVA ver 1.6
- ANT ver 1.8
- Xmanager 2.0 or greater on client machine

Chapter 2 - Usage Instructions

2.0 Procedure for Access to Kepler on GARUDA

1. Kepler is installed at grid headnode (gridfs - 203.200.36.236) at /usr/local/EADS_AAOW

   – **`ssh 203.200.36.236 -l username`**

2. For remotely accessing the Kepler GUI following are the steps:

   a) Invoke the Xmanager client on local desktop (Xstart)

   b) In the Xstart dialog box, the Execution Command field is used to specify whether to open complete remote desktop or only the remote terminal. Hence, Execution Command can have:

   – **`Execution Command = /usr/bin/gnome-session --display $DISPLAY`**

   – **`Execution Command = xterm`**

   c) Fill out the Xstart dialog box and **`Run`**

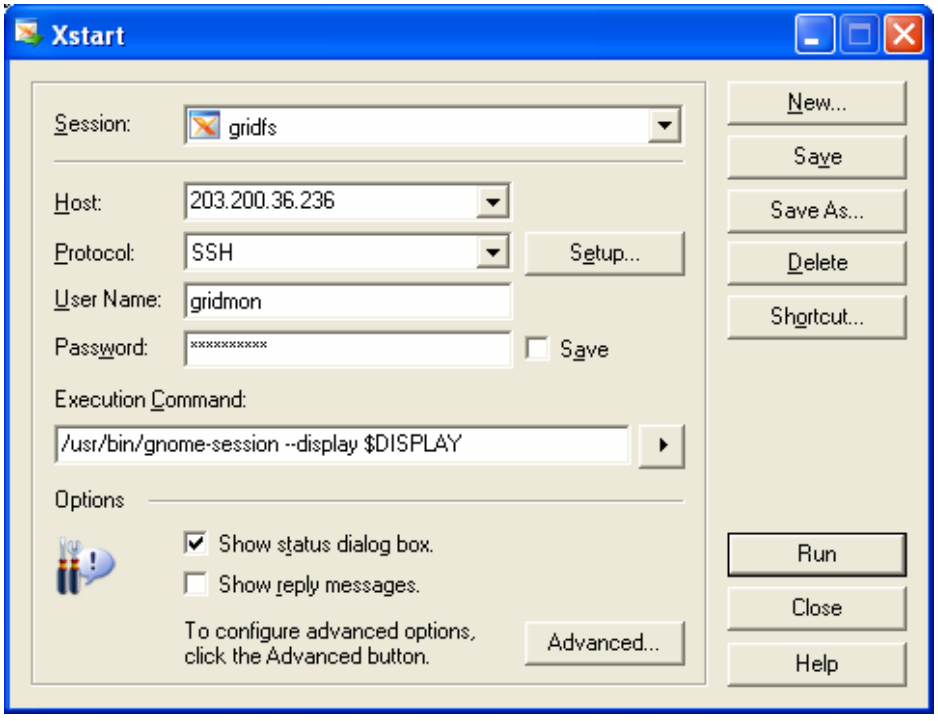


Figure 1: A sample Xstart Dialog Box

d) Set the Environment variable and Path as below:

- `export JAVA_HOME=/usr/local/jdk1.6.0_10`
- `export ANT_HOME=/usr/local/EADS_AAOW/ant/apache-ant-1.8.0`
- `export PATH=$PATH:$ANT_HOME/bin`

e) Invoke Kepler through

```
$./kepler.sh
```

2.1 Procedure for Job Submission on GARUDA

Step 1: Reserving the resources

When a person needs to submit a job to GARUDA, he should first identify the resource. RESERVATION of the GARUDA resources needs to be performed. The Reservation ID obtained should be saved to be mentioned in the RSL file or Job script content.

