AstroGrid-D/Nbody6++ Tutorial

Thomas Bruesemeister
tbruese@ari.uni-heidelberg.de

AstroGrid-D/Nbody6++ Tutorial

by Thomas Bruesemeister

Published September, 2007

Table of Contents

List of Tables

Chapter 1. First steps in AstroGrid-D

User authentication

The authenitication mechanism in AstroGrid-D is based on the Grid Security Infrastructure (GSI) which uses asymmetric (public key) cryptography as the basis for its functionality. Every user and service on the Grid is identified via a certificate, which contains information vital to identifying and authenticating the user or service. The signed X.509 certificate you got from your Certification Authority (CA) is usually not used to interact with grid services directly. Instead you use a so called proxy which consists of a new certificate and a private key. The proxy certificate is signed using your certificate. To create a proxy certificate use the following command:

```
grid-proxy-init
```

The created proxy certificate has a limited lifetime (12h by default) and can be adjusted using the -valid <h:m> option. *Make sure your proxy will not expire before your job has finished!* To check if and how long your proxy is valid you can use

```
grid-proxy-info
```

Both commands expect that you have your grid certificate/private key stored in $HOME/.globus/usercert.pem and $HOME/.globus/userkey.pem

Job submission

The Web-Service based version of the Globus Resource Allocation Manager (WS-GRAM) is the basic job submission facility used in AstroGrid-D. The standard tool to utilize this service is *globusrun-ws* which is part of the Globus Toolkit. The commandline reference for this client can be found on the Globus website: http://www.globus.org/toolkit/docs/4.0/execution/wsgram/rn01re01.html

A list of hostnames where jobs can be submitted can be retrieved from the WebMDS Indexes:

- AstroGrid-D MDS Overview (http://mintaka.aip.de:8080/webmds/webmds?info=indexinfo&xsl=gmondetailxsl) (all AstroGrid-D members should have access to these resources)
- D-Grid Computing Elements (http://webmds.lrz-muenchen.de:8080/webmds/webmds?info=dgridinfo&xsl=ceavailxsl)
- D-Grid hosts (static information) (https://dispatch.fz-juelich.de:8814/dgrid_ressourceall_list/back=%2fDGRIDPROVP)

The simplest form of job submission is executing a command on the execution host. With the -c option a job description will be generated assuming the first argument is the executable and the remaining are arguments. For example

```
globusrun-ws –submit –F hostname –c /bin/uname –a
```

Warning: Every option after -c is treated as arguments for the command, that is, -c must be the last option passed to globusrun-ws.

To get the output of the command on your screen you use the -s option to stream the output back to the submission host:

```
globusrun-ws -submit -F hostname -s -c /bin/date
```

Another possiblity is to save the std. output (option -so) and std. error (option -se) in a file. In this case the files must be transferred back to the submission host using globus-url-copy.

```
globusrun-ws -submit -F hostname -so job.out -se job.err -c /bin/date
```

A more convinient way is to transfer files in this case is to

Submitting a job with the job description (RSL)

Here is an example of a simple job description

```
<job>
  <executable>/bin/echo</executable>
  <argument>this is an example_string </argument>
  <argument>Globus was here</argument>
  <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
  <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
</job>
```

Tell globusrun-ws to read the job description from a file, using the -f option:

```
globusrun-ws -submit -F host -Ft PBS -f super_simple.xml
```

Specifying file staging in the job description

In order to do the file staging, one must add specific elements to the job description and delegate credentials appropriately.

```
<fileStageIn>
  <transfer>
    <sourceUrl>gsiftp://job.submitting.host:2811/tmp/mySourceFile</sourceUrl>
    <destinationUrl>file:///${GLOBUS_USER_HOME}/my_transfered_file</destinationUrl>
  </transfer>
</fileStageIn>
```

Job managers

Access to computing clusters is usually controlled by batch systems (sometimes also referred to as Local Resource Management Systems (LRMS) or Queuing Systems). The task of creating a batch skrip and submitting it to the batch server is done by Globus job managers. The user just has to pass the correct name for the job manager to the globusrun-ws command via the -Ft ("FactoryType") option. In the following example the job is submitted to the default queue on a PBS host.

```
globusrun-ws -submit -F host -Ft PBS -c /bin/date
```

Data Transfer

Data tranfer in AstroGrid-D is handled through GridFTP which adds GSI facilities to the plain old FTP protocol. The basic URL based GridFTP Client "globus-url-copy" which is part of the Globus can do multi protocol data movement. It supports gsiftp://, ftp://, http://, https:// and file:// protocol specifiers in the URL.

