An Introduction to High Performance Computing on the Minerva Cluster

Stuart Rankin
sjr20@cam.ac.uk

Research Computing Services (http://www.hpc.cam.ac.uk/)
University Information Services (http://www.uis.cam.ac.uk/)

24th April 2018 / NPL User Training

Welcome

- Stuart Rankin — Research Computing User Services Team
- Paul Browne — Research Computing Platforms Team
- Course files can be downloaded from: `www.csd3.cam.ac.uk`
- Please ask questions and let us know if you need assistance.

Plan of the Course

Part 1: Basics
Part 2: HPC Facilities
Part 2: Using HPC

10:00	WELCOME
11:30-11:45	Break
13:00-14:00	LUNCH
15:30-15:45	Break
16:30	CLOSE

Part I: Basics

Basics: Why Buy a Big Computer?

What types of big problem might require a "Big Computer"?

Compute Intensive: A single problem requiring a large amount of computation.

Memory Intensive: A single problem requiring a large amount of memory.

Data Intensive: A single problem operating on a large amount of data.

High Throughput: Many unrelated problems to be executed in bulk.

Basics: Compute Intensive Problems

- Distribute the work for a single problem across multiple CPUs to reduce the execution time as far as possible.
- Program workload must be *parallelised*:
  - Parallel programs split into copies (processes or threads).
  - Each process/thread performs a part of the work on its own CPU, concurrently with the others.
  - A well-parallelised program will fully exercise as many CPUs as there are processes/threads.
- The CPUs typically need to exchange information rapidly, requiring specialized communication hardware.
- Many use cases from Physics, Chemistry, Engineering, Astronomy, Biology...
- The traditional domain of HPC and the Supercomputer.

Basics: Scaling & Amdahl's Law

- Using more CPUs is not necessarily faster.
- Typically parallel codes have a scaling limit.
- Partly due to the system overhead of managing more copies, but also to more basic constraints;
- Amdahl's Law (idealized):

$$S(N) = \frac{1}{(1 - p + \frac{p}{N})}$$

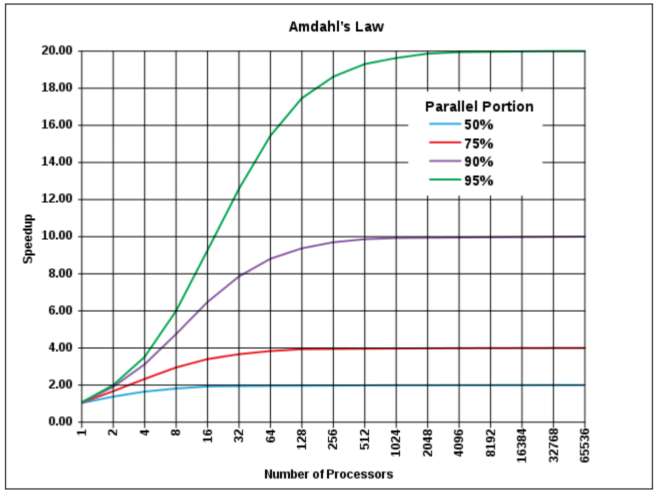
where

$S(N)$ is the fraction by which the program has sped up relative to $N = 1$

p is the fraction of the program which can be parallelized

N is the number of CPUs.

Basics: Amdahl's Law



http://en.wikipedia.org/wiki/File:AmdahlsLaw.svg

The Bottom Line

- Parallelisation requires effort:
  - There are libraries to help (e.g. OpenMP, MPI).
  - First optimise performance on one CPU, then make p as large as possible.
- The scaling limit: eventually using more CPUs becomes detrimental instead of helpful.

Basics: Data Intensive Problems

- Distribute the data for a single problem across multiple CPUs to reduce the overall execution time.
- The *same* work may be done on each data segment.
- Rapid movement of data to and from disk is more important than inter-CPU communication.
- Big Data problems of great current interest -
- Hadoop/MapReduce
- Life Sciences (genomics) and elsewhere.

