

User Guide for the Monash Sun Grid *[Revised: 20080707]*

Introduction

The Monash Sun Grid can be accessed directly by logging into `hn3.its.monash.edu.au` using `ssh` or an equivalent tool like `Putty`. If you don't have a login, you can use Globus client tools to execute jobs on it.

It should be noted that the main role of this machine is the submission of jobs to the execute nodes. Users should not attempt to execute on it any job which will occupy more than 500M of memory or run for more than a few minutes.

Using 'qsub' to Submit Jobs

The `qsub` command can be used as follows to submit a short simple job to the first available execute node thus:

```
cat >simple.sh
#!/bin/sh
#$ -S /bin/sh
hostname
Ctrl-D
qsub simple.sh
```

The standard output and standard error files will be written in the user's home directory. You can learn more about using `qsub` from the “Sun N1 Grid Engine 6.1 User's Guide” which can be found by hitting the “Documentation” link at: <http://www.sun.com/software/gridware/>

Environment Modules

There may be occasions where you need to use different compilers and/or libraries from those found in your usual environment, and you therefore need to adjust your environment variables accordingly. The `module` command makes this easy. Some examples of its use are:

```
module avail .. shows what modules are available on the system
module whatis .. shows what they do
module load openmpi-intel .. makes Intel MPI libraries and Intel Fortran compiler available
module list .. shows what module are loaded
module purge .. unloads all of them
module unload openmpi-intel intel .. unloads Intel MPI and Intel Fortran modules
```

Here's an MPI program which computes the value of Pi by calculating on multiple execute nodes the areas of thin rectangles inside a semi-circle:

```
#!/bin/sh
#$ -S /bin/sh
#$ -pe mpich 3
#$ -cwd
module load openmpi
mpirun pi3a 9999
```

You can find some `mpi` usage examples at <http://www-unix.mcs.anl.gov/mpi/usingmpi/>

Execute Nodes on the Monash Sun Grid

You can show the current status of all MSG jobs thus: `qstat -u *`

To see which nodes you are currently using: `qstat -g t`

If your job will run for more than 60 minutes, you need to declare a maximum runtime so that it will be allocated to execute slts having that capability thus: `qsub -l h_rt=1:00:00 1HourJob.sh`

Likewise, if your job will require more than 3900M of virtual memory, you need to declare that as follows: `qsub -l h_vmem=4G 4GigJob.sh`

For a concise summary of MSG status: `Qstat`

If you have X11 access, you may be able to use the `qmon` GUI tool to view MSG status and job/node attributes.

Linux and Solaris Jobs

The Monash Sun Grid has both Linux and Solaris execute nodes. Jobs will, by default, be submitted to Linux nodes. To force execution under Solaris, do: `qsub -l "arch=sol*" SolJob.sh`

If your job can execute on either type of node, do: `qsub -l "arch=*" JavaJob.sh`

To show which Solaris modules are available:

```
echo module whatis | qsub -l "arch=sol*" -cwd -j y
```

Using NAMD

NAMD is a popular molecular dynamics program. A parallel version of it is available on the MSG. A simple submit script for executing a copy of the 'alanin' example which comes with it is as follows. It uses the 'orte' parallel environment in place of the similar 'mpich' one so as to avail itself of the tighter SGE integration capabilities available in recent implementations of OpenMPI.

```
#!/bin/sh
#$ -S /bin/sh
#$ -pe orte 4
#$ -cwd
module load namd
mpirun namd2 ../work/alanin
```

Using the GNU Scientific Library

The GNU Scientific Library is a numerical library for C and C++ programmers. A simple matrix multiplication program, and an associated submit script are as follows.

```
/* module load gsl
   gcc -o ../bin/Matmult $CFLAGS $LDFLAGS -lgsl -lgslcblas -lm BLAS.c
*/
#include <stdio.h>
#include <gsl/gsl_blas.h>
int main (void) {
    double a[] = { 0.11, 0.12, 0.13,
                  0.21, 0.22, 0.23 };
    double b[] = { 1011, 1012,
                  1021, 1022,
```

```

        1031, 1032 };
double c[] = { 0.00, 0.00,
              0.00, 0.00 };
gsl_matrix_view A = gsl_matrix_view_array(a, 2, 3);
gsl_matrix_view B = gsl_matrix_view_array(b, 3, 2);
gsl_matrix_view C = gsl_matrix_view_array(c, 2, 2);

/* Compute C = A B and print result */
gsl_blas_dgemm (CblasNoTrans, CblasNoTrans,
               1.0, &A.matrix, &B.matrix,
               0.0, &C.matrix);
printf ("[ %g, %g\n", c[0], c[1]);
printf (" %g, %g ]\n", c[2], c[3]);
return 0;
}

#!/bin/sh
#$ -S /bin/sh
#$ -cwd
module load gsl
~/bin/Matmult