- Getting files:

  ```
  $ globus-url-copy gsiftp://hostname/path/to/remote/file file:///path/to/local/file
  ```

- Putting files:

  ```
  $ globus-url-copy file:///path/to/my/file gsiftp://hostname/path/to/remote/file
  ```

- Third party transfers:

  ```
  $ globus-url-copy gsiftp://remotemachine/path/file gsiftp://othermachine/path/file
  ```

Figure 1-1. Performance optimization using parallel streams.

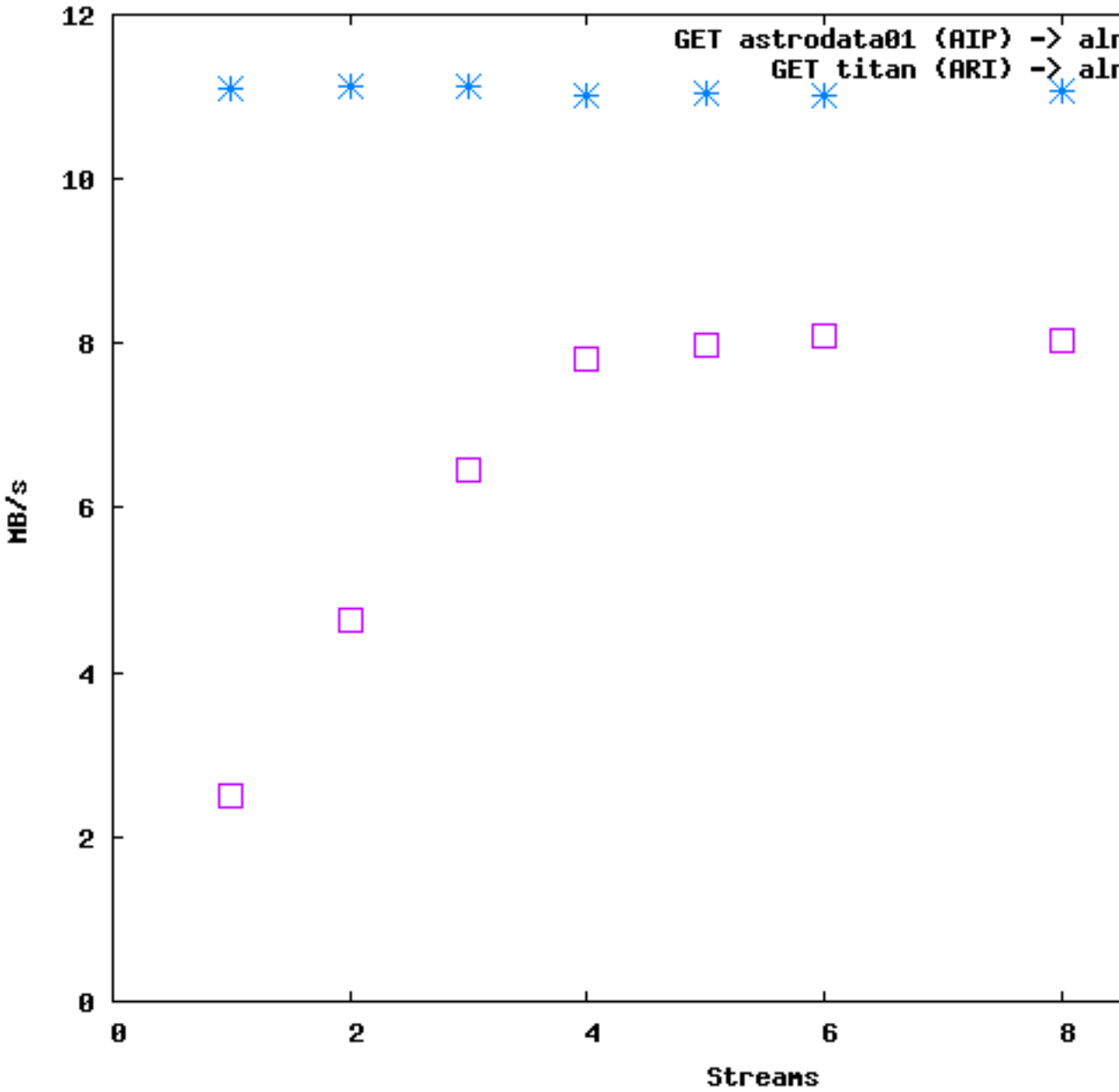

Table 1-1. Useful globus-url-copy options.

-p n	Number of parallel streams (4 parallel streams are recommended for WAN data transfers)

-r	Copies files recursively
-dbg	Debugs FTP connections and prints the entire control channel protocol exchange to STDERR
-f	Reads list of URL pairs from a filename

Chapter 2. NBODY6++ Deployment

Overview

For NBODY6++ there is a deployment package available which contains scripts to submit, monitor and kill NBODY6++ jobs. It also provides automatic data-staging, that is, the NBODY6++ input and output files are automatically transferred back and forth. The NBODY6++ source code which is not part of the submission package must be separately downloaded (usually via SVN).

Note: The deployment package requires a NBODY6++ version which uses the GNU autotools (autoconf, automake) for the build process. Currently only the development version in the trunk of the NBODY Subversion repository supports this.

The AstroGrid-D deployment scripts for NBODY6++ can be downloaded from the AstroGrid-D software repository. The current (stable) branch is 0.2.x.

```
svn co http://svn.ari.uni-heidelberg.de/repos/nbody/deployment/branches/0.2.x nb6deployment
```

The scripts expect the Nbody6++ source code in the 'nbody6src' subdirectory of the deployment package. If the directory does not exist and the user tries to submit a job the script will automatically ask if it should fetch the sources from the SVN server. This step can also be done manually:

Manual download:

```
svn co http://svn.ari.uni-heidelberg.de/repos/nbody/nbody6/trunk nbody6src