Basics: High Throughput

- Distribute independent, multiple problems across multiple CPUs to reduce the overall execution time.
- Workload is trivially (or *embarrassingly*) parallel:
  - ∗ Workload breaks up naturally into *independent* pieces.
  - ∗ Each piece is performed by a separate process/thread on a separate CPU (concurrently).
  - ∗ Little or no inter-CPU communication.
- Emphasis is on throughput over a period, rather than on performance on a single problem.
- Compute intensive capable $\Rightarrow$ high throughput capable
- Compute intensive capable $\not\Leftarrow$ high throughput capable

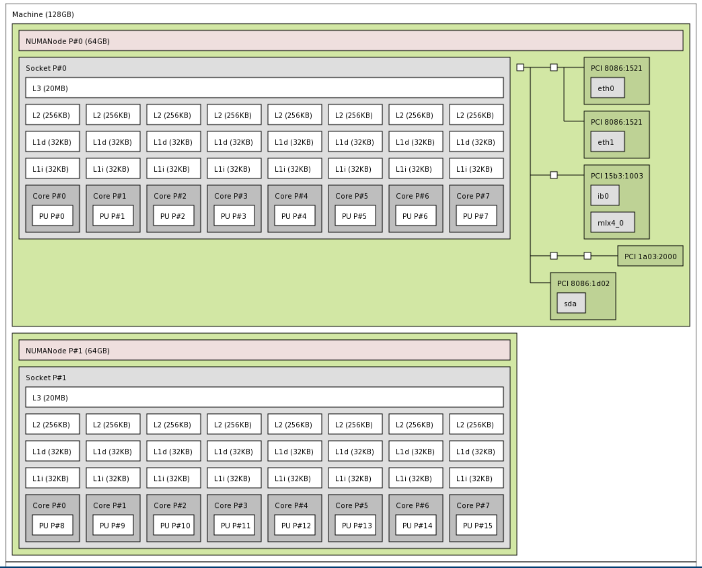
Basics: Memory Intensive Problems

- Require aggregation of large memory into a single system image (i.e. a single computer running Linux).
- Technically more challenging to build machines (very fast, low latency interconnection between all CPUs and all memory).
- Coding/porting easier (memory appears seamless).
- Optimisation harder (memory is actually highly nonuniform).
- Historically, the arena of large SGI systems.
- Nowadays, similar complexity inside single commodity servers.

Basics: Inside a Modern Computer

- Today's commodity servers already aggregate both CPUs and memory to make a single system image in a single box.
- Even small computers now have multiple CPU cores per socket
  $\implies$ each socket is a Symmetric Multi-Processor (SMP).
- Larger computers have multiple sockets (each with local memory):
  all CPU cores (unequally) share the node memory
  $\implies$ the node is a shared memory multiprocessor
  with Non-Uniform Memory Architecture (NUMA)
  but users still see a single computer (single system image).

Basics: Inside a Modern Computer


Machine (128GB)
NUMANode P#0 (64GB)
Socket P#0
L3 (20MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 Core P#1 Core P#2 Core P#3 Core P#4 Core P#5 Core P#6 Core P#7
PU P#0 PU P#1 PU P#2 PU P#3 PU P#4 PU P#5 PU P#6 PU P#7
PCI 8086:1521
eth0
PCI 8086:1521
eth1
PCI 15b3:1003
ib0
mlx4_0
PCI 1a03:2000
PCI 8086:1d02
sda
NUMANode P#1 (64GB)
Socket P#1
L3 (20MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 Core P#1 Core P#2 Core P#3 Core P#4 Core P#5 Core P#6 Core P#7
PU P#8 PU P#9 PU P#10 PU P#11 PU P#12 PU P#13 PU P#14 PU P#15

Basics: How to Build a Supercomputer

- A supercomputer aggregates contemporary CPUs and memory to obtain increased computing power.
- Usually today these are clusters.

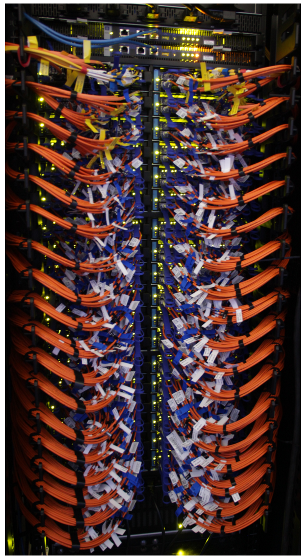
1. Take some (multicore) CPUs plus some memory.
   - Could be an off-the-shelf server, or something more special.
   - A NUMA, shared memory, multiprocessor building block: a node.