Reservation command:

```
$ garuda_compute_reserve gg-blr.tfg "2010-04-20T11:30:00" "2010-04-21T18:58:28" 2
```

Output of Reservation Command:

```
$ R714.gg-blr.tfg
```

Step 2: Check the credentials

Submitting a job requires a valid proxy which is nothing but a short lived credential. The user's credentials should be alive otherwise it needs to be invoked.

Proxy information can be obtained with the help of "grid-proxy-info" command

```
$ grid-proxy-info
subject  : /DC=IN/DC=GARUDAINDIA/O=C-DAC/OU=CTSF/CN=Karuna
(karunap@cdacb.ernet.in)/CN=237891045/CN=2012224773/CN=744875709
issuer   : /DC=IN/DC=GARUDAINDIA/O=C-DAC/OU=CTSF/CN=Karuna
(karunap@cdacb.ernet.in)/CN=237891045/CN=2012224773
identity : /DC=IN/DC=GARUDAINDIA/O=C-DAC/OU=CTSF/CN=Karuna
(karunap@cdacb.ernet.in)
type     : Proxy draft (pre-RFC) compliant impersonation proxy
strength : 1024 bits
path     : /tmp/x509up_u502
timeleft : 0:00:00
```

Proxy can be created by the "grid-proxy-init" command

```
$ grid-proxy-init
Your identity: /DC=IN/DC=GARUDAINDIA/O=C-DAC/OU=CTSF/CN=Karuna
(karunap@cdacb.ernet.in)
Enter GRID pass phrase for this identity:
Creating proxy ........................................
Done
Your proxy is valid until: Tue Jun  8 02:38:47 2010
The job will get a limited proxy
```

Step 3: Upload credentials to MyProxy

Now user has valid credentials to submit a job in GARUDA grid. Certificates should be available in My-proxy server.

```
$ myproxy-init
Your identity: /DC=IN/DC=GARUDAINDIA/O=C-DAC/OU=CTSF/CN=Karuna
(karunap@cdacb.ernet.in)
Enter GRID pass phrase for this identity:
Creating proxy ........................................
Done
Proxy Verify OK
Your proxy is valid until: Mon Jun 14 14:40:18 2010
Enter MyProxy pass phrase:
Verifying - Enter MyProxy pass phrase:
A proxy valid for 168 hours (7.0 days) for user gridmon now
exists on hyd01.
```

For retrieving the certificates.

```
$ myproxy-logon -s hyd01
Enter MyProxy pass phrase:
A credential has been received for user gridmon in
/tmp/x509up_u502.
```

For complete usage instructions of MyProxy please refer Appendix 4.

Step 4: Create Job Description

The job description file has to be input into the Globus job submission actor. Globus uses a Resource Specification Language (RSL) script to describe the executable, input, output and other parameters of the job.

An Example RSL file is shown below:

```
<job>
      <executable>/bin/echo</executable>
      <argument>Welcome to C-DAC</argument>
      <environment>
            <name>LD_LIBRARY_PATH</name>
            <value>/home/soademo/lib</value>
      </environment>
      <queue>batch</queue>
</job>
```

For complete syntax of the Resource Specification Language (RSL) Script file to be used on Globus based GARUDA grid please refer Appendix 3.

For each job submission actor in the workflow the corresponding job description file can be obtained by tweaking the sample RSL above. The RSL has to be given as input in the '**`Job Script Content`**' of the Kepler **`GlobusWSJob`** actor.

Step 5: Create the workflow using Kepler actors

Kepler uses the GlobusWSJob actor for submitting jobs to Globus.

