
AstroGrid-D

Deliverable 1.4



Workgroup 1:
Resource Integration and Grid Support

Running MPI Jobs on Grid Resources¹

Deliverable	1.4 Running MPI Jobs on Grid Resources
Authors	Workgroup 1: Resource Integration and Grid Support
Editors	H. Enke, S. White
Date	01-06-2008
Document Version	1.0.0
Current Version	1.0.0
Previous Versions	0.1.0, 0.1.1, 0.1.2, 0.1.3

A: Status of this Document

Published Deliverable 1.4, Version 1.0.0, describing options and basic procedures for configuration and use of Globus Middleware to submit Message Passing Interface (MPI) batch jobs from the Grid.

B: Reference to project plan

In Astrophysics as in other realms of science, modelling of physical processes by simulations on computer clusters is widely used. Often MPI is employed for data communications between the processes that comprise the simulations. Enabling MPI jobs on cluster resources in the Grid is a necessary task.

¹This work is part of the AstroGrid-D project and D-Grid. The project is funded by the German Federal Ministry of Education and Research (BMBF).

C: Abstract

This document describes how to configure and use Globus to submit MPI batch jobs to clusters over the Grid. Three main points are covered:

- real benefits brought to users by Grid job submission,
- practical details of MPI job submission over the Grid,
- recommendations about several easily-remedied issues with site configuration and the Globus software are made, toward making the software convenient for users.

D: Changes History

Version	Date	Name	Brief summary
0.1.0	13.04.2008	S. White	Working Draft Creation
0.1.1	20.04.2008	S. White	Addition of subsections, more experiences
0.1.2	20.05.2008	H. Enke	Editorial work, Ref. to project plan, Ref. to D-Grid Betriebskonzept
0.1.3	20.05.2008	S. White	Editorial work, more experiences
1.0.0	01.07.2008	H. Enke	Publish accepted document

E:

Contents

1	Submission of cluster jobs from the Grid	5
1.1	Advantages of a Grid based solution	5
1.2	State of Grid cluster configuration	5
1.3	Scope of our tests	7
1.4	Tests with a real world MPI program	7
2	Basic MPI job submission from the Grid	8
2.1	Job description file	8
2.2	Submission command line and batch systems	8
2.3	Job monitoring/cancellation	9
2.4	Submission troubleshooting	9
2.4.1	General errors	9
2.4.2	Globus job submission errors	10
2.5	Building an MPI executable	10
2.5.1	Environment	10
2.5.2	Compiler for MPI executables	10
2.5.3	Testing an executable	11
3	Job submission for a real application	12
3.1	Interaction of job submission and batch systems	12
3.1.1	Where batch job processes run	12
3.1.2	Batch queues	13
3.2	Cores per node	13
3.2.1	Node selection parameters for PBS	13
3.3	Prologue/epilogue script	13
3.3.1	Per-user pro- and epilogue script	14
3.3.2	Modification of batch system adapters	14
3.4	JDD executable syntax	14
3.4.1	Example	14

3.5	File staging	15
3.5.1	Job submission with file staging	15
3.5.2	Tarballs to transfer directories	15
3.5.3	Data on separate server	15
3.5.4	Clean-up	16
3.6	Configuration problems	16
4	Recommendations	17
4.1	Site configuration	17
4.1.1	Grid services	17
4.1.2	Site Documentation	17
4.1.3	Test procedure for site setup	18
4.2	Globus software	18
4.2.1	Globus bug reports	18
A	MPI test program	19
B	Extended job description template	20
C	Sample WS-GRAM prologue script	21
D	Configuration troubleshooting	22
	References	24

1 Submission of cluster jobs from the Grid

The Grid opens up many cluster computer resources to researchers. These clusters are built for the purpose of running parallel programs by offering fast communication between processes running on the cluster nodes. Parallel programs commonly use the Message Passing Interface (MPI) for inter-process communication. There are several MPI implementations. Building and running an MPI program thus often requires specific knowledge about the MPI setup on the cluster.

A cluster's local resource management system (LRMS), handles the allocation of resources (CPU, memory, etc) to jobs. The Globus middleware offers a Grid interface to the LRMS.