Basics: How to Build a Supercomputer

2. Connect the nodes with one or more networks. E.g.

   Gbit Ethernet: 100 MB/sec
   Omni-Path: 10 GB/sec

   Faster network is for inter-CPU communication across nodes. Slower network is for management and provisioning. Storage may use either.



Basics: How to Build a Supercomputer

3. Logically bind the nodes
    - Clusters consist of distinct nodes (i.e. separate Linux computers) on common private network(s) and controlled centrally.
        - ∗ Private networks allow CPUs in different nodes to communicate.
        - ∗ Clusters are distributed memory machines: Each process/thread sees only its local node's CPUs and memory (without help).
        - ∗ Each process/thread must fit within a single node's memory.
    - More expensive machines logically bind nodes into a single system i.e. CPUs and memory.
        - ∗ E.g. SGI UV.
        - ∗ Private networks allow CPUs to see CPUs and memory in other nodes.
        - ∗ These are shared memory machines.
        - ∗ Logically a single system - 1 big node - but very non-uniform.
        - ∗ A single process can span the entire system.

Basics: Programming a Multiprocessor Machine

- Non-parallel (serial) code
    - ∗ For a single node as for a workstation.
    - ∗ Typically run as many copies per node as CPUs, assuming node memory is sufficent.
    - ∗ Replicate across multiple nodes.
- Parallel code
    - ∗ Shared memory methods within a node. E.g. pthreads, OpenMP.
    - ∗ Distributed memory methods spanning multiple nodes. Message Passing Interface (MPI).

Basics: Summary

- Why have a supercomputer?
    - Big single problems, many problems, Big Data.
- Most current supercomputers are clusters of separate nodes.
- Each node has multiple CPUs and non-uniform shared memory.
- Parallel code uses shared memory (pthreads/OpenMP) within a node, distributed memory (MPI) spanning multiple nodes.
- Non-parallel code uses the memory of one node, but may be copied across many.

Part II: HPC Facilities

Minerva

- Each compute node:
  - ∗ 2x16 cores, Intel Skylake 2.6 GHz32 CPUs
  - ∗ 192 GB RAM6 GB per CPU
  - ∗ 100 Gb/sec Omni-Path10 GB/sec (for MPI and storage)
- 16 compute nodes
- 1 high-memory login/head node (minerva-login1.npl.co.uk), 768 GB RAM.
- 1 GPU node, 2 × NVIDIA P100 GPU, 192 GB RAM.

CSD3 - University of Cambridge

- Cambridge Service for Data Driven Discovery
- Peta4 — Intel CPU cluster
- Wilkes2 — NVIDIA GPU cluster
- Hadoop-based data analytic platform
- Burst buffer
- Some NPL users through CORE.

Peta4-Skylake

- Each compute node:
  - ∗ 2x16 cores, Intel Skylake 2.6 GHz32 CPUs
  - ∗ 192 GB or 384 GB RAM6 GB or 12 GB per CPU
  - ∗ 100 Gb/sec Omni-Path10 GB/sec (for MPI and storage)
- 768 compute nodes
- 8 login nodes (login-cpu.hpc.cam.ac.uk)

Coprocessors — GPUs etc

- CPUs are general purpose
- Some types of parallel workload fit vector processing well:
  - Single Instruction, Multiple Data (SIMD)
  - *Think pixels on a screen*
  - GPUs specialise in this type of work
  - Also competitor many-core architectures such as the Intel Phi

Wilkes2-GPU

- Each compute node:
  - 4 × NVIDIA P100 GPU4 GPUs
  - 1x12 cores, Intel Broadwell 2.2 GHz12 CPUs
  - 96 GB RAM96 GB RAM
  - 100 Gb/sec (4X EDR) Infiniband.10 GB/sec (for MPI and storage)
- 90 compute nodes.
- 8 login nodes (login-gpu.hpc.cam.ac.uk).

Peta4-KNL (Intel Phi)

- Each compute node:
  - 64 cores, Intel Phi 7210256 CPUs
  - 96 GB RAM96 GB RAM
  - 100 Gb/sec Omni-Path10 GB/sec (for MPI and storage)
- 342 compute nodes
- Shared login nodes with Peta4-Skylake

Cluster Storage

- Minerva uses NFS to share user directories between all cluster nodes (150 TB).
- CSD3 uses the Lustre cluster filesystem:
  - Very scalable, high bandwidth.
  - Multiple RAID6 back-end disk volumes.
  - Multiple object storage servers.
  - Single metadata server.
  - Tape-backed HSM on newest filesystems.
  - 12 GB/sec overall read or write.
  - Prefers big read/writes over small.