```

Submitting jobs

Jobs can be submitted using the submit.sh script. To put it simply this script packetizes the source code using 'make dist', generates the RSL job description from a template and submits the job to a Grid node. The information how to invoke the script can be obtained by starting the script without any parameters:

```
$ ./submit.sh
Submits Nbody6++ jobs to Globus nodes.
Usage: ./submit.sh [options] <parameter-file>

Options:
  -d                  Delegate full credential.
  -g host             Submits the job to <host> [hydra.ari.uni-heidelberg.de]
  -h                  Print this help.
  -n                  Disable batch mode.
  -m                  Enable MPI. (Experimental).
  -s                  Enable job statistics (Experimental).
  -t job-manager      Use <job-manager> as Globus Job Manager. [GW]

Nbody6++ deployment package for AstroGrid-D, v0.1.1 ($Revision: 2964 $)
```

The only mandatory parameter for the submit.sh script is the NBODY6++ parameter input file. This mandatory parameter must be a relative path to a file in a subdirectory of the deployment package. To submit a job using the in5000.comment parameter input file under the var directory enter the following command:

```
./submit.sh var/in5000.comment
```

From the command-line help you can see that by default the job gets submitted to hydra using the GW job manager which means that GridWay, the resource broker, is used to select an appropriate resource for the job. Other examples for job-managers are Fork, PBS, SGE, LSF or Condor. What type of job-manager(s) are available for a specific resource can be obtained from the MDS.

The status of the jobs can be checked using the 'status.sh' script which needs to be invoked with a local job-id as the first parameter or with the -a options which lists the status of all jobs. The local job-id is printed on the screen when submitting a job and is also written to the job.log under var. After a job has finished the output files are placed in outfiles/<jobid>