MPI on Cirrus and ARCHER

Reusing this material

Access

- ARCHER: **`ssh -XY user@login.archer.ac.uk`**
- Cirrus: **`ssh -XY user@login.cirrus.ac.uk`**
  - multiple login nodes, users assigned to different nodes depending on system load
  - login nodes shared by all users
- You can access EPCC systems using ssh from anywhere
  - Linux: terminal + command line
  - Mac: Mac terminal + command line + enable an X server (e.g. xquartz) to display any graphics
  - Windows: need to install ssh and an X-server, e.g. MobaXterm
- Full instructions on access & login in separate document on course web pages

Useful files and templates

- Take a copy of **`MPP-templates.tar`**
  - see the course web pages
- unpack: **`tar -xvf MPP-templates.tar`**
- Crib sheets for MPI programs available on course web pages

Setting up Cirrus environment

- Load the Intel Compilers
  - module load intel-compilers-19
- Load the Message-Passing Toolkit
  - module load mpt
- To automate, add these lines to your ".bash_profile" file

```
[user@cirrus] emacs -nw ~/.bash_profile
```

Compiling MPI Programs on Cirrus

- C programmers use: **`mpicc -cc=icc`**
- C++ programmers use: **`mpicxx -cxx=icpc`**
- Fortran programmers use: **`mpif90`**

- There is nothing magic about these MPI compilers!
  - simply wrappers which automatically include various libraries etc
  - compilation done by standard (e.g. Intel) compilers
    - icc, icpc and ifort

- You can use the supplied Makefiles for convenience
  - **`make -f Makefile_c`**
  - **`make -f Makefile_cc`**
  - **`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 on Cirrus

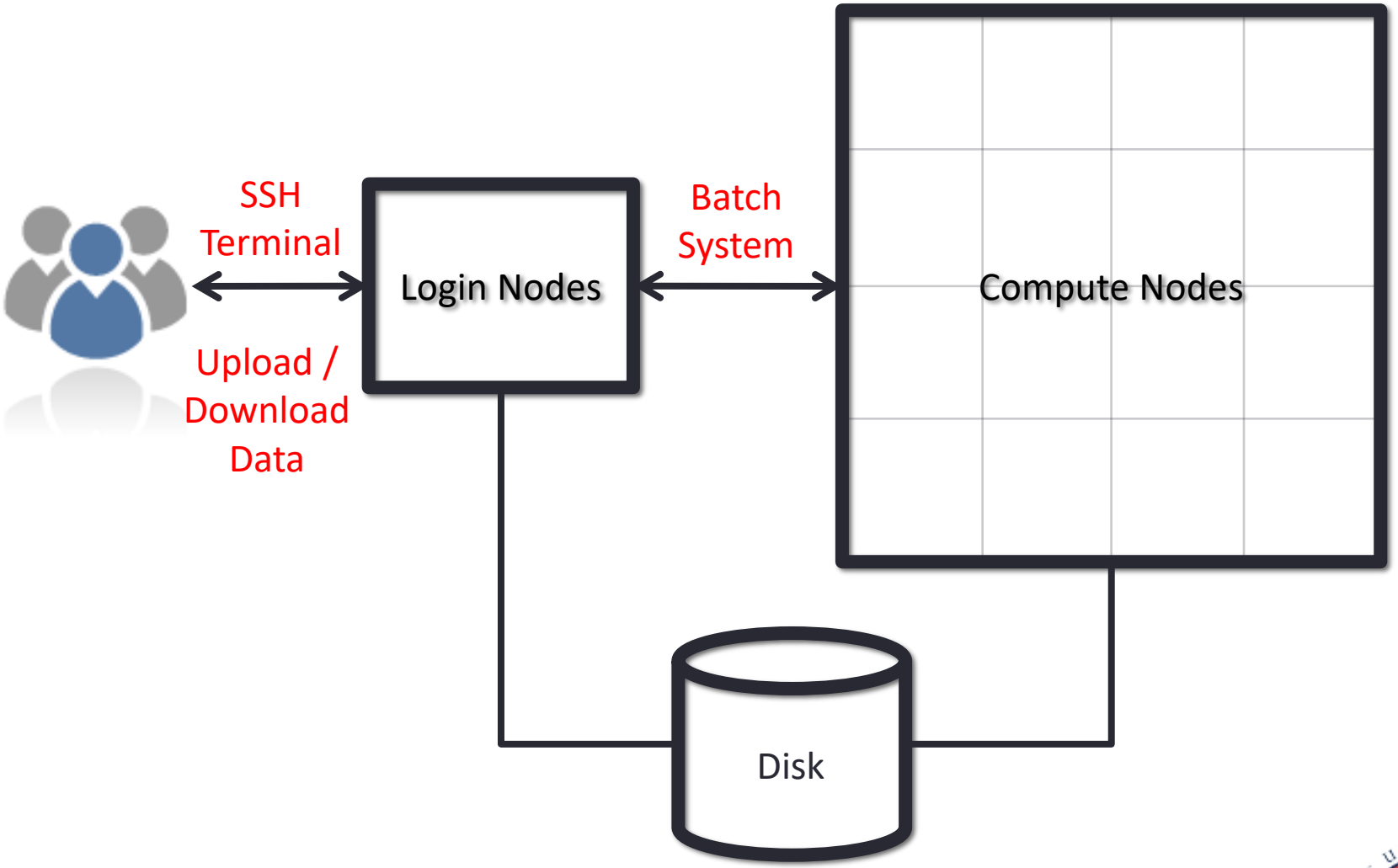
- Timings will not be reliable
  - shared with other users, many more processes than processors
  - but **very useful** during development and for debugging

- **`mpirun –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

- It *may* help to explicitly flush prints to screen
  - **`fflush(stdout);`**
  - **`FLUSH(6)`**

Typical HPC system layout



Running batch jobs on Cirrus

- Run via a batch system
  - Cirrus uses SLURM: you submit a batch script that launches your program
- In **`MPP-templates/`** is a standard batch script: **`cirrusmpi.job`**
  - set up to run a program called “hello” on 4 processors
- To run on 4 processors: **`sbatch cirrusmpi.job`**
  - runs executable called “hello”
  - output will appear in a file called **`hello-XXXX.out`**
  - errors will appear in a file called **`hello-XXXX.err`**
  - can follow job progress using **`squeue`** or **`squeue -u <user>`**
  - e.g. if your username is s1234567 use: **`squeue -u s1234567`**
  - full instructions included as comments in the template
- MSc students should alter charging: dc032 -> dc032-user
  - e.g. `#SBATCH --account=dc032-s1234567`

Cirrus idiosyncrasies

- By default, MPI wrappers are not in your path

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

- To access correct version: **`module load mpt`**
  - defaults to GNU compilers: gcc, g++ and gfortran
  - in SLURM batch system, job launcher is called **`srun`**

- Intel compilers: **`module load intel-compilers-19`**
  - can add these to end of your **`.bash_profile`** file in home directory
  - to check you have the right version (similarly for mpif90)

```
user@cirrus$ which mpicc
/opt/hpe/hpc/mpt/mpt-2.22/bin/mpicc
```

  - to use Intel C compiler: **`mpicc -cc=icc`**
  - to use Intel C++ compiler: **`mpicxx -cxx=icpc`**
  - to use Intel Fortran compiler: **`mpif90 -fc=ifort`**

Compiling MPI Programs on ARCHER

- C programmers use **`cc`**
- C++ programmers use **`CC`**
- Fortran programmers use **`ftn`**
- There is nothing magic about these MPI "compilers" !
  - simply wrappers which automatically include various libraries etc
  - compilation done by standard (Cray) compilers
    - **`craycc`**, **`crayc++`** and **`crayftn`**
- You can use the supplied Makefiles (C, C++, Fortran) for convenience
  - **`make -f Makefile_c`**
  - **`make -f Makefile_cc`**
  - **`make -f Makefile_f90`**
- Easiest to make a copy of your choice called "Makefile"
  - e.g. **`cp Makefile_c Makefile`**
    - also need to change the first line "MF=" in the Makefile itself
  - then you can just type "**`make`**"

ARCHER idiosyncrasies

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

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: **`archermpi.pbs`**
  - make a copy of this file with a name that matches your executable, e.g.
  - **`user@archer$ cp archermpi.pbs hello.pbs`**
- Submit: **`qsub -q RXXXXXX hello.pbs`**
  - 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
- If there is no reserved queue
  - **`qsub -q short hello.pbs`**
  - short queue is for small jobs less than 20 minutes during working hours