GlobusWSJob actor parameters are (as shown in figure)

1 . Job Script Content

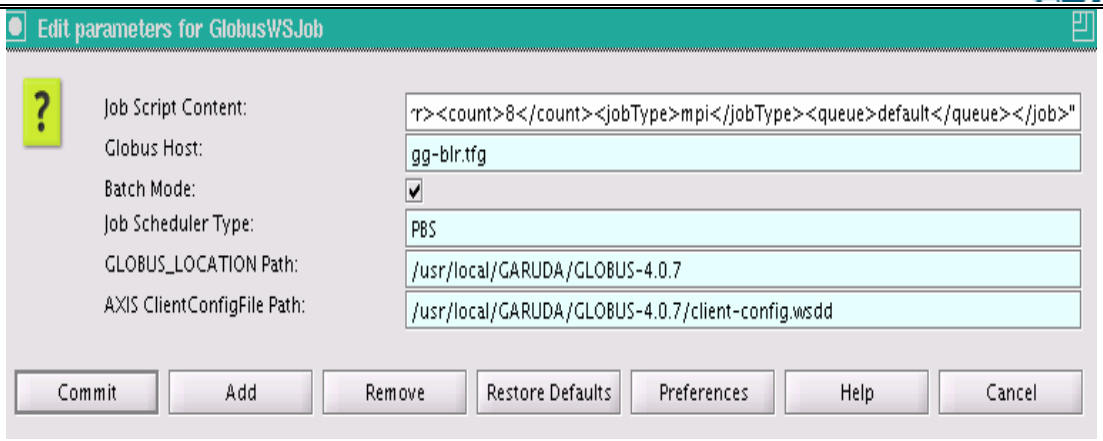
2. Globus Host

3. Batch Mode

4. Job Scheduler Type

5. GLOBUS_LOCATION Path

6. AXISClientConfigFilePath



1. **Job Script Content**: This parameter gives the details of the job to be executed by the particular actor. The complete RSL file can be copy-paste into this field. Please ensure that no extra white spaces are included inadvertently.

2. **Globus Host**: This parameter indicates which machine (cluster) in which the job should execute. As it is desired to run on the GG-BLR cluster of Garuda grid, please mention 'gg-blr.tfg' in this field.

3. **Batch Mode**: The default is taken as 'batch' job.

4. **Job Scheduler Type**: Job scheduling on the GG-BLR cluster is accomplished by the Portable Batch System (PBS). The appropriate job scheduler should be given in this parameter.

5. **GLOBUS_LOCATION Path**: The absolute path on the cluster (GG-BLR in this case) where Globus is installed.

6. **AXISClientConfigFilePath**: The absolute path of the client-config.wsdd in the GLOBUS_LOCATION

Step 6: Executing the Workflow

Once the workflow is created it can be run from the GUI, click on the **`play(|>)`** button from the toolbar.

Also the workflow can be run through the command line also with the following command:

```
ant run-workflow-no-gui -Dworkflow=<saved workflow name>
E.g.
ant run-workflow-no-gui -Dworkflow=<sample.xml>
ant run-workflow-no-gui -Dworkflow=<myworkflow.kar>
```

Note: A workflow can be saved from the File->Save menu in .xml or .kar

Chapter 3 - Testing

Exhaustive testing has been done with the Kepler deployed on GARUDA. Workflows containing combination of both serial and parallel (MPI) jobs were tried out. Also Kepler was tested for access and job submission from different user logins.

Testing of the Kepler tool has been carried out with respect to the following:

- Job submission Actor of Kepler for both sequential and parallel job submission
- MPI and Pthread jobs
- RSL with Reservation ID
- GARUDA certificates, MyProxy
- Small workflow submission
- Both GUI and command line
- Kepler toolbars through remote visualization
- Access via Internet

