

JAVA CoG KIT GUIDE TO USE MPI

Gregor von Laszewski and Kaizar Amin

Software Version: 4.1.3

Url: <http://www.cogkit.org/release/4.1.3/manual/mpi.pdf>

Url: <http://www.cogkit.org/release/4.1.3/manual/mpi/mpi.html>

Last update: September 3, 2005

CONTENTS

1	About this Document	2
2	Registration	2
3	Introduction	3
4	Example MPI Program	3
5	Testing the MPI Program on TeraGrid	3
6	MPI Job on TeraGrid using mpirun	4
7	MPI Job on TeraGrid using qsub	5
8	MPI Job on TeraGrid using MPICH-G2	7
	Appendix	11
A	Documentation	11
B	Downloads	12
C	Availability of the Document	12
D	Bugs	13
E	Administrative Contact	13

1. ABOUT THIS DOCUMENT


This document includes basic information about running MPI programs through the Java CoG Kit.

1.1. Viewing

The best way to read this document is to use the PDF version and read it with Adobe Acrobat Reader. Please make sure you configure Adobe Acrobat Reader appropriately so you can follow hyperlinks. This is the case if you follow the default installation. Acrobat Reader is available at <http://www.adobe.com/products/acrobat/readermain.html>. Because the hyperlinks are not available in the printed form of this manual and we support saving our environment we strongly discourage printing this document.

We recommend that you save this manual locally on your machine and use Acrobat Reader. This has the advantage that you do not lose your anchor points while switching back and forth between different hyperlinks. An HTML version of this manual is planned, but not available yet.

1.2. Format

We have augmented the document with some comments at places where we found issues. Our intend is to address these issues in a future release. The comments are marked by the icon  and the name of the person that will work on the removal of the issue.

2. REGISTRATION

Please be a team player and support us indirectly by registering with us or reporting your use of the Java CoG Kit. Although this software is free, we still need to justify to our funders the usefulness of the projects. If you want to help us with our efforts please take a few seconds to complete this information. We do not use this information for other purposes. If you have special needs or concerns please contact gregor@mcs.anl.gov. The registration form can filled out in a variety of formats. The online form can be found at

<http://www.cogkit.org/register>

This form is available also as ASCII text at

<http://www.cogkit.org/register/form.txt>

which you can FAX to

Gregor von Laszewski, Fax: 630 252 1997

3. INTRODUCTION

The Message Passing Interface (MPI) provides a powerful programming paradigm for high performance computing based on the interchange of messages between processes. An optimized standardized interface is available on almost all high performance computing platforms.

In this guide we will focus on the submission of MPI jobs to a supercomputer or distributed cluster. As an example serves the TeraGrid [1]. However, the examples discussed in this guide are generic enough for you to modify it for your own server environment. We will demonstrate how to submit MPI programs through the Java CoG Kit with the help of the *mpirun* and the *qsub* commands. Although we will use the GT2 provider for our submission examples, other providers such as the SSH provider can also be used.

We assume that you have acquired a TeraGrid accounts and that you have a Grid certificate that allows you to submit jobs on the TeraGrid. If you do not know how to do this, please consult with the TeraGrid Web site at <http://www.teragrid.org>.

4. EXAMPLE MPI PROGRAM

We have chosen an extremely simple test program that prints on each processor the rank and the number of requested processors.

```
#include <stdio.h>
#include "mpi.h"

int main( argc, argv )
int  argc;
char **argv;
{
    int rank, size;
    MPI_Init( &argc, &argv );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    printf( "Hello world from process %d of %d\n", rank, size );
    MPI_Finalize();
    return 0;
}
```

For the rest of this guide, we will assume that you have this program on the TeraGrid under the following directory: *\$HOME/mpitest/helloworld.c*

5. TESTING THE MPI PROGRAM ON TERAGRID

It is important to make sure that you can compile the program on a teragrid frontend node, just to verify that your TeraGrid account is properly configured. As an example configuration we will submit it on the TeraGrid *frontend* node and run it on four *backend* nodes.

To test and build an executable for the TeraGrid, please login first to the frontend node. In this document we will use the name "laszewski" as a place-holder for your username. Please change the example appropriately.

```
> ssh laszewski@tg-login.uc.teragrid.org
```

Place the MPI *helloworld.c* program in the directory *mpitest*. This program can now be compiled with the *mpicc* compiler.

```
> mpicc -o helloworld helloworld.c
```

We recommend that you check out if you can run this program on the TeraGrid by simply using the TeraGrid mpirun program. In our example we execute it on four processors.

```
mpirun -np 4 helloworld
```

You should get the following output:

```
Hello world from process 0 of 4
Hello world from process 3 of 4
Hello world from process 1 of 4
Hello world from process 2 of 4
```

If for some reason this simple test program does not work, your account may not properly configured and you should consult with a TeraGrid systems administrator. However, if your program works, you can proceed to the next examples. The next examples do not require that you are logged into a TeraGrid node. Instead, you can execute them from the clients where you have installed the Java CoG Kit.

6. MPI JOB ON TERAGRID USING MPIRUN

We assume that you will submit the job to the host *tg-grid1.uc.teragrid.org* while using the Globus Toolkit version 2 submission (GRAM) service. We will be using the *mpirun* command that is available in a directory under */soft* on the TeraGrid. We will place the output from this program in your home directory on the TeraGrid with the name *cog-mpirun.out*. We will reuse the helloworld program that you have previously compiled on the TeraGrid as discussed in Section 5.

Before you can submit a job you have to first authenticate your self. Do do so, you create a proxy certificate on the client machine that is used as part of the initialization step.

```
> cog-proxy-init
```

The command for submitting the job is

```
> ./cog-job-submit -s tg-grid1.uc.teragrid.org \
                  -p gt2 \
                  -e /soft/mpich-gm-1.2.5..10-intel-r2a/bin/mpirun \
                  -args "-np 4 helloworld" \
                  -stdout cog-mpirun.out
```

The meaning of the flags is as follows:

- s:** The service location of the globus gatekeeper installed on the teragrid. For the ANL/UC sites this is *tg-grid1.uc.teragrid.org*
- p:** The provider used for this execution. In our example we use GT2 provider
- e:** The executable that is run on the teragrid machine. For our mpi program it is *mpirun*. Note that we give the complete absolute path of the mpirun program. For the ANL/UC site it is */soft/mpich-gm-1.2.5..10-intel-r2a/bin/mpirun*
- args:** The arguments that need to be supplied with the executable. We want to specify that the mpirun should use 4 backend nodes and executes the helloworld program, hence *"-np 4 helloworld"*

-stdout: The remote file that should be used to redirect the standard output. In our example it will be `/home/laszewski/cog-mpirun.out` on the TeraGrid login node (tg-login.uc.teragrid.org)

The output for this command looks similar to the following. However, the contents may be slightly different as we have reformatted the output for this guide.

```
DEBUG [org.globus.cog.core.examples.execution.JobSubmission]
  - Task Identity: urn:cog-1099697597908
DEBUG [org.globus.cog.core.impl.common.CoreFactory]
  - Instantiating org.globus.cog.core.impl.execution.gt2.TaskHandlerImpl
    for provider gt2
DEBUG [org.globus.cog.core.impl.execution.gt2.JobSubmissionTaskHandler]
  - RSL:
    &(executable=/soft/mpich-gm-1.2.5..10-intel-r2a/bin/mpirun)
    (arguments=-np 4 helloworld)
    (stdout=cog-mpirun.out)
DEBUG [org.globus.cog.core.examples.execution.JobSubmission]
  - Status changed to Submitted
DEBUG [org.globus.cog.core.examples.execution.JobSubmission]
  - Status changed to Active
DEBUG [org.globus.cog.core.examples.execution.JobSubmission]
  - Status changed to Completed
DEBUG [org.globus.cog.core.examples.execution.JobSubmission]
  - Job completed
```

On the TeraGrid, the file `cog-mpirun.out` will be created with the following contents.

```
tg-login1:~> cat cog-mpirun.out
Hello world from process 2 of 4
Hello world from process 3 of 4
Hello world from process 0 of 4
Hello world from process 1 of 4
```

If this program does not work, make sure that you log once more in on the TeraGrid and verify with

```
> which mpirun
```

if the location of `mpirun` is properly obtained.