In this document, we explore the basic ideas and the use of the Globus job submission facilities, common problems that arise, and procedures to determine and solve some of these problems.

Note the distinction between submitting from the Grid a cluster MPI job, and an MPI job that sends messages via the Grid. The latter is the subject of another AstroGrid-D report, *MPI over the Grid* [3].

We will not cover here *deployment* issues, such as upload of source code and automatic building executables. The techniques for these things are an extension to what we present here, but to implement them seamlessly across different compute resources on the Grid will require agreements to standards.

1.1 Advantages of a Grid based solution

The availability of large numbers of compute resources in itself raises new difficulties: the setup of submission scripts for multiple clusters would be a daunting task, were it not for the Grid batch job submission facilities of Globus.

The advantages to a researcher in using the Grid to submit jobs include:

- simpler job submission setup and commands
- easy access to remote data resources on the Grid
- simpler switching from one cluster to another

The point of Grid job submission is to hide much of the complexity of conventional batch job submission from the user.

1.2 State of Grid cluster configuration

The biggest difficulty in grid job submission is the initial configuration of the complex layers of hardware and software involved. At least three problem areas are subsequently covered:

- Job submission to a cluster
This is a rather complicated process. We discuss the basic support from the site administration needed by a user to manage it with less pain.

- Grid job submission
We discuss a couple of lacunae in the Globus job submission software, that affect certain types of practical parallel job submissions.
- Misconfiguration of Globus for the LRMS
This often goes unnoticed. We recommend testing procedures that could be used by administrators to check that their systems are properly set up.

1.3 Scope of our tests

For the basic job submission tests, four clusters were available:

- damiana.aei.mpg.de
- lxgt2.lrz-muenchen.de
- mardschana.zib.de
- hydra.ari.uni-heidelberg.de

All four clusters use the ParaStation cluster software [18] from ParTec Cluster Competence Center. This software affects the way MPI jobs are submitted to the batch system. The functionality of the `mpirun` or `mpiexec` commands of other implementations is compiled into the ParaStation MPI executable, so that to start the MPI environment, one calls the executable directly.

Half of the available machines were using the Sun Grid Engine (SGE), which from a Globus job submission perspective, is a batch system. We experienced no problems specific to it. A Grid user will see SGE in job submission only as the argument to the “-Ft” option on the job submission command line. The other half were using PBS as the local resource management system.

None of these systems was properly set up or documented for cluster job submission over the Grid when the work on this deliverable started.

In each case, interaction was required with system administrators, lasting from a few days to several weeks. Finally, a simple MPI test submitted from the Grid ran at each of these sites.

From these tests we learned a lot about common problems encountered in Globus setup. This is summarized in section 3 and appendix D.

1.4 Tests with a real world MPI program

To go beyond the simple “Hello World” program with our tests, we used a program from cosmology² for the study of dark matter halos. This program is a highly parallel MPI program, which reads large input files (four GB), and produces large output files (GB) on each run.

It was convenient to keep the data files on a single large file server, AIP’s AstroData. The Grid made it easy to access this data from a variety of machines.

Furthermore, the AEI has a large fast cluster, Damiana, which has recently become accessible to the Grid. There is a dedicated 10Gb connection between the AEI’s clusters and AIP’s AstroData file server, so it was an ideal environment to do runs on Damiana and keep data on AstroData.

From working with the program, we gathered more experience regarding job submission and the needs of real users. This is discussed in section 3.

²We thank S. Knollmann from AIP for his collaboration.

2 Basic MPI job submission from the Grid

First we will discuss the task of submitting an MPI job to a cluster that is a Grid resource.

2.1 Job description file

The job may be described by a WS-GRAM Job Description document (JDD)[9], which is an XML document³.

A JDD file "simple-mpi.xml" for describing a batch job running an MPI executable "mpihello" (see Appendix A for an example program) might look like this:

```
<job>
<executable>./mpihello</executable>
<directory>${GLOBUS_USER_HOME}/mpitest</directory>
<stdout>${GLOBUS_USER_HOME}/mpitest/grid_run.stdout</stdout>
<stderr>${GLOBUS_USER_HOME}/mpitest/grid_run.stderr</stderr>
<count>4</count>
<jobType>mpi</jobType>
</job>
```

This specifies

- which executable to run on the compute resource,
- from which directory to run the executable,
- files in which the job's standard input and output are saved,
- the number of parallel process to run,
- that the executable is to be run in an MPI environment.