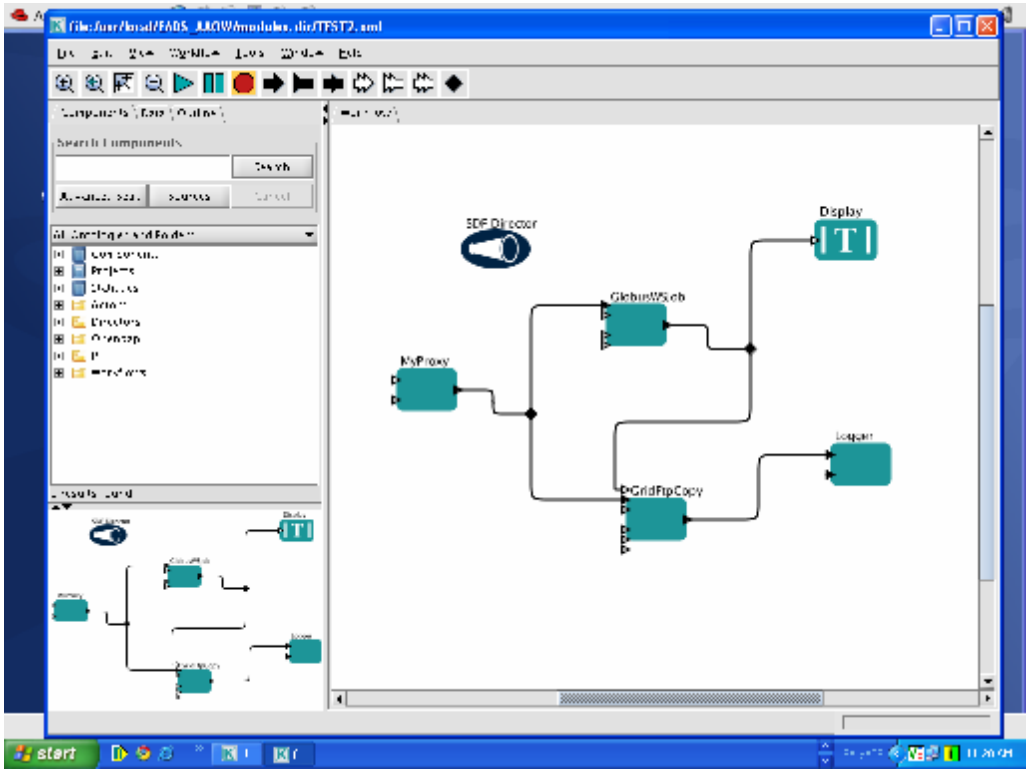


Figure 1: A sample workflow using SDF director using GloubsWSJob and GridFtpCopy actor.*

* SDF – Synchronous Dataflow (Refer Chapter 4 FAQ 5)

