

MPI on ARCHER



Documentation

- See <https://www.archer.ac.uk/documentation/user-guide/>
 - Accessing the service
 - Resource Allocation and Job Execution



Access

- SSH access: `ssh -X login.archer.ac.uk`
 - flag `-X` ensures graphics are sent back to your workstation/laptop
- Using ssh
 - Trivial for Linux (open a terminal)
 - Mac (open a terminal)
 - local X server must be enabled to display any graphics
 - Windows
 - require an ssh-client, e.g. putty
 - <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>
 - select SSH -> X11 -> “Enable X11 forwarding”
 - require an X server, e.g. xming
 - <http://sourceforge.net/projects/xming/>



Setup

- Take a copy of `MPP-templates.tar`

```
wget http://archer.ac.uk/training/course-  
material/2016/11/mpi_Imperial/Exercises/MPP-  
templates.tar
```

- unpack: `tar -xvf MPP-templates.tar`



Compilers on ARCHER

- ARCHER has 3 compilers available
 - Intel
 - GNU
 - Cray
- Cray compiler is the default when logging on
- Software on ARCHER is controlled using modules
 - the GNU “modules” framework to support multiple software versions and to create integrated software packages



Default modules example

```
adrianj@eslogin001:~> module list
Currently Loaded Modulefiles:
 1) modules/3.2.6.7
 2) nodestat/2.2-1.0500.41375.1.85.ari
 3) sdb/1.0-1.0500.43793.6.11.ari
 4) alps/5.0.3-2.0500.8095.1.1.ari
 5) MySQL/5.0.64-1.0000.7096.23.1
 6) lustre-cray_ari_s/2.3_3.0.58_0.6.6.1_1.0500.7272.12.1-1.0500.44935.7.1
 7) udreg/2.3.2-1.0500.6756.2.10.ari
 8) ugni/5.0-1.0500.0.3.306.ari
 9) gni-headers/3.0-1.0500.7161.11.4.ari
10) dmapp/6.0.1-1.0500.7263.9.31.ari
11) xpmem/0.1-2.0500.41356.1.11.ari
12) hss-llm/7.0.0
13) Base-opts/1.0.2-1.0500.41324.1.5.ari
14) craype-network-aries
15) craype/1.06.05
16) cce/8.2.0.181
...
```



Viewing available modules

- There are hundreds of possible modules available to users.
 - Beyond the pre-loaded defaults there are many additional packages provided by Cray
- Users can see all the modules that can be loaded using the command:
 - `module avail`
- Searches can be narrowed by passing the first few characters of the desired module, e.g.

```
adrianj@eslogin001 :~> module avail gc
```

```
----- /opt/modulefiles -----  
--  
gcc/4.6.1          gcc/4.7.2          gcc/4.8.0  
gcc/4.6.3          gcc/4.7.3          gcc/4.8.1(default)
```



Modifying the default environment

- Loading, swapping or unloading modules:
 - The default version of any individual modules can be loaded by name
 - e.g.: `module load perftools`
 - A specific version can be specified after the forward slash.
 - e.g.: `module load perftools/6.1.0`
 - Modules can be swapped out in place
 - e.g.: `module swap intel intel/13.1.1.163`
 - Or removed entirely
 - e.g.: `module unload perftools`
- Modules will automatically change values of variables like PATH, MANPATH, LM_LICENSE_FILE... etc
 - Modules also provide a simple mechanism for updating certain environment variables, such as PATH, MANPATH, and LD_LIBRARY_PATH
 - In general, you should make use of the modules system rather than embedding specific directory paths into your startup files, makefiles, and scripts




```
adrianj@eslogin008:~> module show fftw
```

```
-----  
/opt/cray/modulefiles/fftw/3.3.0.4:
```

```
setenv          FFTW_VERSION 3.3.0.4  
setenv          CRAY_FFTW_VERSION 3.3.0.4  
setenv          FFTW_DIR /opt/fftw/3.3.0.4/sandybridge/lib  
setenv          FFTW_INC /opt/fftw/3.3.0.4/sandybridge/include  
prepend-path    PATH /opt/fftw/3.3.0.4/sandybridge/bin  
prepend-path    MANPATH /opt/fftw/3.3.0.4/share/man  
prepend-path    CRAY_LD_LIBRARY_PATH /opt/fftw/3.3.0.4/sandybridge/lib  
setenv          PE_FFTW_REQUIRED_PRODUCTS PE_MPICH  
prepend-path    PE_PKGCONFIG_PRODUCTS PE_FFTW  
setenv          PE_FFTW_TARGET_interlagos interlagos  
setenv          PE_FFTW_TARGET_sandybridge sandybridge  
setenv          PE_FFTW_TARGET_x86_64 x86_64  
setenv          PE_FFTW_VOLATILE_PKGCONFIG_PATH  
/opt/fftw/3.3.0.4/@PE_FFTW_TARGET@/lib/pkgconfig  
prepend-path    PE_PKGCONFIG_LIBS  
fftw3f_mpi:fftw3f_threads:fftw3f:fftw3_mpi:fftw3_threads:fftw3  
module-whatism FFTW 3.3.0.4 - Fastest Fourier Transform in the West  
-----
```



Compilers on ARCHER

- All applications that will run in parallel on the Cray XC should be compiled with the standard language wrappers.

