



Arbeitspaket 3: Langzeitarchivierung von Forschungsdaten

JHOVE2 over the Grid¹

Autoren	Arbeitspaket 3: Langzeitarchivierung von Forschungsdaten
Editoren	White
Datum	28. März 2011
Dokument Status	Entwurf
Dokument Version	0.1.0

Änderungen

Version	Datum	Name(n)	Kurzinfo
0.1.0	16.03.2011	White	Initiated

¹This work is created by the WissGrid project. The project is funded by the German Federal Ministry of Education and Research (BMBF).

Inhaltsverzeichnis

1	Project Overview	3
1.1	Mission and Scope	3
1.2	Status	3
1.3	Project Documents	3
2	Design	4
2.1	Introduction	4
2.2	Design Checklist	4
3	Technische Umsetzung	5
3.1	Type of Implementation	5
3.2	Runtime Environment	5
3.3	Operational Procedures	6
4	Installation und Benutzung	7
4.1	Introduction	7
4.2	Minimal System Requirements	7
4.3	Installation	7
4.4	Getting Started	7
5	GDIST : Versionsinformationen	8
5.1	Introduction	8
5.2	Installation and Upgrade Notes	8
5.3	Known Problems and Workarounds	8
5.4	Programming information	8
0.5	Introduction	9
0.6	Minimal System Requirements	9

0.7	Installation	9
0.8	Getting Started	9
1	Running file from GDIST package	10
2	Troubleshooting file from GDIST package	13

1 Project Overview

1.1 Mission and Scope

This is a feasibility study for doing metadata processing over the grid

1.2 Status

Tests of the JHOVE2 executable have been deployed to grid resources using grid services, and run on grid compute hosts, and the results have been successfully retrieved from the hosts via grid services.

1.3 Project Documents

- JHOVE2
- GDIST
- For Management: . . .

2 Design

2.1 Introduction

Requirements

1. To run the JHOVE2 application on the Grid
2. To send the JHOVE2 data via Grid protocols
3. To retrieve the JHOVE2 results via Grid protocols

What are the prioritized goals of this design?

1. Feasibility
2. Understandability
3. Modularity
4. Extensibility

2.2 Design Checklist

Feasibility: What indicates that this design can be implemented and tested with the planned amount of time and effort?

To run the JHOVE application using the existing GDIST scripts required a few hours' work, mostly in modifying one script and testing. As little further modification would be required, we can say that running further JHOVE applications would be easily achieved.

Understandability: What makes this design understandable?

The GDIST scripts are intended to remove several layers of technical complications from the job of executing distributed jobs on the grid. To do a specific run, typically all that is required is to produce input tarballs, and refer to them in a single command line.

Modularity: How have concerns been separated and addressed in distinct modules?

The GDIST scripts are modularized to some degree, to separate issues of host selection from job execution and job logging.

Extensibility: How can new features can be easily added later?

Any changes in the way individual JHOVE jobs are run can be accommodated in the `single.pl` script.

3 Technische Umsetzung

3.1 Type of Implementation

Type of system:

- Unix-style command
- Distributed computing system

Programming Language(s):

- Perl

Data Storage:

- Computer files

Security Technologies:

- Authentication: Grid certificates
- Encryption: SSL

3.2 Runtime Environment

Processes:

- Main application script `griddist.pl`.
- Globus middleware running on initiating machine
- Globus middleware running on compute cluster head node
- Cluster resource-management system running on cluster
- Job processes running on cluster compute nodes (the `jhove2` application and the script `single.pl` that unpacks it, launches it, and packs up its output)

Configuration Files:

- `grid_cluster_hosts.csv`: Choice of preferred grid cluster hosts and information about them.
- `single.sh`: shell script detailing how input tarballs are to be unpacked and run, and how the output is to be packed.

Data files:

- *input.tgz*: a set of input tarballs including the executable and input data and parameters.

Temporary files:

- *~/grid_dist/timestamp.xml*: Globus job description file

Log files:

- *~/grid_dist/timestamp.log*: log file for the job identified by its timestamp.
- On the grid compute host head node, *~/GDIST/timestamp/**: for each individual job process.

3.3 Operational Procedures

- **Obtain grid certificates:**
- **Install:** See the file *Install* On the initiating system a group 'gdist' must be defined. All input files must be readable by group gdist, and the output directory must be writable by that group.
- **Configure:** See the file *Install* Particularly, the file *single.sh* must be customized.
- **Create input tarballs:** These contain the binary executable and whatever inputs and parameters are required. They are unpacked and run by *single.sh*, and also their output is packed for return by that script.
- **Run main job script:**

```
perl grid_dist.pl outdir [options] input1.tgz input2.tgz ...
```
- **Monitor Activity:** Script *job_status.pl* is provided to monitor progress of running jobs. Output is like that of the main job script, so the main script can be killed, but the job it started remains running, and monitoring can be continued.
- **Periodic Cleanup:** Script *job_cleanup.pl* is provided to clean up leftover files and connections from any dead or stalled processes.

4 Installation und Benutzung

4.1 Introduction

This document describes how to install and get started with GDIST for JHOVE2.

4.2 Minimal System Requirements

The same as the minimal requirements for a Globus installation.

4.3 Installation

What other software must be installed first?

Before you can install this product, you must install the following packages:

- [Globus GTK 4.0.8](#)
- Perl packages for modules
 - XML::Writer
 - Net::Domain
 - File::Spec
 - List::Util
 - Getopt::Long

Obtain the GDIST software package

```
svn co svn://svn.gac-grid.org/software/grid_dist
```

How do I install GDIST?

See the file `Install`.

What if I encounter problems?