2.2 Submission command line and batch systems

To submit the job via Globus is very easy; a line like this will often suffice:

```
globusrun-ws -b -submit -Ft batch -F rsrc-hostname -f simple-mpi.xml
```

where *batch* specifies the cluster's batch system, which (as of this writing) may be one of

- PBS for the Torque (OpenPBS) batch system[12]

³In future releases of Globus (4.1.1 or greater), the preferred job description language will change from JDD to JSDL. In pre-Web Services versions of Globus, the required job description language was RSL (Resource Specification Language).

- LSF for the Platform LSF batch system[13]
- Condor for Condor batch system[15]
- SGE for Sun Grid Engine “Scheduler Adapter Interface” [14]
- Loadleveler for IBM LoadLeveler batch system[11]

The flag `-b` causes the command to exit immediately after submission, returning a standard EPR (see section 2.3) which can be later used to monitor job progress, or cancel the job. This may be omitted to watch the progress real-time.

Globus Job submission is part of the “WS-GRAM” (*Web Services Grid Resource Allocation and Management*) module, for which there is a User’s Guide [8].

2.3 Job monitoring/cancellation

Batch jobs can be monitored or canceled using the job’s EPR (*End Point Reference*).

After a successful batch job submission, an EPR file is printed to standard output. This can be redirected to a file by using the “`-o`” option on the submit line.

To see the current status of the job, use a command like this, with the job’s EPR in *job.epr*:

```
globusrun-ws -status -j job.epr
```

The job can be canceled so:

```
globusrun-ws -kill -j job.epr
```

Progress has also been made toward real-time monitoring of multiple jobs. See [4].

For further information, see the `globusrun-ws` page in the WS GRAM documentation [8].

2.4 Submission troubleshooting

2.4.1 General errors

- Naming the executable file “`test`”, which is of course a shell intrinsic name. Surprisingly, this caused no problem on one system, but resulted in strange errors on another (“cannot contact local daemon”).
- Defining shell variables in the shell startup files that interfere with the environment set by an environment module. Result was an error loading shared libraries. Bash users should check especially both the files `~/.bashrc` and `~/.bash_profile` for such variables.

2.4.2 Globus job submission errors

These are problems specific to Globus job submission, that may easily happen by mistake, or unfamiliarity with Globus commands.

- The `globusrun-ws` command exits immediately with:

```
Job failed: The jobType mpi feature is not available on this resource.
```

This will usually occur when the incorrect batch system type (“factory type” in Globus parlance) was specified in the `-Ft` option of the command line.

- In job standard error:

```
Warning: Process-to-Processor binding indeterminable.  
One reason could be asymmetric binding of existing processes.
```

This occurred in a situation where the MPI executable was launched by a shell script specified as the executable in the JDD file. The particular shell was not set up to have the right MPI environment. This was solved by putting in the first line of the script a direct invocation of a shell that was configured correctly.

- The `globusrun-ws` command exits with:

```
globusrun-ws: Job failed: Staging error for RSL element fileStageIn.  
A stagingCredentialEndpoint element was not specified in the RSL,  
but is needed for staging.
```

File staging was requested in the job description file, but the “-S” flag wasn’t present on the command line.

2.5 Building an MPI executable

To build an MPI executable for the specific hardware is necessarily a site-specific task. The site’s system administrators should provide the necessary information (typically, on a web page).

Direct access to the head node of a cluster requires that the Globus service `gsissh` be available.

2.5.1 Environment

Often, clusters use such mechanisms as “module” [16] or `SoftEnv`[17] to set up an appropriate build environment. However at this time there is no standard way of naming the required environments; it must be determined by consulting site documentation. (See section 4).

2.5.2 Compiler for MPI executables

Once the proper environment has been set up, building an executable usually doesn’t take more than to compile with the proper MPI compiler wrapper:

`mpicc, mpiCC, mpif90, mpif77`

for C, C++, Fortran 90, or Fortran 77 code, respectively.