The compiler drivers for each language are:

- `cc` – wrapper around the C compiler
 - `CC` – wrapper around the C++ compiler
 - `ftn` – wrapper around the Fortran compiler
- These scripts will choose the required compiler version, target architecture options, scientific libraries and their include files automatically from the module environment.
- Use them exactly like you would the original compiler, e.g. To compile `prog1.f90` run

```
ftn -c prog1.f90
```



Compilers on ARCHER

- The scripts choose which compiler to use from the PrgEnv module loaded

PrgEnv	Description	Real Compilers
PrgEnv-cray	Cray Compilation Environment	crayftn, craycc, crayCC
PrgEnv-intel	Intel Composer Suite	ifort, icc, icpc
PrgEnv-gnu	GNU Compiler Collection	gfortran, gcc, g++

- Use module swap to change PrgEnv, e.g.
 - `module swap PrgEnv-cray PrgEnv-intel`
- PrgEnv-cray is loaded by default at login. This may differ on other Cray systems.
 - use `module list` to check what is currently loaded
- The Cray MPI module is loaded by default (`cray-mpich`).
 - To support SHMEM load the `cray-shmem` module.
- The drivers automatically support an MPI build
 - No need to use specific wrappers such as `mpiifort`, `mpicc`



Compiling MPI Programs ARCHER

- Fortran programmers use ftn
- C programmers use cc (CC for C++)
- There is nothing magic about these compilers!
 - simply wrappers which automatically include various libraries etc
 - compilation done by standard compilers
 - ifort and icc
 - gfortran and gcc
 - craycc and crayftn
- You can use the supplied Makefiles for convenience
 - make -f Makefile_c
 - make -f Makefile_f90
- Easiest to make a copy of one of these called “Makefile”
 - also need to change the line “MF=” in the Makefile itself



Running interactively

- All jobs on ARCHER must be run through the batch system
 - This controls resource allocation and usage
 - Ensures fair access, charges usage against budgets, etc...
- General batch jobs are run for you
 - No access to the running job, cannot see what is happening until the job finishes.
- It is possible to do interactive runs so you can run the MPI program yourself (although you still don't get access to the compute nodes)
 - To submit a interactive job reserving 8 nodes (192 cores) for 1 hour you would use the following qsub command:

```
qsub -IVl select=8,walltime=1:0:0 -A [project code]
```

- When you submit this job your terminal will display something like:

```
qsub: waiting for job 492383.sdb to start
```



Running on ARCHER

- Run via a batch system
 - ARCHER uses PBS (Portable Batch System)
 - submit a script that then launches your program
- In MPP-templates/ is a standard batch script: mpibatch.pbs
 - make a copy of this file with a name that matches your executable, eg
 - `user@eslogin003$ cp mpibatch.pbs hello.pbs`
- For 4 processors:

```
qsub -q resnum -l select=1:mpiprocs=4 hello.pbs
```

 - automatically runs executable called “hello”
 - resnum should be replaced with the id of the reservation we are using
 - output will appear in a file called `hello.pbs.oXXXXX`
 - can follow job progress using `qstat -u $USER`
 - script also times your program using the Unix “time” command
 - full instructions included as comments in the template
 - no need to alter the script - just rename it as appropriate
 - eg to run a program “pingpong” make another copy called “pingpong.pbs”



Filesystems on ARCHER

- ARCHER has 3 file systems:
 - home – NFS, not accessible on compute nodes
 - For source code and critical files
 - Backed up
 - > 200 TB total
 - /work – Lustre, accessible on all nodes
 - High-performance parallel filesystem
 - Not backed-up
 - > 4PB total
 - RDF – GPFS, not accessible on compute nodes
 - Long term data storage



Filesystems on ARCHER

- Cannot run from the home file system
 - back-end nodes can only see the work file system
 - /home/y14/y14/guestXX/
- Recommendation
 - do everything in /work
 - change directory to /work/y14/y14/guestXX/
 - Copy and compile code there, submit jobs from there