Obtaining an Account and Support

- Please contact the NPL IT Service Desk:
  - itservicedesk@npl.co.uk
  - Room F12-CS1
  - Tel. 6000
- Second line support is provided by the Cambridge RCS.

Part III: Using HPC

Using HPC: Connecting

- SSH secure protocol only.
  Supports login, file transfer, remote desktop. . .

Connecting: Windows Clients

- putty, pscp, psftp
  http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html
- WinSCP
  http://winscp.net/eng/download.php
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- Cygwin (provides an application environment similar to Linux)
  http://cygwin.com/install.html
  Includes X server for displaying graphical applications running remotely.
- MobaXterm
  http://mobaxterm.mobatek.net/

Connecting: Linux/MacOSX/UNIX Clients

- ssh, scp, sftp, rsync
  Installed (or installable).
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- On MacOSX, install XQuartz to display remote graphical applications.
  http://xquartz.macosforge.org/landing/

Connecting: Login

- From graphical clients:
  Host: minerva-login1.npl.co.uk
  Username: **npl**\abc123 (your NPL AD account name)
- From Linux/MacOSX/UNIX (or Cygwin):

  ssh -Y **npl**\\abc12@minerva-login1.npl.co.uk

  Note the double backslash — this is because UNIX command interpreters treat \\ as special.

Connecting: First time login

- The first connection to a particular hostname produces the following:

```
The authenticity of host 'minerva-login1.npl.co.uk (139.143.201.10)' can't be established.

ECDSA key fingerprint is SHA256:k/eB+LjcAfQW56XCzK9QptT0wVWF7j3a/CPxPRd7+lE.
ECDSA key fingerprint is MD5:18:9a:97:e2:87:4c:07:60:cb:43:46:f2:bb:d8:3d:01.

Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'minerva-login1.npl.co.uk (139.143.201.10)' (ECDSA) to the list of known
hosts.
```

- One should always check the fingerprint before typing “yes”.
- Graphical SSH clients *should* ask a similar question.
- Designed to detect fraudulent servers.

Connecting: First time login

- Exercise 1 - Log into your Minerva account.
- Exercise 2 - Simple command line operations.

Connecting: File Transfer

- With graphical clients, connect as before and drag and drop.
- From Linux/MacOSX/UNIX (or Cygwin):

  rsync -av **old_directory/**
  npl\\\\abc12@minerva-login1.npl.co.uk:hpc-work/new_directory
  copies contents of old_directory to ~/hpc-work/new_directory.

  rsync -av **old_directory**
  npl\\\\abc12@minerva-login1.npl.co.uk:hpc-work/new_directory
  copies old_directory (and contents) to ~/hpc-work/new_directory/old_directory.

  - Rerun to update or resume after interruption.
  - All transfers are checksummed.
  - For transfers in the opposite direction, place the remote machine as the first argument.
- Exercise 3 - File transfer.

Connecting: Remote Desktop

- First time starting a remote desktop:

```
[sjr20@login-a-1 ~]$ vncserver

You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? n

New 'login-a-1:99 (sjr20)' desktop is login-a-1:99

Starting applications specified in /home/sjr20/.vnc/xstartup
Log file is /home/sjr20/.vnc/login-a-1:99.log
```

- NB Choose a different password for VNC.
- The VNC password protects your desktop from other users.
- Remember the unique display number (99 here) of your desktop.

Connecting: Remote Desktop

- Remote desktop already running:

```
[sjr20@login-a-1 ~]$ vncserver -list

TigerVNC server sessions:

X DISPLAY #     PROCESS ID
:99             130655
```

- Kill it:

```
[sjr20@login-a-1 ~]$ vncserver -kill :99
Killing Xvnc process ID 130655
```

- Typically you only need one remote desktop.
- Keeps running until killed, or the node reboots.

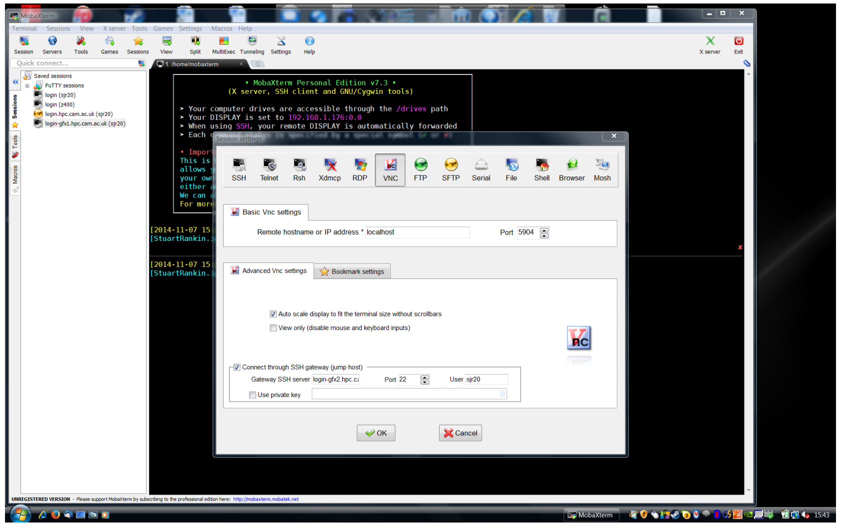
Connecting: Remote Desktop

- To connect to the desktop from Linux:

```
vncviewer -via npl\\abc12@minerva-login1.npl.ad.local localhost:99
```

- The display number 99 will be different in general and unique to each desktop.
- You will be asked firstly for your AD login password, and secondly for your VNC password.
- Press F8 to bring up the control panel.
- Exercise 4 - Remote desktop (from Windows)

Connecting: Remote Desktop (MobaXterm)


MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Settings Help
X server Exit
Quick connect...
1. /home/mobaxterm
Saved sessions
PuTTY sessions
login (sjr20)
login (z400)
login.hpc.cam.ac.uk (sjr20)
login-gfx1.hpc.cam.ac.uk (sjr20)
Sessions
Tools
Macros
Session settings
SSH Telnet Rsh Xdmcp RDP VNC FTP SFTP Serial File Shell Browser Mosh
OK
Cancel
MobaXterm
15:43

```
• MobaXterm Personal Edition v7.3 •
(X server, SSH client and GNU/Cygwin tools)

➤ Your computer drives are accessible through the /drives path
➤ Your DISPLAY is set to 192.168.1.176:0.0
➤ When using SSH, your remote DISPLAY is automatically forwarded
➤ Each
• Import
This is
allows
your own
either
We can
For more

[2014-11-07 15
[StuartRankin.

[2014-11-07 15
[StuartRankin.
```

Basic Vnc settings

Remote hostname or IP address * localhost   Port 5904

Advanced Vnc settings | Bookmark settings

- [✓] Auto scale display to fit the terminal size without scrollbars
- [ ] View only (disable mouse and keyboard inputs)

[✓] Connect through SSH gateway (jump host)

Gateway SSH server login-gfx2.hpc.ca   Port 22   User sjr20

[ ] Use private key

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

Using HPC: User Environment

- CentOS Linux 7.4 (Red Hat Enterprise Linux 7.4 rebuild)
  - bash shell
  - Gnome or XFCE4 desktop (if you want)
  - GCC compilers and other development software.
- But you don't need to know that.

User Environment: Filesystems

- /home/abc12@npl.ad.local
  - 50GB quota.
  - Visible equally from all nodes.
  - Single storage server.
  - Regular backups.
  - Not intended for job outputs or large/many input files.
- ~/hpc-work
  - Visible equally from all nodes.
  - Larger (1TB initial quota).
  - Intended for job inputs and outputs.
  - Not backed up by default.

Filesystems: Quotas

- quota

```
[sjr20@login-a-1 ~]$ quota -s
Disk quotas for user sjr20 (uid 1004):
     Filesystem   space   quota   limit   grace   files   quota   limit   grace
10.44.82.252:/hpc-work
                     0K   1024G   1126G               1       0       0
10.44.82.252:/home
                 13272K  51200M  56320M             345       0       0
```

- Aim to stay below the soft limit (*quota*).
- Once over the soft limit, you have 7 days grace to return below.
- When the grace period expires, or you reach the hard limit (*limit*), no more data can be written.
- It is important to rectify an out of quota condition ASAP.