```

Using the Atlas Linear Algebra Suite

The Atlas Linear Algebra Suite contains BLAS and LAPACK routines, and can be used in programs which start as shown in the following program excerpt:

```

/* Double Precision Matrix Multiplication using Atlas CBLAS
   module load atlas
   gcc -o MatMula -O3 -I$INCLUDE -L$LIBRARY_PATH \
       -lcblas -latlas MatMula.c
*/
#include <stdio.h>
#include <stdlib.h>
#include <cblas.h>plots.ps

```

Using R

R is a software environment for statistical computing and graphics. A simple R program and an associated submit script are as follows. The output in this instance appears in a PostScript file named: Rplots.ps

```

#!/usr/bin/env Rscript
# PlotCos.r      R program to plot cosine values.
seq(-10, 10, by=.1) -> x
plot (x, cos(x), type="l")

#!/bin/sh
#$ -S /bin/sh
#$ -cwd
module load R
../src/PlotCos.r

```

Using Octave

GNU Octave is a high-level language primarily intended for numerical computations. Its commands are in general compatible with those of Matlab. A simple Octave program and an associated submit script are as follows:

```
#!/usr/bin/env octave
# montepi.m Monte Carlo computation of Pi
function mypi = pimontecarlo(n)
x = rand(n,1); y = rand(n,1); # Generate random points inside a square,
z = x.^2 + y.^2;             # count how many of them lie with a circle.
v = (z <= 1) ;
m = sum(v) ; mypi = 4*m/n; # Note: this function can reside in a
end                          # separate file called: pimontecarlo.m
if nargin < 1
    printf ('Usage: %s iterations [iterations .. iterations]\n', program_name);
end
for i = 1:nargin
    iterations = nth (argv, i)
    result = pimontecarlo(str2num(iterations))
    printf ('\n');
end

#!/bin/sh
#$ -S /bin/sh
#$ -cwd
module load octave
../src/montepi.m 1000 2000 5000
```

An Octave MPI toolbox is also available. The following script illustrates how it can be used to run one of the examples which comes with it:

```
#!/bin/sh
#$ -S /bin/sh
#$ -cwd
#$ -pe mpich 4
module load mpitb
mpirun -wd $MPITB_HOME/Pi octave -q --eval "Pi(2E7,'r')"
```

Using MrBayes

MrBayes is a program which performs Bayesian inference of phylogeny. A simple 4-slot MrBayes submit script is as follows; it uses the `primates.nex` sample script which you can copy from the MrBayes installation directory.

```
#!/bin/sh
#$ -S /bin/sh
#$ -pe orte 4
#$ -cwd
module load mrbayes
mpirun mb <<EOF
set autoclose=yes nowarn=yes
```

```

execute primates.nex
lset nst=6 rates=gamma
mcmc nruns=1 ngen=10000 samplefreq=10 file=primates.nex1
mcmc file=primates.nex2
mcmc file=primates.nex3
sump burnin=250
sumt burnin=250
quit
EOF

```

Using the Intel Cluster Maths Kernel Library

The Intel Cluster Maths Kernel Library provides optimised implementations of mathematics routines for engineering, scientific and financial programs. The example hereunder shows how it can be used to compile and execute an eigenvalue sample program provided with that library.

```

#!/bin/sh
#$ -S /bin/sh
#$ -cwd
module load intel cmkl
cd /opt/sw/intel/cmkl/9.1.023/examples/lapack/source
ifort -o $HOME/work/ssterfx.$$ -L$LD_LIBRARY_PATH -lmkl_lapack -lmkl -lpthread -lm ssterfx.f
$HOME/work/ssterfx.$$ </opt/sw/intel/cmkl/9.1.023/examples/lapack/data/ssterfx.d
rm -f $HOME/work/ssterfx.$$

```

Installing Globus Client Tools on your Workstation

You can install the VDT Globus client tools on your Scientific Linux 5 or equivalent (e.g. CentOS 5) workstation as shown in the following paragraphs. Note that a gridftp server will be set up on your workstation so that data can be acquired from and returned to it by Globus jobs.

- Request and install on your workstation a host certificate from an IGTF-Accredited Certificate Authority. This is most easily done from a machine where Globus client tools are already installed, as detailed at <http://wiki.arcs.org.au/bin/view/APACgrid/HostCertRequestAPAC>

- Ensure that you don't have a 'globus' user.

- As the root user, do:

```

cd /etc/yum.repos.d &&
wget http://grid.its.monash.edu.au/dist/packages/sl5/msg-sl5-prod.repo

```

- then: yum install MSGglobus

- and: /usr/local/sbin/MsgRcVdt181.sh

- finally .. ensure that the DN associated with your personal certificate appears with an appropriate username in: /etc/grid-security/grid-mapfile

If you don't have Globus Client Tools on your Workstation

You can submit command-line Globus jobs from any host on which a host certificate from an IGTF-

Accredited CA has been installed. You'll also need to ensure that the Distinguished name (DN) on your personal certificate appears in the host's grid-mapfile (or that the host makes PRIMA callouts to a GUMS server to determine a suitable username under which your jobs will be executed); you may need to contact the host's administrator(s) for this to happen.