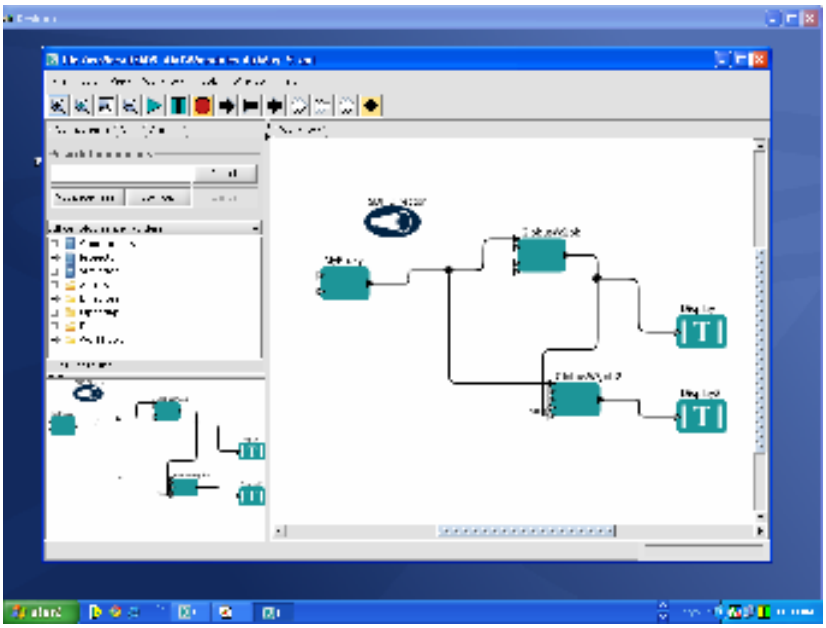


Figure 2: Submitting of jobs through GUI

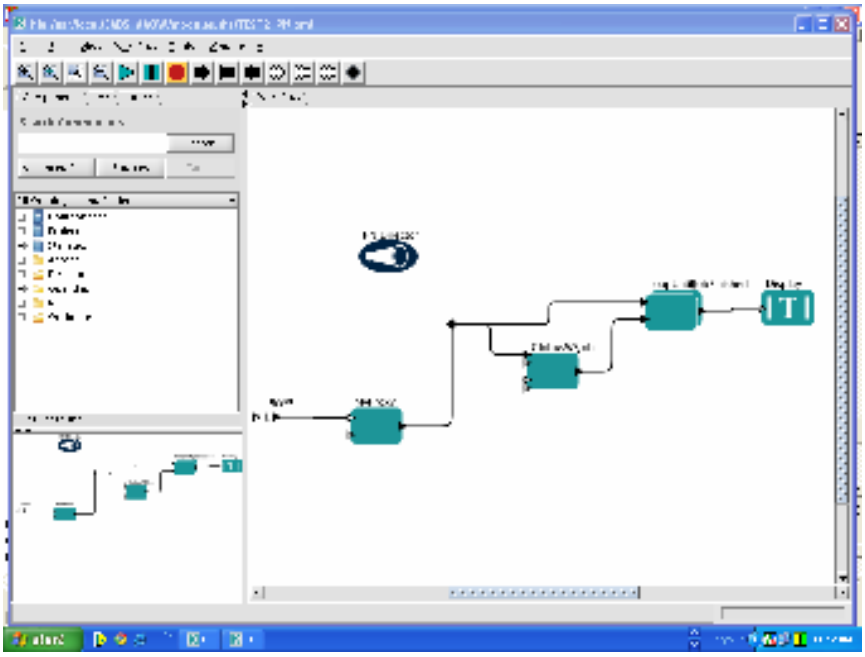


Figure 3: Submitting of jobs using PN director.*

* PN – Process Network (Refer Chapter 4 FAQ 5)

Chapter 4 - Frequently Asked Questions

1. **How can I start kepler?**

   Login to gateway m/c through ssh and to gridfs m/s

   $ cd /usr/local/EADS_AAOW/

   $./kepler.sh

   Before this pls check if the JAVA_HOME is set to usr/local/jdk1.6.0_1

   And ANT_HOME is set to =/usr/local/EADS_AAOW/ant/apache-ant-1.8.0

   And path set as $ANT_HOME/bin

2. **How can I invoke the grid actors.**

   Since the globus4.0.7 is not present in the kepler repository, it has to be manually instantiated onto the workflow canvas.

   To instantiate a component: on the menu bar

   Tools ->Instantiate component->Type in the correct class name of the component to be instantiated

   For instance some of the globus actors and their classnames

Actor	classname
GlobusWSJob	org.kepler.actor.globus.wsgram.GlobusWSJob
MyProxy	org.kepler.actor.MyProxy
GridFtpCopy	org.kepler.actor.globus.GridFtpCopy

3. **How to connect the actors**

   Once the actors have been instantiated on the workflow canvas, each actor has to be connected, which can be done by clicking on of the port and dragging it until the other actor's port.

4. **How can an actor be configured?**

   Each actor has to be configured, which can be done either by double-clicking the actor or by right-click->configure actor. Enter the appropriate values and commit the entries.

5. **What is a director?**

   Kepler uses a director/actor metaphor to visually represent the various components of a workflow. A director controls (or directs) the execution of a workflow, just as a film director oversees a cast and crew. The actors take their execution instructions from the

director. In other words, actors specify what processing occurs while the director specifies when it occurs.