Filesystems: Quotas

- quota

```
[sjr20@login-a-1 ~]$ quota -s
Disk quotas for user sjr20 (uid 1004):
     Filesystem   space   quota   limit   grace   files   quota   limit   grace
10.44.82.252:/hpc-work
                     0K   1024G   1126G               1       0       0
10.44.82.252:/home
                 13272K  51200M  56320M             345       0       0
```

- Aim to stay below the soft limit (*quota*).
- Once over the soft limit, you have 7 days grace to return below.
- When the grace period expires, or you reach the hard limit (*limit*), no more data can be written.
- It is important to rectify an out of quota condition ASAP.

Filesystems: Permissions

- Be careful and if unsure, please ask support.
  - Can lead to accidental destruction of your data or account compromise.
- Avoid changing the permissions on your home directory.
  - Files under /home are particularly security sensitive.
  - Easy to break passwordless communication between nodes.

User Environment: Software

- Free software accompanying Red Hat Enterprise Linux is (or can be) provided.
- Other software (free and non-free) is available via modules.
- Proprietary software currently available includes Matlab and COMSOL.
- New software may be possible to provide on request.
- Self-installed software should be properly licensed.
- *sudo will not work.* (You should be worried if it did.)

User Environment: Environment Modules

- Modules load or unload additional software packages.
- Some are required and automatically loaded on login.
- Others are optional extras, or possible replacements for other modules.
- Beware unloading default modules in ~/.bashrc.
- Beware overwriting environment variables such as PATH and LD_LIBRARY_PATH in ~/.bashrc. If necessary append or prepend.

User Environment: Environment Modules

- Currently loaded:

```
module list
Currently Loaded Modulefiles:
  1) dot                        3) centos7/global
  2) slurm                      4) centos7/default-basic
```

- Available:

```
module av
```

User Environment: Environment Modules

- Whatis:

```
module whatis openmpi-3.0.0-gcc-4.8.5-n2hvjgm
openmpi-3.0.0-gcc-4.8.5-n2hvjgm: The Open MPI Project is an open source...
```

- Load:

```
module load openmpi-3.0.0-gcc-4.8.5-n2hvjgm
```

- Unload:

```
module unload openmpi-3.0.0-gcc-4.8.5-n2hvjgm
```

User Environment: Environment Modules

- Matlab

```
module load matlab/r2018a
```

- Invoking matlab in batch mode:
  matlab -nodisplay -nojvm -nosplash command
  where the file command.m contains your matlab code.
- The current site license contains the Parallel Computing Toolbox.

User Environment: Environment Modules

- Purge:

```
module purge
```

- Defaults:

```
module show centos7/default-basic
module load centos7/default-basic
```

- Run time environment must match compile time environment.

User Environment: Compilers

- GCC

```
gcc -O3 -mtune=native code.c -o prog
gfortran -O3 -mtune=native code.f90 -o prog
```

```
module load openmpi-3.0.0-gcc-4.8.5-n2hvjgm
mpicc -O3 -mtune=native mpi_code.c -o mpi_prog
mpif90 -O3 -mtune=native mpi_code.f90 -o mpi_prog
```

- Exercise 5: Modules and Compilers

Using HPC: Job Submission

Using HPC: Job Submission

- Compute resources are managed by a scheduler:
  SLURM/PBS/SGE/LSF/. . .
- Jobs are submitted to the scheduler
  — analogous to submitting jobs to a print queue
  — a file (*submission script*) is copied and queued for processing.

Using HPC: Job Submission

- Jobs are submitted from the login node
  — not itself managed by the scheduler.
- Jobs may be either non-interactive (batch) or interactive.
- Batch jobs run a shell script on the first of a list of allocated nodes.
- Interactive jobs provide a command line on the first of a list of allocated nodes.

Using HPC: Job Submission

- Jobs may use part or all of one or more nodes
  — the owner can specify `--exclusive` to force exclusive node access.
- Template submission scripts are available under
  ~/job_templates.