2.5.3 Testing an executable

Since Grid resources differ in so many ways, it is often helpful to do initial testing of an MPI executable directly on the cluster where it is intended to run.

How to do this depends on the cluster's batch system. It is usually possible to run the MPI job in an "interactive session". The batch system then allocates the requested number of nodes, and logs the user into one of them. From there, the MPI executable can be run, as it is otherwise run from a batch script. If no interactive mode is available, submit a job to the queue for testing, and look at the environment variables.

If strange errors persist, it may also prove useful to replace the job executable with a shell script that just prints out the environment variables, as seen by a running job.

3 Job submission for a real application

In practice, cluster job submission is complicated by further application requirements, by inhomogeneity of cluster resources on the Grid, and by some shortfalls of the Globus software. We discuss here some of these issues, and present work-around's and recommendations.

In appendix B we present a template `practical.xml` for a practical MPI job submission over the Grid. It features:

- data staging from a remote server
- unpacking of a staged-in tarball using a shell script
- a work-around for the Globus job prologue shortfall

3.1 Interaction of job submission and batch systems

A Grid-enabled cluster consists of a head node with a Globus installation and access to the LRMS.

The middleware typically offers for clusters two jobmanagers: the default Fork-jobmanager or the LRMS-jobmanager. The parallel jobs we discuss are submitted to the middleware's LRMS jobmanager.

3.1.1 Where batch job processes run

It is important to understand on what machines the job scripts and parallel processes run.

For each job, the batch system provides a list of allocated cluster nodes, and then starts one process on the first of these nodes (the *root node*). It is then the responsibility of this process to start parallel processes on the other nodes in this list.

A conventional batch job is started by submitting a batch script to the cluster's batch system. This script is run on the root node, and in turn calls something like `mpirun` to launch the parallel executable on the allocated compute nodes. The script may also do per-job housekeeping before and after the parallel processes run.

Grid job submission does not refer directly to the local batch system. This has a couple of consequences.

On many systems compute nodes have limited network connectivity, and no Globus installation. This means that remote file transfers can't generally be done from processes running on compute nodes. Thus the need for "stage-out" and "cleanup" sections in the Globus job description. These are executed on the cluster head node.

Globus job submission currently has no provision for per-job housekeeping. (The `<executable>` section of a JDD file is the program to be run in parallel on the compute nodes.) There are work-around's for this, which we discuss in subsection 3.3.

3.1.2 Batch queues

Most batch systems have named *queues*, such as "long" or "short", "big" or "small", intended to improve job throughput. These named queues are shorthand for allocation of the number of processors, job wall-time and other parameters.

Most systems feature a default queue in which a job is to be run in the absence of a queue specification.

There is no standard regarding the naming or functionality of queues, each job submission should specify a named queue that the user must determine for the cluster at hand.

The JDD format provides the <queue> tag just for this purpose.

3.2 Cores per node

The nodes of modern clusters typically have multiple processor cores that share the node's memory. In some applications, the amount of memory available to each process is more important than the number of processor cores working on the data. For these types of applications, users often specify that fewer processes than available cores are to be started on each node, in order to dedicate more of each node's RAM to each process.

Unfortunately Globus does not currently possess a machine-independent means of specifying RAM per process, but there are several options.

A feature request (see 4.2.1) has been filed with the Globus developers on this point.

3.2.1 Node selection parameters for PBS

Recent versions of Globus have introduced extensions [10] to WS-GRAM that allow a specification of cores to use on each node, but only for machines running the PBS batch system. Older Globus versions can be upgraded to support these extensions as well.

For example, this JDD directive will specify 20 nodes using 2 computing cores per node.

```
<extensions>
  <resourceAllocationGroup>
    <hostCount>20</hostCount>
    <cpusPerHost>2</cpusPerHost>
  </resourceAllocationGroup>
</extensions>
```

3.3 Prologue/epilogue script

The file passed in the JDD <executable> section is the parallel program executed <count> times on the compute nodes. However, a user will often need to do some housekeeping just once per job, such as unpack tarballs, do preprocessing, or signal other machines. It is often unacceptable for such things to be done by the parallel program binary.

One would expect the JDD specification to provide tags for a *prologue* and an *epilogue* script that are run just once before and after the parallel execution starts, for such housekeeping purposes.