7. MPI JOB ON TERAGRID USING QSUB

Previously we showed how to submit an MPI program with the help of `mpirun`. In certain circumstances you may need more control over the queue parameters or may just submit a non MPI program. In these cases it is useful to use the job submission routine that is part of the batch processing system installed on the machine. At time of writing of this guide, the UC teragrid node used the `qsub` program from torque. To submit a job through `qsub` you need to make sure that you create a script that is submitted as part of the submission process. We assume that you have placed the script (`qsub-script`) in your home directory on the TeraGrid (`/home/laszewski/qsub-script`).

```
> ./cog-job-submit -s tg-grid1.uc.teragrid.org \
                  -p gt2 \
                  -e /soft/torque-1.1.0p0-r1a/bin/qsub \
                  -args "qsub-script"
```

The meaning of the flags has been explained in the previous section. They just have different values.

The output for this command looks similar to the following. However, the contents maybe slightly different as we have reformatted the output for this guide.

```
DEBUG [org.globus.cog.core.examples.execution.JobSubmission]
- Task Identity: urn:cog-1099698205741
DEBUG [org.globus.cog.core.impl.common.CoreFactory]
- Instantiating org.globus.cog.core.impl.execution.gt2.TaskHandlerImpl
  for provider gt2
DEBUG [org.globus.cog.core.impl.execution.gt2.JobSubmissionTaskHandler]
- RSL: &(executable=/soft/torque-1.1.0p0-r1a/bin/qsub)(arguments=mpi/script)
DEBUG [org.globus.cog.core.examples.execution.JobSubmission]
- Status changed to Submitted
DEBUG [org.globus.cog.core.examples.execution.JobSubmission]
- Status changed to Completed
DEBUG [org.globus.cog.core.examples.execution.JobSubmission]
- Job completed
```

This example assumes you have the script "qsub-script" on the remote machine. The contents of the file is as follows.

```
> cat qsub-script
#!/bin/sh
#
#PBS -q dque
#PBS -N example
#PBS -l nodes=4:ia64-compute:ppn=2
#PBS -l walltime=0:05:00
#PBS -A TG-ABC
#PBS -o helloworld-pbs.out
#PBS -e helloworld-pbs.err
#
## Export all my environment variables to the job
#PBS -V
#
## Change to my working directory
cd $HOME/mpi/
#
## Run my parallel job (the PBS shell knows PBS_NODEFILE)
mpirun -machinefile $PBS_NODEFILE -np 4 ./helloworld
```

However, in order to make this script work, you must adapt the account information appropriately (we use here *TG-ABC*).

Once this program is run, the output in form of the standard output and standard error are written into the files "helloworld-pbs.out" and "helloworld-pbs.err". The contents of the file helloworld-pbs.out will look similar to the one listed below.

```
cat helloworld-pbs.out
-----
Begin PBS Prologue Fri Nov  5 17:45:00 CST 2004
Job ID:          146641.tg-master.uc.teragrid.org
Username:        laszewski
```

```

Group:          allocate
Nodes:          tg-c047 tg-c048 tg-c051 tg-c052
End PBS Prologue Fri Nov  5 17:45:07 CST 2004
-----
Hello world from process 2 of 4
Hello world from process 0 of 4
Hello world from process 3 of 4
Hello world from process 1 of 4
-----
Begin PBS Epilogue Fri Nov  5 17:45:29 CST 2004
Job ID:         146641.tg-master.uc.teragrid.org
Username:       laszewski
Group:          allocate
Job Name:       example
Session:        12505
Limits:         nodes=4:ia64-compute:ppn=2,walltime=00:05:00
Resources:      cput=00:00:04,mem=1312kb,vmem=3808kb,walltime=00:00:14
Queue:          dque
Account:        TG-ABC
Nodes:          tg-c047 tg-c048 tg-c051 tg-c052

Killing leftovers...

End PBS Epilogue Fri Nov  5 17:45:38 CST 2004
-----

```

For more details on PBS scripts please take a look at the TeraGrid help pages for PBS scripts on: http://www.teragrid.org/userinfo/guide_jobs_pbs.html

8. MPI JOB ON TERAGRID USING MPICH-G2

In this section we discuss MPI job submissions on the TeraGrid from the Java CoG Kit using the Globus Toolkit's MPICH-G2 libraries. We assume that the user is familiar with formulating MPICH-G2 tasks as well as the TeraGrid MPICH-G2 environment. A good introductory background on basic MPICH-G2 concepts is available at <http://www3.niu.edu/mpi/>. Information on establishing the MPICH-G2 environment on TeraGrid is available at http://www.teragrid.org/userinfo/guide_jobs_mpich_g2.html. For this Section, we assume the TeraGrid UC/ANL site. Further, we assume the following "soft" environment on the ANL machines.

```

> cat ~/.soft
@remove +globus
@remove +mpich-gm-intel
@remove +mpich-gm-gcc
+globus-2.4.3-intel-r5
+mpich-g2-intel
@teragrid

```

For our example, we will use the same *helloworld.c* program described in Section 4. Place the MPI *helloworld.c* program in the directory *mpichg2*. This program can now be compiled with the MPICH-G2 based *mpicc* compiler.

```

> which mpicc
/soft/globus-2.4.3-intel-r5/mpich-g2-1.2.6/bin/mpicc

```

```
> mpicc -o helloworld helloworld.c
```

Once the compilation is successful, the server-side environment setup is completed. Next, we submit an MPICH-G2 task from the client machine which has the Java CoG Kit installed. For our example, we will formulate the following RSL:

```
+(
  &(resourceManagerContact=tg-grid1.uc.teragrid.org/jobmanager-pbs_gcc)
  (count=4)
  (hostcount=4:ia64-compute)
  (project=<your project number>)
  (jobtype=mpi)
  (label=subjob 0)
  (environment=
    (GLOBUS_DUROC_SUBJOB_INDEX 0)
    (LD_LIBRARY_PATH
      /soft/globus-2.4.3-intel-r5/lib/:/soft/intel-c-8.0.066-f-8.0.046/lib/))
  (directory=/home/laszewski/mpichg2)
  (executable=/home/laszewski/mpichg2/helloworld)
  (stdout=/home/laszewski/mpichg2/helloworld.out)
  (stderr=/home/laszewski/mpichg2/helloworld.err)
)
```

The above RSL is formulated based on the information available at http://www.teragrid.org/userinfo/guide_jobs_mpich_g2.html and our TeraGrid environment. The various parameters should be appropriately adjusted to reflect your TeraGrid server environment. Additionally, replace the “(project=your project number)” with the appropriate TeraGrid project number.

Please Note that at the time of writing this guide, it was required to include */soft/intel - c - 8.0.066 - f - 8.0.046/lib/* directory in the “LD_LIBRARY_PATH” for this example to work. This was due to an MPICH-G2 installation problem at the ANL/UC site, which might have been fixed now.

Next, we create a proxy certificate on the client machine.

```
> ./cog-proxy-init
```

The command for submitting the job using the Java CoG Kit is as follows:

```
> ./cog-job-submit -specification
'+(
  &(resourceManagerContact=tg-grid1.uc.teragrid.org/jobmanager-pbs_gcc)
  (count=4)
  (hostcount=4:ia64-compute)
  (project=<your project number>)
  (jobtype=mpi)
  (label=subjob 0)
  (environment=
    (GLOBUS_DUROC_SUBJOB_INDEX 0)
    (LD_LIBRARY_PATH
      /soft/globus-2.4.3-intel-r5/lib/:/soft/intel-c-8.0.066-f-8.0.046/lib/))
  (directory=/home/laszewski/mpichg2)
  (executable=/home/laszewski/mpichg2/helloworld)
```

```
(stdout=/home/laszewski/mpichg2/helloworld.out)
(stderr=/home/laszewski/mpichg2/helloworld.err)
),'
```

The output for this command looks similar to the following. However, the contents maybe slightly different as we have reformatted the output for this guide.

```
DEBUG [org.globus.cog.abstraction.examples.execution.JobSubmission] -
Task Identity: urn:cog-1123748613059
DEBUG [org.globus.cog.abstraction.examples.execution.JobSubmission] -
Status changed to Submitted
DEBUG [org.globus.cog.abstraction.examples.execution.JobSubmission] -
Status changed to Active
DEBUG [org.globus.cog.abstraction.examples.execution.JobSubmission] -
Status changed to Completed
Job completed
```

On the TeraGrid, the file *helloworld.out* will be created with the following contents.

```
> cat helloworld.out
-----
Begin PBS Prologue Thu Aug 11 01:45:01 CDT 2005
Job ID:          196562.tg-master.uc.teragrid.org
Username:       laszewski
Group:          allocate
Nodes:          tg-c059 tg-c060 tg-c061 tg-c062
End PBS Prologue Thu Aug 11 01:45:03 CDT 2005
-----
Hello world from process 0 of 4
Hello world from process 1 of 4
Hello world from process 3 of 4
Hello world from process 2 of 4
-----
Begin PBS Epilogue Thu Aug 11 01:45:26 CDT 2005
Job ID:          196562.tg-master.uc.teragrid.org
Username:       laszewski
Group:          allocate
Job Name:       STDIN
Session:        13536
Limits:         nodes=4:ia64-compute,walltime=00:01:00
Resources:      cput=00:00:04,mem=1312kb,vmem=3808kb,walltime=00:00:10
Queue:          dque
Account:        <your project number>
Nodes:          tg-c059 tg-c060 tg-c061 tg-c062

Killing leftovers...

End PBS Epilogue Thu Aug 11 01:45:40 CDT 2005
-----
```

REFERENCES

[1] "TeraGrid," Web Page, 2001. [Online]. Available: <http://www.teragrid.org/>

- [2] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, “A Java Commodity Grid Kit,” *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8-9, pp. 643–662, 2001. [Online]. Available: <http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--cog-cpe-final.pdf>
- [3] “Java CoG Kit Wiki,” 2004. [Online]. Available: <http://www.cogkit.org/wiki>
- [4] “Java CoG Kit Registration,” 2004. [Online]. Available: <http://www.cogkit.org/register>

Additional publications about the Java CoG Kit can be found as part of the vita of Gregor von Laszewski <http://www-unix.mcs.anl.gov/~laszewsk/vita.pdf>. Most documents are available online if you follow the links. In future we intend to provide this information without Gregors vita data.

If you need to cite the Java CoG Kit, please use [2].

A. DOCUMENTATION

The Java CoG Kit documentation is distributed as a series of guides. These guides for version 4.1.3 include:

A.1. Java CoG Kit Guides

Short Title	Audience	Description	Format
Install	All	A guide to the different ways of installing the Java CoG Kit	[PDF] [HTML]
Commands	User	A guide to the command line tools of the Java CoG Kit	[PDF] [HTML]
Workflow	User	A guide to the Karajan Workflow	[PDF] [HTML]
Abstractions	User	A guide to the Java CoG Kit abstractions API	[PDF] [HTML]
MPI	User	A guide to execute MPI programs on the TeraGrid	[PDF] [HTML]
Coding	Developer	A guide to the Coding rules for the Java CoG Kit	[PDF] [HTML]

A.2. Java CoG Kit Guides Under Construction

More guides are under development. The following guides are not yet completed, but are listed here to help us improving these guides. Please, explore them and send us e-mail about improvement suggestions. If you like to contribute a guide yourself, please contact gregor@mcs.anl.gov.