C++ Interface

- MPI is not an OO interface
 - however, can be called from C++
- Function calls are different, eg:
 - `MPI::Intracomm comm;`
 - `...`
 - `MPI::Init();`
 - `comm = MPI::COMM_WORLD;`
 - `rank = comm.Get_rank();`
 - `size = comm.Get_size();`
- Compiler is called CC
 - see `hello.cc` and `Makefile_cc`



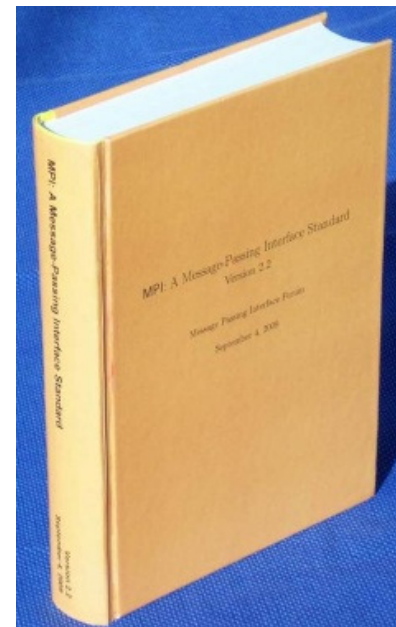
C++ interface is
now deprecated

Advised to cross-
call to C

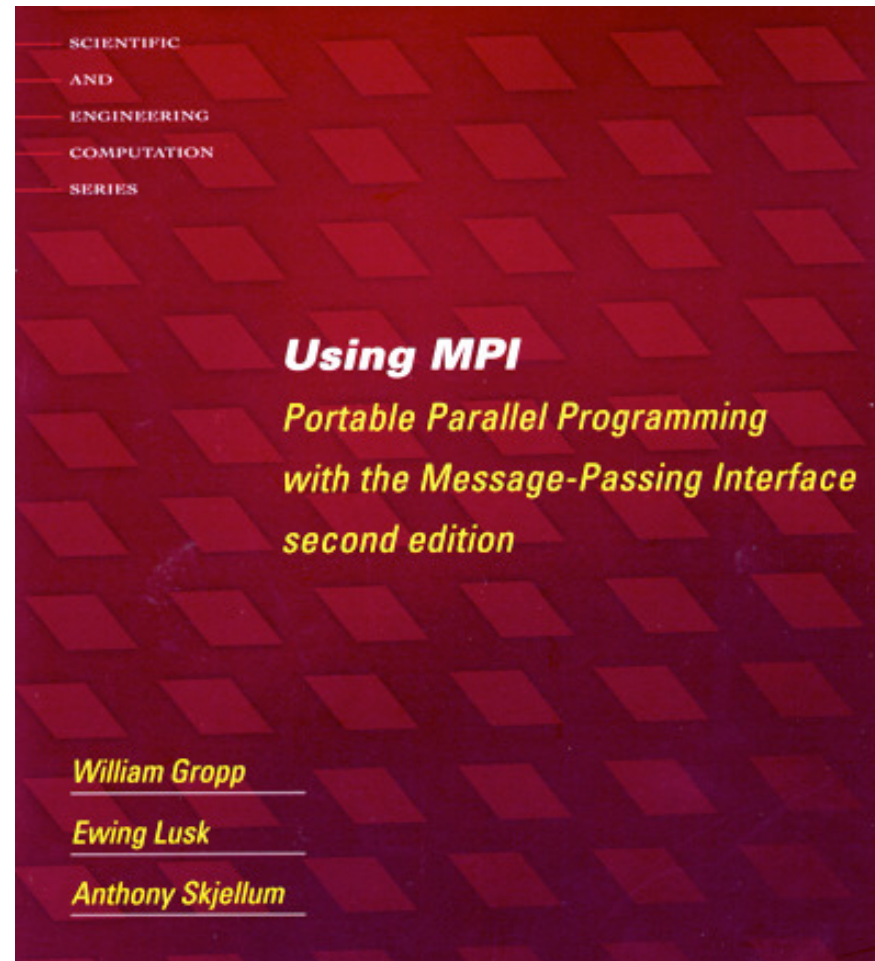


Documentation

- MPI Standard available online
 - See: <http://www.mpi-forum.org/docs/>
- Available in printed form
 - <https://fs.hlrs.de/projects/par/mpi//mpi30/>
- Man pages
 - must use the C style of naming: `man MPI_Routine_name`, eg:
 - `user@eslogin003$ man MPI_Init`



MPI Books



Exercise: Hello World

The minimal MPI program

- See Exercise 1 on the exercise sheet
- Write an MPI program that prints a message to the screen
- Main purpose is to get you compiling and running parallel programs on ARCHER
 - also illustrates the SPMD model and use of basic MPI calls
- We supply some very basic template code
 - see pages 4 and 5 of the notes as well
 - Password for ARCHER:

