epcc

MPI on morar and ARCHER

Access

- morar available directly from CP-Lab machines
- external access to morar:

  gateway: **`ssh -Y user@ph-cplab.ph.ed.ac.uk`**

  then: **`ssh -Y cplabXXX`** (pick your favourite machine)
- external access to ARCHER:

  `ssh -Y user@login.archer.ac.uk`
- You can access systems using ssh from anywhere
  - Trivial for Linux
  - Mac
    - enable the X server (xquartz) to display any graphics
  - Windows
    - need to install an X server program, eg xming (which is free!)

Useful files and templates

- Take a copy of `MPP-templates.tar`
  - see the course web pages
- unpack: `tar xvf MPP-templates.tar`

Compiling MPI Programs on morar

- Fortran programmers use mpif90
- C programmers use mpicc
- There is nothing magic about these MPI compilers!
  - simply wrappers which automatically include various libraries etc
  - compilation done by standard (Portland Group) compilers
    - pgf90 and pgcc
- 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

- Timings will not be reliable
  - shared with other users, many more processes than processors
  - but **very useful** during development and for debugging
- mpiexec –n 4 ./mpiprog.exe
  - runs your code on 4 processes
- NOTE
  - output might be buffered
  - if your program crashes, you may see no output at all
- May need to explicitly flush prints to screen
  - FLUSH(6)
  - fflush(stdout);

Running on morar

- Run via a batch system
  - on morar we use Sun Grid Engine (SGE)
  - submit a script that then launches your program

- In MPP-templates/ is a standard batch script: mpibatch.sge
  - make a copy of this file with a name that matches your executable, eg
  - **`user@cplab$ cp mpibatch.sge hello.sge`**

- To run on 4 processors: **`qsub -pe mpi 4 hello.sge`**
  - automatically runs executable called “hello”
  - output will appear in a file called **`hello.sge.oXXXXX`**
  - can follow job progress using **`qmon`** GUI or **`qstat`** or **`qstat -u "*"`**
  - 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.sge”

Morar idiosyncracies

- By default, MPI wrappers are not in your path

```
user@cplab$ mpicc
-bash: mpicc: command not found
```

- To access correct version: `module load PrgEnv-pgi`
  - add this to end of your `.bash_profile` file in home directory
  - to check you have the right version (similarly for mpif90)

```
user@cplab$ which mpicc
/opt/pgi/linux86-64/2015/mpi/mpich/bin/mpicc
```

Compiling MPI Programs on ARCHER

- Fortran programmers use ftn
- C programmers use cc
- There is nothing magic about these MPI compilers!
  - simply wrappers which automatically include various libraries etc
  - compilation done by standard (Cray) compilers
    - crayftn and craycc
- 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

ARCHER idiosyncracies

- Not possible to run directly on front-end
- Can be a substantial delay in batch queues
  - we may sometimes have dedicated queues for the course
  - instant turnaround!
- Cannot run from the home file system
  - back-end nodes can only see the work file system
- Recommendation
  - do everything in /work/
  - change directory to /work/y14/y14/guestXX/

Running on ARCHER back-end

- Run via a batch system
  - on ARCHER we use the Portable Batch System (PBS)
  - 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@archer$ cp mpibatch.pbs hello.pbs`**

- Submit: **`qsub -q <reserved queue ID> hello.pbs`**
  - we have a reserved queue **RXXXXXX** for the courses
  - you will need to alter **NPROCS** (the argument to “aprun”) by hand
  - … and **select** more than one node for more than 24 processes
  - output will appear in a file called **`hello.pbs.oXXXXX`**
  - can follow job progress using **`qstat`** command
  - script also times your program using the Unix “time” command
  - full instructions included as comments in the template

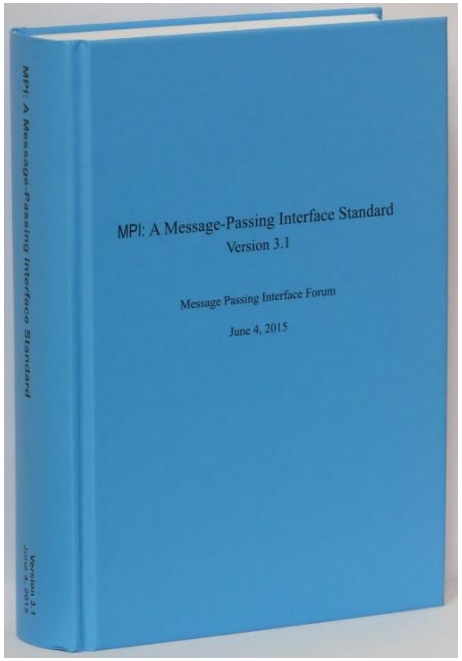
C++ Interface

- 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 mpicxx
  - see hello.cc and Makefile_cc