Unfortunately, there is no such provision. A feature request (see 4.2.1) has been filed with the Globus developers on this point.

This is a rather bad weakness. A substantial amount of personnel time at various sites has gone into developing work-arounds for it. Some are listed below.

3.3.1 Per-user pro- and epilogue script

It is possible to specify in the JDD a script that implements this prologue functionality then launches the parallel executable. Although this adds another file, and another layer of complication to the whole job submission process, some users may find this useful. We present an example⁴ of such a script in appendix C.

3.3.2 Modification of batch system adapters

The Globus batch system adapters may be modified on each site to provide a prologue and epilogue facility⁵.

This approach requires less involvement by users and isn't hard to do, however, it requires a policy and system administrator involvement, and in the end it is site-specific.

3.4 JDD executable syntax

The use of JDD in running scripts requires some explanation.

First, the `<executable>` element which must contain precisely the path to an executable file. Arguments to the executable are specified in `<argument>` elements that follow. These should each contain just one of the tokens that are normally separated by spaces on the command line.

Furthermore, for debugging purposes one might want to redirect output as one does in a Unix shell. There are two subtleties in doing this in a JDD file. First, the shell redirect symbol “>” is an argument to the shell *not* to the executable, whereas the `<argument>` elements in the JDD file are just arguments for the executable. To do a shell redirect then, one must put the shell binary as the executable. Second, the “>” symbol is of course not permissible in a XML document like JDD. One must use the character entity “>” instead.

3.4.1 Example

These JDD lines would append the output of the `hostname` command to a file “myhosts”

⁴Jan Ploski has made another such script for Condor. See <https://bi.offis.de/wisent/tiki-index.php?page=Condor-GT4-BigJobs>

⁵Gabriel Mateescu at LRZ has shown how to do this: http://www.grid.lrz.de/en/mware/globus/download_preamble.html

```
<executable>/bin/sh</executable>  
<directory>$GLOBUS_USER_HOME</directory>  
<argument>-c</argument>  
<argument>/bin/hostname &gt;&gt; myhosts</argument>
```

Here, the second `<argument>` element (which in this case is a shell command line) is passed to the executable: the shell `/bin/sh`.

3.5 File staging

The provisions for file staging in the JDD format are useful tools to stage in scripts and executables and stage results out over the Grid.

For examples, see the sample JDD file in appendix B.

3.5.1 Job submission with file staging

To submit a job that stages data, an extra option is required on the command line, “-S”. Building from section 2.2, the full job submission line is then:

```
globusrun-ws -b -S -submit -Ft batch-sys \  
    -F rsrc-hostname -f practical.xml
```

3.5.2 Tarballs to transfer directories

The existing stage-in facilities do not include a convenient way to transfer directory trees.

To compensate, we tar up the directories containing executables and input files, stage them in, then unpack these tar files with a script.

Likewise, job output can be tarred up by the script after the parallel executable has finished, and then be staged out by the job description file.

This is complicated by the lack in current Globus job submission of a provision for per-job scripts. See subsection 3.3.

3.5.3 Data on separate server

It is often useful to keep data files on a file server that is different from the machine from which the job is launched.

In this case, multiple `<fileStageIn>` and `<fileStageOut>` sections may be used to make the distinction.

3.5.4 Clean-up

Clean-up can't be accomplished by a script in the job description executable section, because it naturally has to happen after the job output has been staged out, and stage-out can generally be accomplished only from the head node after the job executable has finished. (See subsection 3.1.1 above.) For this reason, it is done by Globus with a special directive in the job description. See the example in appendix B.

To delete directories, the directory name must end with a slash "/" character.

3.6 Configuration problems

In the process of getting the several Grid resources to accept a Grid submission of our simple MPI program and run it, we encountered and solved a number of problems. These problems are documented in appendix D.

4 Recommendations

4.1 Site configuration

The site configuration required to enable submission of MPI jobs from the Grid includes several steps.

4.1.1 Grid services

Although in principle grid jobs can be submitted with knowledge of just the machine architecture and the type of the batch system, real users will often have to log in and run jobs interactively for debugging purposes. Thus not only should the WS-GRAM module be available, but also `gssh` into the cluster should be enabled, either to the head node of the cluster or some equivalent node.