Each workflow must have a director that controls the execution of the workflow using a particular model of computation. For example, workflow execution can be synchronous, with processing occurring one component at a time in a pre-calculated sequence (SDF Director). Alternatively, workflow components can execute in parallel, with one or more components running simultaneously (which might be the case with a PN Director). For more details refer Kepler User Manual (Appendix 1).

6. How to run a workflow?

To run the workflow, click on the play(||>) button from the toolbar. Otherwise the workflow can be run through the command line also with the following command

```
ant run-workflow-no-gui -Dworkflow=Sample.xml
```

7. Can I run the workflow from command line?

With the following comman, gui will not be invoked and you can run workflow **ant run-workflow-no-gui -Dworkflow=Sample.xml**

8. How to make an RSL file?

A sample rsl file is listed:

```
<job>
<executable>testgg</executable>
<environment>
<name>GARUDA_RESV_ID</name>
<value>R714.gg-blr.tfg</value
</environment>
<stdout>/home/tools/gridmon/kepler/stdout/testgg23.out.wrapper<
/stdout>
<stderr>/home/tools/gridmon/kepler/stdout/testgg23.err.wrapper<
/stderr>
<count>8</count>
<jobType>mpi</jobType>
<queue>default</queue>
</job>
```

9. What is the command to obtain the reservation ID.

garuda_compute_reserve gg-blr.tfg "2010-04-20T11:30:00" "2010-04-21T18:58:28" 2

Refer Appendix 2

10. What are the parameters of the GlobusWSjob actor?

GlobusWSJob actor parameters are

1 . Job Script Content

2. Globus Host

3. Batch Mode

4. Job Scheduler Type

5. GLOBUS_LOCATION Path

6. AXISClientConfigFilePath="/usr/local/GARUDA/GLOBUS-4.0.7/client-config.wsdd"

11. Can workflows be saved and run at later point of time?

It can be saved and run as ant run-workflow-no-gui -Dworkflow=Test.xml

Appendix 1 – Detailed Kepler User Manual

The project website (http://www.kepler-project.org/) contains the complete source code and documentation for Kepler Workflow software.

The Kepler User manual is available at: https://kepler-project.org/users/documentation

Appendix 2 – GARUDA Compute Reservation Commands

GARUDA grid compute reservations commands

The commands related to grid compute node reservation are located at

`/opt/garudaresv/bin` directory of gridfs.ctsf.cdac.org.in machine.

Command for getting the free resources in a particular time slot.

`/opt/garudaresv/bin/garuda_freeresources` can be used to find the free compute nodes available in a particular time slot. This command will return the cluster FQDN with number of free cpus.

Usage: garuda_freeresources "<start_time>" "<end_time>"

Example: garuda_freeresources "2008-08-05T14:38:28" "2008-08-05T14:58:28"

The date and time that appears as single digit must be made to double digit by prepending '0'. For example, August 5, 2008 must be specified as 2008-08-05 and not 2008-8-5. Same is applicable to time also.

Making a compute reservation

The command /opt/garudaresv/bin/garuda_compute_reserve can be used to create a compute reservation. This command will return the grid reservation id for the corresponding reservation slot.

Usage: garuda_compute_reserve <Cluster_FQDN> "<start_time>" "<end_time>" <number_of_CPU>

Example: garuda_compute_reserve gg-hyd.cdac.org.in "2008-08-05T14:38:28" "2008-08-05T14:58:28" 1

Job submission with reservation id

The command /opt/garudaresv/bin/garuda_job_submit can be used to submit jobs in a cluster corresponding to a compute node reservation. The command must be used to submit job only when the corresponding reservation slot is running or active.

Usage: **`garuda_job_submit <garuda_resv_id> <gridway_job_template>`**

Example: **`garuda_job_submit 99.gg-hyd.cdac.org.in /tmp/test.jt`**

Modifying a grid compute reservation

/opt/garudaresv/bin/garuda_compute_reserve_modify command can be used to modify an unexpired compute node reservation.

