



Site Guidelines for Implementing the NGS Software Stack Module “M1: User Applications and Compilation”

Author

Jonathan Churchill jonathan.churchill@stfc.ac.uk (STFC RAL)

Change log

Version 1.1	13 th Oct 2009	Added MPICH reqment 3.4
Version 1.0	12 th Oct 2009	First initial draft

Table of contents

1. Introduction.....	1
2. Compilers, software libraries and compilation documentation	2
3. Supporting MPI	3
3.1 Vanilla Globus MPI.....	3
3.2 gLite WMS.....	4
3.3 gLite mpistart / WMS	5
3.4 GRIS Glue Publishing for MPI.....	5
4. Exposing existing local site applications to the NGS via UEE.....	6
5. Installing New Software for the NGS	7
6. Joining the NGS Bioinformatics database mirrors.....	8

1. INTRODUCTION

The majority of NGS end users run pre-installed software applications or compilations of code provided by other authors. It is less frequently used for the development of code developed by the end user themselves. Unlike grid projects such as EGEE / GridPP where virtual organisations (VOs) have designated software installers, who have login access to sites to install software for their VO users, the NGS relies on the local sites to install and expose (through GRIS and BDII services) software applications for users, or provide compilation environments for users to do this for themselves.

The NGS provides a wide range of software to end users. The current list is automatically updated from the NGS BDII server and made available at <http://www.ngs.ac.uk/applications>

The software provided by an NGS site may be software already available to local users or software specifically installed for NGS users at their or the NGS's request. The NGS provides a very simple mechanism to expose existing local site software for grid use – the Uniform Execution Environment (UEE) without requiring any special software installation locations or necessarily any sophisticated environment variable management (Eg environment ‘modules’ package is not necessarily required)

NGS sites have no SLD obligations to install and expose end user software but by doing so will contribute to the NGS community and more fully utilise hardware

resources. Sites that choose not to expose end user software, should provide as a minimum, some documented compilation environment.

This document provides guidelines and some, minimal, requirements for :

- 1) Compilers, software libraries and compilation documentation
- 2) Supporting MPI
- 3) Exposing existing local site applications to the NGS via UEE
- 4) Installing new Software for the NGS
- 5) Joining the NGS Bioinformatics database mirrors

2. COMPILERS, SOFTWARE LIBRARIES AND COMPILATION DOCUMENTATION

An NGS site should provide at least a default compiler suitable for their OS and a batch environment. For example on Linux the gcc/gfortran/g++ compilers should be provided, whereas a Windows platform may provide the Intel Compiler but not need to provide Visual C++. However some applications codes are sensitive to the compiler and provision of either the free or more up to date paid for Intel Compilers and/or the paid for Portland Group (PGI) Compilers are desirable.

Where possible sites should at a minimum provide MKL libraries suitable for their site architecture eg the ACML libraries from AMD for AMD processors and/or the free or paid for Intel MKL libraries.

Where a site has compilers and MKL libraries installed, they should provide sufficient documentation to end users, showing how to use and link them to their applications. This documentation should be provided via the site pages on the NGS web site. An example of such documentation can be seen here on the old web site:

<http://www.ngs.ac.uk/sites/ral/libraries/compilation.html>

or

<http://www.ngs.ac.uk/stfc/ngs2/compilation.html>

on the new web site. Site documentation should also include which site machine(s) users can login to (using gsissh) to compile code and any directory paths required should be given for both the compilation/login node as well as that seen on a cluster slave node (if different).

Compilations are not expected to be done remotely via WMS or globus batch jobs, although sites may choose to support (and document) this if appropriate. The NGS 'model' is that users compile code on the local site then either reference this in their job submissions (Eg a binary at \$HOME/bin/myprogram) or copy the compiled software back to the WMS UI and re-stage the executable along with input files. Sites may also offer to provide a shared NGS community software directory allowing users to compile code and make it available to other NGS users. The shared directory should have 1777 unix permissions (ie with the sticky bit set).

Compilers and MKL/software libraries often require environmental variables set. For example for shared libraries the LD_LIBRARY_PATH environment often needs updating. Sites may choose to manage these variables conveniently by installing the 'Modules Environment' (<http://modules.sourceforge.net/>) or by other methods. Exposing an application to the NGS does not require the modules environment but it does make managing larger numbers of applications easier – see section 4.

The compilers and software libraries available at a site are only visible to users via site documentation on the NGS web site. There are no automated mechanisms for exposing this information.