Recommendation 1: *Globus GRAM, WS-GRAM and gssh should be enabled for appropriate grid access to the cluster.*

4.1.2 Site Documentation

The minimal site information needed by Grid users of a cluster is:

- environment set-up for building and running MPI programs
 - available options (module, SoftEnv, etc)
 - procedures to put environment options into effect
 - basic information on the types of compilers, and the preferred compiler. (Likewise, in case multiple libraries, such as MPI, are present.)
- file system
 - access to mass storage (mount points)
 - storage quotas, limitations, policies
- local batch system
 - available queues (their intended use and limitations)
 - example batch scripts
 - examples of job submission
- machine information
 - architecture
 - speed
 - cores per node
 - ram per node
 - type of interconnect

Recommendation 2: *Standardized and brief documentation for each grid site covering the above items.*

4.1.3 Test procedure for site setup

Below, a simple test procedure for cluster administrators is described to check if the site setup is fully operational for MPI jobs submitted over the Grid.

We assume that the cluster and its batch system have already been configured, and that MPI is installed and configured.

- Compile the test program `mpihello.c`, as listed in appendix A.
- Submit the binary to the local batch system, and check its output.
- From a remote computer, submit the job to the cluster over the Grid, using the job description file listed in section 2.1 and the command line 2.2. Check that the job's standard output file contains correct output listed in appendix A.
- Note in public user documentation any special actions required (e.g., modules setup, special queue commands).

Recommendation 3: *Each site administrator should verify the site passes this basic tests.*

4.2 Globus software

4.2.1 Globus bug reports

The following bug reports have already been made to the Globus developers.

- *Allow a prologue/epilogue script for 'mpi' and 'multiple' jobs*
http://bugzilla.mcs.anl.gov/globus/show_bug.cgi?id=5698
- *specification of RAM per process in parallel jobs*
http://bugzilla.mcs.anl.gov/globus/show_bug.cgi?id=6072
- *JDD documentation issues*
http://bugzilla.mcs.anl.gov/globus/show_bug.cgi?id=6069

Recommendation 4: *Stress to the Globus developers the importance of these few improvements to the software, and aid them in obtaining good solutions.*

A MPI test program

The following is the C code for a rudimentary MPI program, that simply initializes the MPI “world” and prints out the basic parameters: the number of processes (the “size”) and the MPI process number (the “rank”).

See section 2 for instructions; also 4.1.3.

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <unistd.h>

int
main( int argc, char **argv )
{
    int rank, size;
    char hostname[HOST_NAME_MAX] = { '\0' };

    gethostname( hostname, sizeof( hostname ) ) ;

    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    MPI_Comm_size( MPI_COMM_WORLD, &size );

    fprintf( stdout, "MPI world of size %d inited on %s, rank %d.\n",
             size, hostname, rank );

    MPI_Finalize();
    return 0;
}
```

Below is sample output of the code, run on four nodes of the machine *astrodata* (which has numbered nodes, each with two processor cores):

```
MPI world of size 4 inited on astrodata01.gac-grid.org, rank 0.
MPI world of size 4 inited on astrodata01.gac-grid.org, rank 1.
MPI world of size 4 inited on astrodata02.gac-grid.org, rank 2.
MPI world of size 4 inited on astrodata02.gac-grid.org, rank 3.
```

Here, the output lines happen to be ordered by MPI rank, but that will not typically be the case.

B Extended job description template

This JDD job description file is meant to be used with the prologue script in appendix C. By executing the argument tagged as PRLG just once, it effects a per-job housekeeping task, which in this case un-packs a tarball that has been staged in.

It also illustrates several other techniques in Globus job submission, including file stage-in, and clean-up.

```
<job>
<executable>./prologue.pl</executable>
<directory>${GLOBUS_USER_HOME}</directory>
<argument>PRLG tar -xvf input.tar; cd rundir; ./doit.sh</argument>
<argument>EXEC cd mpitest; /bin/tcsh -c ./mpihello</argument>
<argument>EPLG echo Ha Ha all done!</argument>

<stdout>${GLOBUS_USER_HOME}/p.stdout</stdout>
<stderr>${GLOBUS_USER_HOME}/p.stderr</stderr>

<count>8</count>
<jobType>mpi</jobType>

<fileStageIn>
  <transfer>
    <sourceUrl>gsiftp://cashmere.aip.de/z/swhite/input.tar</sourceUrl>
    <destinationUrl>file:///${GLOBUS_USER_HOME}/</destinationUrl>
  </transfer>
</fileStageIn>

<fileCleanUp>
  <deletion>
    <file>file:///${GLOBUS_USER_HOME}/input.tar</file>
  </deletion>
  <deletion>
    <file>file:///${GLOBUS_USER_HOME}/rundir/</file>
  </deletion>
</fileCleanUp>
</job>
```

Here, the text marked by the strings PRLG, EXEC, and EPLG are single shell command lines, comprising the prologue, executable, and epilogue, as discussed in section 3.3.

C Sample WS-GRAM prologue script

See section 3.3.

```
#!/usr/bin/perl
use strict;

# For use with Globus job submission: runs per-job prologue (and optional
# epilogue) scripts just before (and after) the parallel process runs.
#
# Pass scripts in RSL arguments list, prefixed by keywords PRLG, EXEC, EPLG.

$_ = join( ' ', @ARGV );      # trick: args passed by WS-GRAM as big string

( /PRLG (.*)\s+EXEC (.*)\s+EPLG (.*)/ || /PRLG (.*)\s+EXEC (.*)/ )
    or die " Usage: \n $0 PRLG pro EXEC exec [EPLG epi]\n";

my ( $prologue, $parallelProg, $epilogue ) = ( $1, $2, $3 );

my $jobid = $ENV{JOB_ID};      # specific to Sun Grid Engine
my @scampi_process_param = split( ' ', $ENV{SCAMPI_PROCESS_PARAM} );
my $mpiNode = $scampi_process_param[3];      # specific to Sun Grid Engine

my $lockF = "$ENV{HOME}/MPIJOB.$jobid.lock"; # name for lock file in home dir

if( $mpiNode eq '0' ) {      # on root node, run prologue script
    my $prlgStat = system( $prologue );
    open( FH, ">$lockF" ) or die "$0 couldn't create file $lockF";
    print FH "$prlgStat\n";      # save prologue status in lock file
    close FH;
}
else {      # on other nodes, snooze until see lock file
    while( ! -f $lockF ) { sleep( 1 ); }
}

# on all nodes, check that prologue succeeded
open( FH, "<$lockF" ) or die "$0 couldn't open file $lockF";
my $prlgStat = <FH>;
( $prlgStat == 0 ) or die "prologue error " . ( $prlgStat >> 8 ) . "\n";

system( $parallelProg );      # run the parallel command

if( $mpiNode eq '0' ) {      # on root node:
    ( $epilogue ) and system( $epilogue );      # run epilogue, if specified
    unlink( $lockF );      # delete the lock file
}
```

D Configuration troubleshooting

The following are configuration problems encountered on various systems. In some cases, we quote explanations given by system administrators.

- (At LRZ) Which machine to use? Documentation, although copious, is not easy to follow.

- `globusrun-ws` fails immediately, printing

```
Error submitting job
globus_xio_gsi: gss_init_sec_context failed.
GSS Major Status: Unexpected Gatekeeper or Service Name
```

(Maybe LRZ-specific) “globus cant handle alias names (cnames). But its also a configuration issue of LRZ that we are using this cnames.”

- `globusrun-ws` fails immediately, printing

```
Error submitting job
globus_soap_message_module: Failed sending request
ManagedJobFactoryPortType_createManagedJob
```

This was fixed by re-starting the Globus container.

- `globusrun-ws` fails immediately, printing

```
globus_service_engine_module: Session failed to start
```

This was a crashed Globus daemon.

- Standard error of job says

```
Globus is not available on this node.
PSILogger: Child with rank 0 exited with status 1.
...
```

ParaStation was not completely configured on all the nodes, due to a recent reconfiguration of the hardware.

- `module avail` doesn't show the expected list of available modules

In this case, the cause was a bad symlink.

- a reasonable PBS `qsub` results in