C++ interface is now deprecated

Advised to cross-call to C

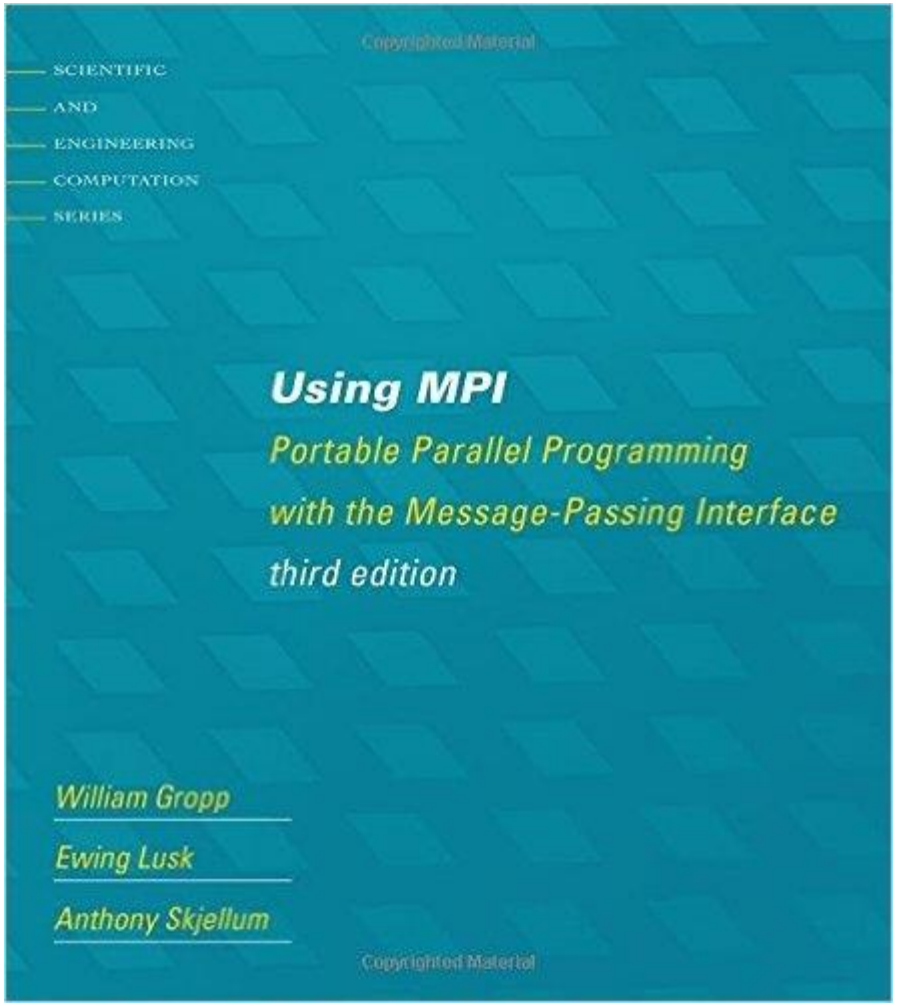
- MPI Standard available online
  - See: http://www.mpi-forum.org/docs/
  - currently version 3.1



- Available in printed form
  - http://www.hlrs.de/mpi/mpi31/

- Man pages available on CP-Lab and ARCHER
  - must use the C style of naming: man MPI_Routine_name, eg:
  - **`user@computer$ man MPI_Init`**

MPI Books


Copyrighted Material
SCIENTIFIC
AND
ENGINEERING
COMPUTATION
SERIES
Using MPI
Portable Parallel Programming
with the Message-Passing Interface
third edition
William Gropp
Ewing Lusk
Anthony Skjellum
Copyrighted Material

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 ness
  - 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