3. SUPPORTING MPI

Some sites may wish to support MPI applications. This may be appropriate for a site with a dedicated HPC cluster with high speed low latency slave node interconnect (Eg Myrinet or Infiniband or a dedicated slave node Gigabit local network) and a homogeneous pool of worker nodes. It is not likely to be appropriate for site offering access to a Condor pool for heterogeneous machines.

A site offering MPI capability should provide at least one installed and tested MPI stack. Popular stacks on the NGS are

MPICH (using myrinet gm or mx drivers)
<http://www.mcs.anl.gov/mpi/mpich1/index.htm>

OpenMPI
<http://www.open-mpi.org/>

Installation of an MPI stack usually requires passwordless ssh connections between nodes (based on host based authentication) in order for jobs to be able to launch copies of themselves on the other nodes in the job.

There are three possible ways that grid jobs can launch MPI jobs to a site:

- Vanilla Globus MPI : RSL = (Jobtype=MPI)(Count=n) { (HostCount=n) }
- gLite WMS : RSL = (jobtype=single)(Count=n){ (HostCount=n) }
- gLite mpistart / WMS : RSL = (jobtype=single)(Count=n) { (HostCount=n) }

3.1 Vanilla Globus MPI

This method requires the configuration of the local globus job manager (Eg /opt/vdt/globus/lib/perl/Globus/GRAM/JobManager/workq.pm) to instantiate the correct "mpirun" or "mpiexec" command and arguments for globus jobtype=mpi , setting the number of CPUs and hostnames for the job based on the local scheduler (eg PBS, LSF, SGE etc)

Sites should be careful to handle the globus (hostCount) RSL variable in the jobmanager. This has had different meanings in the past and some client tools assume that count = hostCount which for modern hardware is rarely the case. If hosts have 4 cpus each and a job requests (count=16)(hostcount=16) sites need to decide if this should launch on 16 physical hosts with 1 CPU allocated on each host or on 4 physical hosts with 4 cpus allocated on each host (or some other mix). However in the case

where (count=16)(hostcount=4) a user could be indicating that the job should run on 4 physical hosts on all 4 cpus of those hosts and trying to balance the application performance, so hostCount cannot be simply ignored. Some codes behave poorly, or crash, if distributed with different numbers of cpus of different hosts (For example PC-GAMESS).

3.2 gLite WMS

This is the preferred method for submitting all jobs to the NGS. Users submit jobs from a UI machine (Eg ngsui03.ngs.ac.uk) to the WMS service NGS WMS resource broker service which chooses which NGS site to run the job on based on the JDL criteria. For MPI jobs users will have set JobType=MPICH and CpuNumber=16, in their input job description, JDL, files :

```
jobname.jdl
=====
Jobtype="MPICH";
CpuNumber = 16;
Executable = /usr/ngs/NAMD;
Arguments = "namd2 inputfile";
InputSandBox = {"inputfile"};
OutputSandBox = {"outputfile"};
StdOutput = "tal.out";
StdError = "tal.err";
```

The WMS translates this into a globus RSL string of:

```
(queue=workq)(count=16)(hostCount=16)(jobtype=single)
(environment=(EDG_WL_JOBID 'https://ngswms01.ngs.ac.uk:9000/id-T1MxLaFtDLGODagcr9g'))
```

And specifies the executable as a shell script job wrapper which very simplified:

```
WMS Job Wrapper (very simplified !)
=====
#!/bin/sh

globus_url_copy gsiftp://ngswms01.ngs.ac.uk/...../inputfile
                file://$HOME/gram_scratch_ID/.mpi/https_JOBHANDLE/inputfile
# create machinefile as file 'host$$' based on local
# batch environment variables ie LSB_HOSTS for
# LSF or use $PBS_NODEFILE for PBS
# scp the input files to the slave nodes in the machinefile
#(ie No assumption of a shared homedir)
mpirun -np 16 -machinefile host$$ /usr/ngs/NAMD namd2 inputfile > tal.out 2>
tal.err
globus_url_copy file://$HOME/gram_scratch_ID/.mpi/https_JOBHANDLE/inputfile
                gsiftp://ngswms01.ngs.ac.uk/...../outputfile
exit
```

The key points are that:

- a) The globus jobtype is 'single' not 'mpi'
- b) The RSL is (count=16)(hostCount=16)
- c) The 'mpirun' command is the first one found in the users batch shell environment *on the worker/slave node*.
- d) The machinefile created by the WMS wrapper script may not be correct for the local site environment setup.

As the 'mpirun' command is embedded in the shell script submitted by the WMS and has fixed arguments, the local 'mpirun' in the users batch command PATH may have to be wrapped with another script that parses these arguments. For example a site may use 'mpiexec' to start up an MPI job or use an mpirun command that accepts '-n' option rather than '-np'.

The machine file is created by the shell script based on assumptions about the local scheduler environment ie \$LSB_HOSTS in LSF and the \$PBS_HOSTFILE in PBS. Again this may not be appropriate for the local site.

When testing MPI submissions, codes which use stdout redirect for output (such as NAMD) as well as codes that have named output file arguments (such as Gromacs -o option) should be tested.

3.3 gLite mpistart / WMS

gLite mpistart jobs use the gLite i2g-mpistart (<http://www.grid.ie/mpi/wiki/SiteConfig>) mechanisms to launch an MPI job. This is not the preferred way of submitting MPI jobs to the NGS although it may be required to support MPI jobs for sites that are also EGEE sites. This has the same issues as generic gLite WMS mpich jobs where a shell script that runs on the slave/worker nodes issues its own 'mpirun' command instead of the generic WMS job wrapper script. The mpirun command must still be found in the worker node path and be setup to behave correctly.

The globus RSL submitted for an mpistart job is the same as a generic WMS mpich job. The user however specifies jobtype=mpich for the generic job and jobtype=single for the mpistart job.

Follow the guidelines in the above URL to install and configure mpistart.

3.4 GRIS Glue Publishing for MPI

Sites providing MPI capability must publish this via their local GRIS and hence the BDII. The Glue entry:

GlueHostApplicationSoftwareRunTimeEnvironment: MPICH

This can be published by a simple GIP plugin file:

```
/.../vdt/lcg/var/gip/plugin/ngs-mpich-gip-plugin
```

Of contents similar to (replacing <TAB> with the tab character):

```
#!/bin/sh
#
# $Id: ngs-mpich-gip-plugin 1654 2009-09-04 22:49:17Z lecjpl $
#
# Simple GIP plugin to ensure that MPICH is reported as an application
#

fqdn=${GLOBUS_HOSTNAME:-`hostname --fqdn`}

dn="GlueSubClusterUniqueID=$fqdn,GlueClusterUniqueID=$fqdn,mds-vo-name=local,o=grid"

cat <<-END
<TAB>dn: $dn
<TAB>GlueHostApplicationSoftwareRunTimeEnvironment: MPICH
<TAB>END
```

4. EXPOSING EXISTING LOCAL SITE APPLICATIONS TO THE NGS VIA UEE

The NGS Uniform Execution Environment is a simple mechanism for exposing and publishing end use applications on the NGS. More extensive details can be found here <http://www.grid-support.ac.uk/files/UniformExecutionEnvironment.pdf> or in the presentation here: <http://www.ngs.ac.uk/ops/surgery/NGS%20UEE.ppt> but in essence, if a site already has an application installed under /usr/local/applications/myapp a site can expose this as available to the NGS by providing a simple wrapper script /usr/ngs/MYAPP which in concept is:

```
#!/bin/bash

module load myapp/1.2
# Or
# export PATH=$PATH:/usr/local/applications/myapp/bin
# export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/applications/myapp/lib

EXECUT=$1
shift
ARGS="$@"
$EXECUT $ARGS
exit $?
```

End user NGS Application documentation is then written with an example WMS job JDL:

```
Type = "Job";
JobType = "normal";
Executable = "/usr/ngs/MYAPP";
Arguments = "-i inputfile -o outputfile ";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {"inputfile"};
OutputSandbox = {"std.err", "outputfile", "std.out"};
Requirements = (
member("NGS-UEE-MYAPP", other.GlueHostApplicationSoftwareRunTimeEnvironment)
);
```

Or a globus command line example such as:

```
globusrun -b -r ngs.rl.ac.uk/jobmanager-lsf \
'&(jobtype=singe)(directory=/home/xxx/myappruns) \
(arguments=-I inputfile -o outputfile) \
(stderr=/home/xxx/myappruns/std.err)(stdout=/home/xxx/myappruns/std.out) \
(executable=/usr/ngs/MYAPP)'
```

All sites with myapp installed will provide a similar script at /usr/ngs/MYAPP which is customised to the site setups of app location and environment variables, so the location of the 'executable' appears to be the same for all users at all sites even though the installation may be completely different at each site.