```
qsub: Job rejected by all possible destinations
```

Here we had requested just 1 minute walltime, but the batch system required *at least* 5 minutes to be specified!

- Standard error says “Missing or wrong argument: `-np`”, although the JDD script has `jobType` set to `mpi`.

“Erzeugung der Job-Skripte durch den PBS JobManager von Globus. In GT 4.0.5 scheint dieses Problem gefixt zu sein. Auf mardschana wird zur Zeit noch GT 4.0.3 eingesetzt.

Der JobManager fuer PBS wurde aktualisiert, so dass das Problem nun nicht mehr auftauchen sollte.”⁶

- Standard error has

```
Module ZIBenv loaded
libibverbs: Fatal: couldn't read uverbs ABI version.
libibverbs: Fatal: couldn't read uverbs ABI version.
```

```
PSILogger: done
```

This problem was reported and fixed in these interchanges with the DGUS system (available to D-Grid users):

```
https://iwrldgus.fzk.de/pages/ticket\_details.php?ticket=283
```

```
https://iwrldgus.fzk.de/pages/ticket\_details.php?ticket=284
```

But the exact cause has not been sorted out.

- In standard error, see “Missing or wrong argument: -np.”

In file

```
$GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/pbs.pm,
```

“replaced the pbs_cmd_script.sh script with the real executable”

- (on Hydra) The default queue is “express”, which has a limit of 4 nodes. So except for the smallest jobs, a user must explicitly request queue “long”.
- In an otherwise successful job submission, staged-in files are corrupted.

This was a known bug with GT 4.0.6. It was fixed by applying on the resource the Globus patch of 2008-03-17: “globus_wsrft_service_java-0.37”.

⁶Generation of job scripts by pbs jobmanager fixed. This problem seems to be resolved in GT 4.0.5. Currently we are running GT 4.0.3 on mardschana. The jobmanager was actualized, so this problem should not occur any longer.

References

- [1] Deliverable 1.2 (AGD-WG1), *Integration of AstroGrid-D Resources*, http://www.gac-grid.org/project-documents/deliverables/wp1/ResourceIntegration-D_1_2_1.0.0.pdf
- [2] Deliverable 1.3 (AGD-WG1), *Resource Integration and Grid Support* http://www.gac-grid.org/project-documents/deliverables/wp1/Working_AGD_Grid-D_1_3_1.0.1.pdf
- [3] Deliverable 1.5 (AGD-WG1), *MPI over the Grid* http://www.gac-grid.org/project-documents/deliverables/wp1/MPI_over_Grid-D_1_5.pdf
- [4] Deliverable 2.6 (AGD-WG2), *Advanced Prototype Implementation of Metadata Information Providers* http://www.gac-grid.org/project-documents/deliverables/wp2/D2_6_Information_Providers.pdf
- [5] Deliverable 5.1 (AGD-WG5) *Ressourcen-Management für Grid-Jobs*, <http://www.gac-grid.org/project-documents/deliverables/wp5/D5.1.pdf>
- [6] MPI Standard <http://www-unix.mcs.anl.gov/mpi/>
- [7] Globus Toolkit 4 Manuals <http://www.globus.org/toolkit/docs/4.0/>
- [8] WS-GRAM User's Guide <http://www.globus.org/toolkit/docs/4.0/execution/wsgram/user-index.html>
- [9] WS-GRAM Job Description Document (JDD) schema http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/gram_job_description.html
- [10] Job Description Extensions Support http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Job_Desc_Extensions.html
- [11] IBM LoadLeveler <http://www-03.ibm.com/systems/clusters/software/loadleveler/>
- [12] Cluster Resources' PBS Torque <http://www.clusterresources.com/pages/products/torque-resource-manager.php>
- [13] Platform LSF <http://www.platform.com/Products/platform-lsf-family/platform-lsf/>
- [14] Sun Grid Engine <http://gridengine.sunsource.net/>
- [15] Condor <http://www.cs.wisc.edu/condor/>
- [16] Environment Modules package <http://modules.sourceforge.net/>
- [17] SoftEnv, part of the MCS Systems Group's "Msys Toolkit": <http://www-new.mcs.anl.gov/systems/software/>
- [18] ParaStation from ParTec Cluster Competence Center GmbH <http://cluster-competence-center.com/>