Compiling MPI Programs on NEXTGenIO

- C programmers use: **`mpicc`**
- C++ programmers use: **`mpicxx`**
- Fortran programmers use: **`mpif90`**

- There is nothing magic about these MPI compilers!
  - simply wrappers which automatically include various libraries etc
  - compilation done by standard (here, GNU) compilers
    - gcc, g++, gfortran

- You can use the supplied Makefiles for convenience
  - **`make -f Makefile_c`**
  - **`make -f Makefile_cc`**
  - **`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 on NEXTGenIO

- Timings will not be reliable
  - shared with other users, many more processes than processors
  - but **very useful** during development and for debugging

- **`mpirun –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

- It may help to explicitly flush prints to screen
  - **`fflush(stdout);`**
  - **`FLUSH(6)`**

C++ Interface

- MPI is not an OO interface
  - however, can be called from C++
- Originally had different function calls, e.g.
  - `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 removed

Must therefore cross-call to C

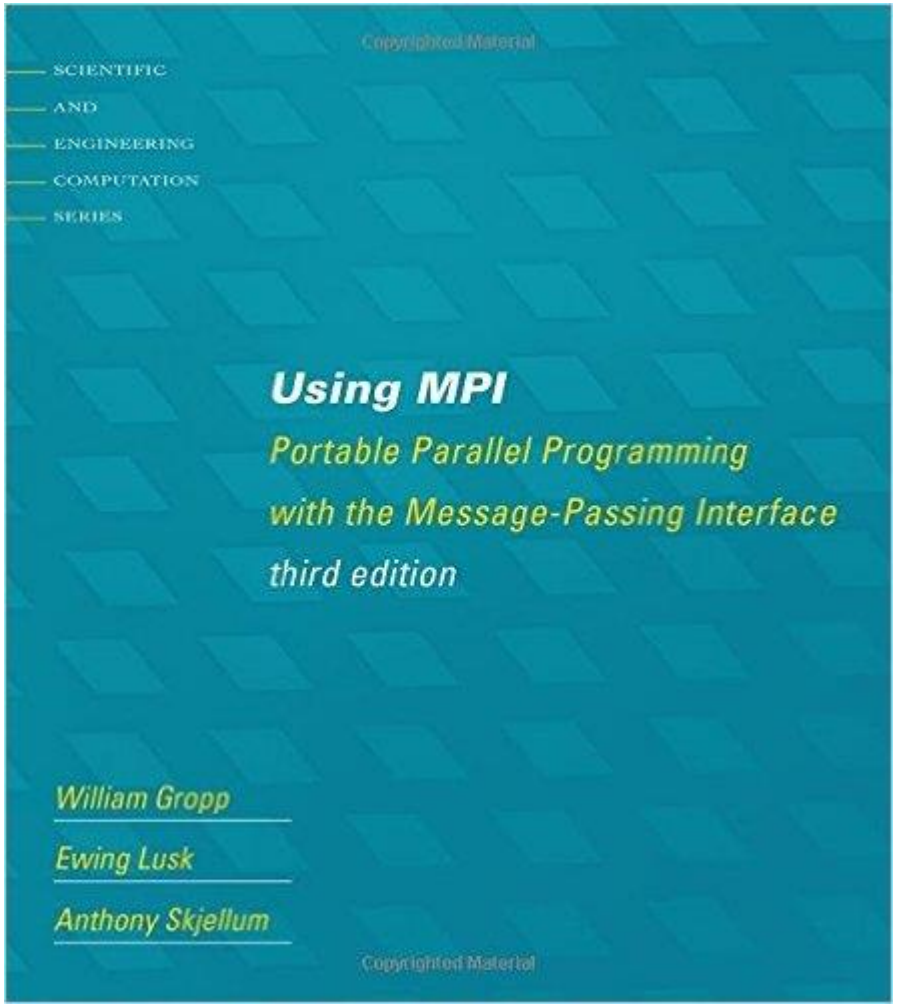
Documentation

- 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 Cirrus and ARCHER
  - must use the C style of naming: **`man MPI_Routine_name`**, e.g.:
  - **`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

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 cirrus
  - both on login node and on compute nodes via SLURM and sbatch
  - also illustrates the SPMD model and use of basic MPI calls
- We supply some very basic template code
  - you need to add appropriate calls to compute rank and size