Short Title	Audience	Description	Format
Writing Guides	Developer	A preliminary guide to document writing guides	[PDF] [HTML]
Examples	User	A preliminary guide to examples	[PDF] [HTML]
Release Process	Developer	A preliminary guide to document the release process	[PDF] [HTML]

A.3. API Documentation

- [JGlobus](#)
- [Common classes and interfaces for the Java CoG Kit abstraction layer](#)
- [Utility Classes](#)
- [Graph Editor](#)
- [Various examples for the Java COG Kit abstraction layer](#)
- [GT2 Provider for Java CoG Kit abstractions](#)
- [GT2 Provider for Java CoG Kit abstractions with fault tolerance](#)
- [GT3.0.2 provider for the abstraction layer](#)
- [GT3.2.0 provider for the abstractions](#)

- [GT3.2.1 provider for abstractions](#)
- [GT4.0.0 provider for abstractions](#)
- [Native Condor provider for the Java CoG Kit abstractions](#)
- [SSH provider for abstractions](#)
- [WebDav provider for abstractions](#)
- [Local provider for abstractions](#)
- [Karajan Workflow Engine](#)
- [Setup Wizard](#)
- [GridCertRequest Tool](#)
- [Certificate Management Applications](#)
- [GridShell](#)
- [GridFace Classes](#)

B. DOWNLOADS

Before downloading the Java CoG Kit, users should read the “Guide to Installing the Java CoG Kit” [\[PDF\]](#) [\[HTML\]](#). We hope that you will find this guide useful to decide which bundles you need. For the more experienced user, we provide the following index.

Binary Distributions • Complete (all providers) [\[tar.gz\]](#)[\[zip\]](#)

- Separate providers
 - Main package (includes GT2 providers) [\[tar.gz\]](#)[\[zip\]](#)
 - Common GT 3.x.x package [\[tar.gz\]](#)[\[zip\]](#) (required for all GT 3.x.x providers)
 - GT 3.0.2 provider [\[tar.gz\]](#)[\[zip\]](#)
 - GT 3.2.0 provider [\[tar.gz\]](#)[\[zip\]](#)
 - GT 3.2.1 provider [\[tar.gz\]](#)[\[zip\]](#)
 - GT 4.0.0 and 4.0.1 provider [\[tar.gz\]](#)[\[zip\]](#)
 - Condor provider [\[tar.gz\]](#)[\[zip\]](#)
 - SSH provider [\[tar.gz\]](#)[\[zip\]](#)
 - WebDAV provider [\[tar.gz\]](#)[\[zip\]](#)
 - Local provider [\[tar.gz\]](#)[\[zip\]](#)

Source Distributions • Source Distribution [\[tar.gz\]](#)[\[zip\]](#)

API Documentation • Complete API [\[tar.gz\]](#)[\[zip\]](#) [\[HTML\]](#)

Module List • Module list [\[HTML\]](#)

CVS • Please consult the Installation Guide [\[HTML\]](#) [\[PDF\]](#)

C. AVAILABILITY OF THE DOCUMENT

The newest version of this document can be downloaded by the developers from

1. `cvs -d:pserver:anonymous@cvs.cogkit.org:/cvs/cogkit checkout manual/guide`

It is not allowed to reproduce this document or the source. This documentation is copyrighted and is not distributed under the CoG Kit license.

D. BUGS

We use Bugzilla for tracking bugs and for enhancement suggestions. It is located in the bugzilla.globus.org, but you may find it easier to use one of the following quick links:

- [Submit new bug report](#)
- [List current bugs](#)
- [Make a simple query](#)
- [Make an advanced query](#)
- [Create a new account](#)

E. ADMINISTRATIVE CONTACT

The Java CoG Kit project has been initiated and is managed by Gregor von Laszewski. To contact him, please use the information below.

Gregor von Laszewski
Argonne National Laboratory
Mathematics and Computer Science Division
9700 South Cass Avenue
Argonne, IL 60439
Phone: (630) 252 0472
Fax: (630) 252 1997
gregor@mcs.anl.gov

INDEX

Administrative Contact, [13](#)

Contact, [13](#)

Registration, [2](#)