Obtaining a Personal Certificate

A personal certificate is used to identify you when you use Globus client tools to submit a job to a host which may be at another site. Such a host will need to have a trust-relationship with the issuer of your personal certificate. It is therefore recommended that you obtain your personal certificate from a Certificate Authority (CA) accredited by the International Grid Trust Federation (IGTF).

One such IGTF-accredited CA is that operated by the APAC National Grid. Some instructions requesting an APAC Grid certificate can be found at: <http://wiki.arcs.org.au/bin/view/Main/GridCertificates>

Simple Globus Jobs

You should ensure that you have a personal key and its associated certificate issued by an IGTF-recognised CA installed on your workstation or submit-host in your `~/.globus` directory (as `userkey.pem` and `usercert.pem`), and that the directory `~/.globus/scratch` exists. Also ensure that your fully-qualified-name appears in a grid-mapfile on both your workstation and the head-node through which you will be submitting jobs. Then perform the following steps (assuming that you are submitting to `hn3.its.monash.edu.au`):

- `./opt/vdt/setup.sh ..` establishes the environment so that Globus client tools can be found. On some machines (e.g. `hn3.its.monash.edu.au`) `vdt` has been defined as a system-wide alias for this, so you need only enter: `vdt`
- `grid-proxy-init ..` creates a proxy valid (by default) for 12 hours.
- `globusrun -a -r hn3.its.monash.edu.au ..` performs an authentication test.
- `globus-job-run hn3.its.monash.edu.au /bin/uname -a ..` submits a GRAM Fork job and awaits return of output.
- `globus-job-run hn3.its.monash.edu.au/jobmanager-sge /bin/hostname ..` submits a GRAM SGE job and awaits return of output.
- `globusrun-ws -submit -s -S -F hn3 -Ft Fork -c /bin/mkdir -pv .globus/scratch ..` submits a WS-GRAM Fork job and awaits return of output.
- `globusrun-ws -submit -s -S -F hn3 -Ft SGE -c /bin/hostname ..` submits a WS-GRAM SGE job and awaits return of output.

More Complex Globus Jobs

You can copy to your scratch directory a Fortran program which calculates π using MPI, then compile and execute it as follows:

- `globus-url-copy gsiftp://gridftp.its.monash.edu.au/opt/MSG/Demo/pi3a.f \`
`file://$HOME/.globus/scratch/pi3a.f`
- `globusrun-ws -submit -s -S -F hn3.its.monash.edu.au -Ft SGE -f gt4-f77.rsl ..`

where `gt4-f77.rsl` is a file containing:

```
<job>
  <executable>/usr/bin/mpif77</executable>
  <directory>${GLOBUS_SCRATCH_DIR}</directory>
  <argument>pi3a.f</argument>
  <jobType>single</jobType>
  <extensions>
    <module>openmpi</module>
  </extensions>
</job>
```

- `globusrun-ws -submit -s -S -F hn3.its.monash.edu.au -Ft SGE -f gt4-pi3a.rsl ..`
where `gt4-pi3a.rsl` is a file containing:

```
<job>
  <executable>${GLOBUS_SCRATCH_DIR}/a.out</executable>
  <argument>120</argument>
  <count>2</count>
  <jobType>mpi</jobType>
  <extensions>
    <module>openmpi</module>
  </extensions>
</job>
```

When there are several users whose Certificates are mapped to a common `userid`, there is a danger that different users will be using the same file names at the same time. You can get a program which demonstrates some mechanisms to circumvent this as follows:

```
globus-url-copy gsiftp://gridftp.its.monash.edu.au/opt/MSG/Demo/Gpicalc \
  file:///tmp/
```

- **Then:** `chmod a+rx /tmp/Gpicalc` **And:** `/tmp/Gpicalc`
- You can copy all the files in the `Demo` directory thus:

```
globus-url-copy -r -cd gsiftp://gridftp.its.monash.edu.au/opt/MSG/Demo/ \
  file:///tmp/
```

- Amongst those files you'll see: `GpicalcI`. This is an Intel-compiler version of the above; it uses a tag to set an appropriate environment.
- You also find: `Ggzip` (demonstrating WS file staging) and: `Glzma` (demonstrating non-WS procedures).
- There's also: `Gbzip2` (which uses `mpibzip2` on multiple nodes for fast compression).

Globus Jobs on Solaris Nodes

The following example shows how execution on a Solaris slot might be requested:

```
<!-- Usage: globusrun-ws -submit -s -S -F hn3 -Ft SGE -f gt4-hostname.rsl -->
<job>
```

```
<executable>/bin/hostname</executable>  
<maxWallTime>1</maxWallTime>  
<jobType>single</jobType>  
<extensions>  
  <OSFamily>solaris</OSFamily>  
</extensions>  
</job>
```

If your job can execute on either a Solaris slot or a Linux slot, you should include both operating-system names in the 'extensions' field thus:

```
<extensions>  
  <OSFamily>solaris</OSFamily>  
  <OSFamily>linux</OSFamily>  
</extensions>
```