See the file `Troubleshooting`.

4.4 Getting Started

See the file `Running`.

5 GDIST : Versionsinformationen

5.1 Introduction

This is an early access release for evaluation and usage by select partners. Your feedback is important to us, please help us make this the best product possible.

5.2 Installation and Upgrade Notes

Installation See the file `Install` for full details.

Manifest

This release consists of the following items:

- Release notes (this file)
- Installation instruction
- Running guide
- Product source code

Minimum System Requirements

- **Operating System:** Linux
- **Networking:** Internet access
- **Existing Software:** Globus GTK 4.0.8

5.3 Known Problems and Workarounds

Installation See the file `Troubleshooting` for full details.

5.4 Programming information

The Perl files of the GDIST package contain programming documentation in the Perl POD format, which can be extracted with the appropriate tools, for instance, `pot2text` or `pod2html`.

0.5 Introduction

This document describes how to install and get started with GDIST for JHOVE2.

0.6 Minimal System Requirements

The same as the minimal requirements for a Globus installation.

0.7 Installation

What other software must be installed first?

Before you can install this product, you must install the following packages:

- [Globus GTK 4.0.8](#)
- Perl packages for modules
 - XML::Writer
 - Net::Domain
 - File::Spec
 - List::Util
 - Getopt::Long

Obtain the GDIST software package

```
svn co svn://svn.gac-grid.org/software/grid_dist
```

How do I install GDIST?

See the file `Install`.

What if I encounter problems?

See the file `Troubleshooting`.

0.8 Getting Started

See the file `Running`.

1 Running file from GDIST package

How to run the grid_dist.pl script.

command syntax

=====

```
grid_dist.pl output_directory input-filepath1 input-filepath2 ...
```

The paths may be relative or full file paths.

The input_filepaths must be readable by the grid user

The output_directory must be writable and executable by the grid user

switches

--verbose

print data involved in intermediate steps

--status

print everything that goes to the log file, and other info

--limit-cores

takes an integer value. limit number of processes

run on a single node to this number.

grid hosts file

=====

A list of grid hosts, with some information about how to run jobs on them, is in the comma-separated-value file

grid_cluster_hosts.csv

The script tries the grid hosts in the order they appear in this file.

The fields are

IP-address, local-resource-manager, queue, cores-per-node

The local resource manager is meaningful for cluster hosts, and can take values

PBS, SGE, Condor, LSF, LoadLeveler, GridWay

corresponding to the cluster job manager running on that host.

The default manager is Fork, which runs the job as a unix process (Fork is NOT RECOMMENDED for clusters! -- people will react).

The queue refers to a local resource manager queue, and is only

needed if you want your job to run in a special queue on the

cluster resource manager.

The part of a line after a hash mark '#' is ignored.

debugging

=====

log files of grid_dist.pl script

The script on the grid host appends messages to a log file

~/grid_dist/<timestamp>.log

The log contains a report of job statuses reported as the script

polls its list of jobs.

StageIn input files being copied
Pending job is in the local batch system's queue
Active the local batch system has started the execution script
StageOut output files being copied
CleanUp remaining files are being deleted on the compute host
Unsubmitted waiting for local batch system to take job (bad)
Done
Failed

Note that this polling may easily miss a step: for example, if a job is only briefly in status "Pending", so that that status lasts only between polls, it may not show up in the log.

job output files

The standard output and standard error from the globusrun-ws job started on the compute host is stored in the grid account on the compute host. These are useful to inspect the execution time status of the job.

To view them, gsissh to the compute host, and look in
~/GDIST/

They are named according to the script start time.

debug switch

A script variable '\$debug' set to a string containing:
'status' to print basic state information, and anything that normally goes to the local log file.
'verbose' to display the inputs to various functions.

monitoring

=====

In normal operation, the script polls all the running jobs until they have all stopped, to report progress. Besides starting the jobs, however, it has no further influence on them.

The scripts

job_status.pl
job_cleanup.pl

are provided to check the present status of any running jobs and to kill undesired jobs.

They act on Globus EPR files found in the ~/.grid_dist directory. By default, they act on all the *.epr files they find. However, they both take as argument a substring to match in the filename, and in the presence of an argument, act only on *.epr files that contain the substring.

The job_cleanup.pl script will attempt to properly kill and clean up any remaining jobs, and then deletes the associated EPR file.

job description files

=====

Each attempt to run the job on a grid host will result in an XML "job description" file being written into the directory
~/grid_dist .

It is used only for input to the Globus job running command, and are normally deleted by the script immediately after use.

If the script is interrupted however, you may see these files, named according to the script start time, in the GDIST/ directory. They may safely be deleted.

2 Troubleshooting file from GDIST package

- 1) permissions
 - the home directory of the submitting unix user will need at least 'execute' privileges for the corresponding grid user

 - the directory where the executable is run will need at least read and execute permissions for the grid user

 - the directory where the output file will go needs full write privileges for the grid user.
- 2) globus running on submit host as well as compute host
(also requires postgres daemon with the RSL database to be running)

It is normal for the script to fail on some compute hosts: on the grid, some machines will be down or otherwise non-functional.

Something like this is typical of Globus not running on the submit host:

```
-----  
globusrun-ws: Job failed: Staging error for RSL element fileStageIn.  
Transient transfer error  
Server refused performing the request. Custom message: (error code 1)  
[Nested exception message: Custom message: Unexpected reply:  
500-Command failed. : globus_l_gfs_file_open failed.  
500-globus_xio: Unable to open file ../multi-proc.pl  
500-globus_xio: System error in open: Permission denied  
500-globus_xio: A system call failed: Permission denied  
-----
```