The application is published to the NGS via the site

GlueHostApplicationSoftwareRunTimeEnvironment GRIS/BDII database entries. These are automatically generated by the ngs-uee-gip-plugin script (from here <http://forge.nesc.ac.uk/projects/ngs/>) which is installed by default by the NGS vdt installer script for a new site. Any entry in the /usr/ngs directory on a site will appear automatically as "NGS-UEE-filename" in the GRIS and BDII and can then be found by the WMS (see the Requirements line in the above example) and by the scripts that generate <http://www.ngs.ac.uk/applications>

The `/usr/ngs/MYAPP` is usually a symbolic link to `/usr/ngs/MYAPP_2_1` ie a version specific version of the script indicating the default application version for the site. WMS users can choose any site that has any version of MYAPP installed or be more specific about the version required for their job. A published application must have the default link and the version specific script even if only one version is available. If sites have similar scripting elsewhere in their file systems, the entries in `/usr/ngs` can all be symbolic links to those locations.

Note that `/usr/ngs` must be available on the head node and all slave/worker nodes at the site either by replication or by shared directories.

For the UEE mechanism to be uniform over all nodes on the NGS the UEE script names , version names and API must be identical at all sites, even if the content of the UEE scripts are different. These definitions are currently held at https://www.ngs.ac.uk/ops/privops/ngs2_config/UEEdocs.html and sites installing applications that are not previously available on the NGS should update this page appropriately. The script names must be in uppercase and the first ‘_’ designates the start of the version string. Version strings eg 2.3.1 should replace ‘.’ By ‘_’. This convention is documented in the .pdf and .ppt files at the start of this section.

The UEE script API is usually made to behave like the normally command line application, so that if an application is launched on the command line as:

```
myapp -i inputfile -o outputfile
```

the equivalent UEE is

```
/usr/ngs/MYAPP -i inputfile -o outputfile
```

so that the users have to make very little change to their command submission from local to grid usage.

However some applications have more than one executable. For example the Gromacs package contains 89 executables. It is not feasible or desirable to provide a UEE entry for each executable, so the UEE API for ‘package’ applications is normally set up so that the first argument is the name of normal binary eg:

```
/usr/ngs/MYAPP bin1.exe -i input -o output
```

where `bin1.exe` is the name of the binary from the package. Again this is to make the grid command line as similar as possible to a users experience.

The UEE scripts are also used to wrap MPI applications. Note that the ‘mpirun’ ‘mpiexe’ is not required in the UEE script. This is provided by the globus jobmanager (or WMS job wrapper), so that the equivalent MPI launch command line will be:

```
mpirun {-n 16} {-machinefile hostnames.txt} /usr/ngs/MYMPIAPP -i in -o out
```

5. INSTALLING NEW SOFTWARE FOR THE NGS

If a site would like to install an application locally that is already available on the NGS elsewhere, there are often simple installer scripts available for that application that capture compiler arguments and installation flows for other libraries as well as

download locations for the files. To check if such as script exists, sites should mail ngs-operations@jiscmail.ac.uk or the NGS helpdesk (support@grid-support.ac.uk).

If a site plans to install an application that is not available on the NGS there is a recommended flow and checklist available here:

https://www.ngs.ac.uk/ops/privops/ngs2_config/ngs2-apps-install-flow.html
that may be useful.

Note that when the application is documented, this should be added to the website <http://www.ngs.ac.uk/applications/category/appname> and be as generic as possible, so that it documents how an application is submitted to the NGS in general. It can obviously mention the lead installing site and site specific quirks (ideally none !) but the intent is that users should be able to run the application via the WMS or the NGS portal transparently without having to worry where the job will run.

Any software that has specific licensing requirements should be protected via unix groups or by other mechanism, so that users are forced to register via the NGS helpdesk to gain access to the software. Examples of license restrictions can range from agreeing to use a specific citation from the code web site to ensuring that they have a signed license from the code authors. The application documentation on the NGS web site should always state the terms of any license even if it is licensed under GPL or similar, and state how users register to use the code (if they need to).

A blank pro-forma application documentation page for the NGS web site can be found at <http://www.ngs.ac.uk/applications/blanktemplate>

6. JOINING THE NGS BIOINFORMATICS DATABASE MIRRORS

Many bioinformatics applications require access to preformatted copies of the various DNA and protein sequence databases (for example BLAST , Gromacs etc). The NGS has setup a mechanism where these databases are downloaded and formatted daily from EBI on one site on the NGS and the other sites rsync to this copy. This is much quicker than each site downloading and processing their own copies, reduces the load on EBI and ensures that all NGS sites have databases in sync with one another, allowing cross site searches to give consistent results.

Sites wishing to use bioinformatics applications are welcome to join this database mirroring scheme. Please mail support@grid-support.ac.uk for more details. See here for the range of databases on offer in this scheme:

<http://www.ngs.ac.uk/applications/bioinformatics/>