Usage: garuda_compute_reserve_modify <garuda_resv_id> <number_of_nodes>

Example: garuda_compute_reserve_modify 99.gg-hyd.cdac.org.in 1

Appendix 3 - Resource Specification Language (RSL)

Common notation for exchange of information between components. RSL provides

- Resource requirements: Machine type, number of nodes, memory, etc.
- Job configuration: executable, path, arguments,environment, etc.

Important RSL tags

Argument	count
Directory	environment
Executable	job
jobType	stdin
stdout	stderr
queue	

Simple RSL example

```
<job>
      <executable>/bin/echo</executable>
      <argument>Welcome to C-DAC</argument>
      <environment>
            <name>LD_LIBRARY_PATH</name>
            <value>/home/soademo/lib</value>
      </environment>
      <queue>batch</queue>
</job>
```

RSL (MPI Job)

```
<job>
      <executable>/tmp/hello</executable>
      <argument>C-DAC</argument>
      <directory>/home/soademo/tools</directory>
      <stdin>/dev/null</stdin>
      <stdout>stdout.${GLOBUS_USER_NAME} </stdout>
      <stderr>stderr.execution</stderr>
      <count>2</count>
      <jobType>mpi</jobType>
</job>
```

For complete syntax of the Resource Specification Language (RSL) refer http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/gram_job_description.html

Appendix 4 - MyProxy

MyProxy Usage Instructions

MyProxy is a open source software for managing X.509 Public Key Infrastructure (PKI) security credentials (certificates and private keys). It combines an online credential repository with an online certificate authority. It allows users to securely obtain credentials when and where needed. The advantage of using MyProxy is that the user need not keep their Certificates on Grid; instead they can upload and get certificate from the secured Myproxy Server whenever required.

Storing a credential in the MyProxy repository

Rather than storing your X.509 credentials (certificate and private key) on each machine you use, you can store them in a MyProxy repository and retrieve a proxy credential from the MyProxy repository when needed.

To store a credential in the MyProxy repository, run the myproxy-init command on a computer where your Grid credentials are located. For example:

```
$ myproxy-info
username: gridmon
owner: /DC=IN/DC=GARUDAINDIA/O=C-DAC/OU=CTSF/CN=Karuna
(karunap@cdacb.ernet.in)
timeleft: 0:00:00


$ myproxy-init
Your identity: /DC=IN/DC=GARUDAINDIA/O=C-DAC/OU=CTSF/CN=Karuna
(karunap@cdacb.ernet.in)
Enter GRID pass phrase for this identity:
Creating proxy .....................................Done
Proxy Verify OK
Your proxy is valid until: Mon Jun 14 14:40:18 2010
Enter MyProxy pass phrase:
Verifying - Enter MyProxy pass phrase:
A proxy valid for 168 hours (7.0 days) for user gridmon now
exists on hyd01.
```

The myproxy-init command prompts first for the pass phrase of your private key (similar to grid-proxy-init) and then prompts twice for a new pass phrase to use to secure the credentials on the MyProxy server. By default, the credential is stored under your Unix username for 7 days and can be used to retrieve credentials with 12 hour lifetimes.

Retrieving a credential from the MyProxy repository

Once you've stored a credential in the MyProxy repository, you can retrieve a proxy credential whenever you need one with the **myproxy-logon** command. For example:

```
$ myproxy-logon -s hyd01
Enter MyProxy pass phrase:
A credential has been received for user gridmon in
/tmp/x509up_u502.
```

The **myproxy-logon** command prompts for the pass phrase you set previously with **myproxy-init**, retrieves a proxy credential for you, and stores it in the correct default location for use with other Globus Toolkit programs.

The detailed list of MyProxy Command reference can be found at http://www.globus.org/toolkit/docs/4.0/security/myproxy/Cred_Mgmt_MyProxy_Interface_Commandline_Frag.html or MyProxy Information