Job Submission: Using SLURM

- Prepare a shell script and submit it to SLURM:

```
[abc123@login-a-1]$ sbatch slurm_submission_script
Submitted batch job 790299
```

Job Submission: Show Queue

- Submitted job scripts are copied and stored in a queue:

```
[abc123@login-a-1]$ squeue -u abc123
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            790299   skylake     Test3  abc123 PD       0:00      2 (PriorityResourcesAssocGrpCPUMinsLimit)
            790290   skylake     Test2  abc123  R   27:56:10      2 cpu-a-[1,10]
```

Job Submission: Monitor Job

- Examine a particular job:

```
[abc123@login-a-1]$ scontrol show job=790290
```

Job Submission: Cancel Job

- Cancel a particular job:

```
[abc123@login-a-1]$ scancel 790290
```

Job Submission: Scripts

- SLURM
  In ~/job_templates, see examples: slurm_submit.skylake.generic, slurm_submit.skylake.matlab.

```
#!/bin/bash
#! Name of the job:
#SBATCH -J myjob
#! Which project should be charged:
#SBATCH -A NPL-GENERAL-CPU
#! How many whole nodes should be allocated?
#SBATCH --nodes=1
#! How many tasks will there be in total? (<= nodes*32)
#SBATCH --ntasks=116
#! How much wallclock time will be required?
#SBATCH --time=02:00:00
#! Select partition:
#SBATCH -p skylake
...
```

- #SBATCH lines are *structured comments*
  — correspond to sbatch command line options.
- The above job will be given 1 cpu16 cpus on 1 node for 2 hours (by default there is 1 task per node, and 1 cpu per task).

UNIVERSITY OF CAMBRIDGE

Job Submission: Accounting Commands

- How many core hours available do I have?

```
mybalance

User            Usage |        Account     Usage | Account Limit Available (hours)
---------- --------- + -------------- --------- + ------------- ---------
sjr20               3 |    SUPPORT-CPU     2,929 |    22,425,600 22,422,671
sjr20               0 |    SUPPORT-GPU         0 |        87,600    87,600
```

- How many core hours does some other project or user have?

```
gbalance -p SUPPORT-CPU

User            Usage |        Account     Usage | Account Limit Available (hours)
---------- --------- + -------------- --------- + ------------- ---------

pfb29           2,925 |    SUPPORT-CPU     2,929 |    22,425,600 22,422,671
sjr20 *             3 |    SUPPORT-CPU     2,929 |    22,425,600 22,422,671
...
(Use -u for user.)
```

- List all jobs charged to a project/user between certain times:

```
gstatement -p NPL-GENERAL-CPU  -u xyz10 -s "2018-04-01-00:00:00" -e "2018-04-30-00:00:00"
       JobID      User    Account    JobName  Partition                 End ExitCode      State  CompHrs
------------ --------- ---------- ---------- ---------- ------------------- -------- ---------- --------
263              xyz10 support-c+ _interact+    skylake 2018-04-18T19:44:40      0:0    TIMEOUT      1.0
264              xyz10 support-c+ _interact+    skylake 2018-04-18T19:48:07      0:0 CANCELLED+      0.1
275              xyz10 support-c+ _interact+    skylake             Unknown      0:0    RUNNING      0.3
...
```

UNIVERSITY OF CAMBRIDGE

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=

#SBATCH --mem=5990
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=  # For OpenMP across  cores
$application $options
...
```

UNIVERSITY OF CAMBRIDGE

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=1
# Default is 1 cpu (core) per task
#SBATCH --mem=5990
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=  # For OpenMP across  cores
$application $options
...
```

UNIVERSITY OF CAMBRIDGE

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=32 # Whole node

#SBATCH --mem=5990
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=32  # For OpenMP across 32 cores
$application $options
...
```

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=16  # Half node

#SBATCH --mem=5990
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=16  # For OpenMP across 16 cores
$application $options
...
```

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=32 # Whole node

#SBATCH --mem=5990
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=16  # For OpenMP across 16 cores (using all memory)
$application $options
...
```

Job Submission: MPI Jobs

- Parallel job across multiple nodes.

```
#!/bin/bash
...
#SBATCH --nodes=4
#SBATCH --ntasks=128     # i.e. 32x4 MPI tasks in total.
#SBATCH --cpus-per-task=2
...
mpirun -np 128 $application $options
...
```

- SLURM-aware MPI launches remote tasks via SLURM.
- The template script uses $SLURM_TASKS_PER_NODE to set PPN.

Job Submission: MPI Jobs

- Parallel job across multiple nodes.

```
#!/bin/bash
...
#SBATCH --nodes=4
#SBATCH --ntasks=64     # i.e.  16x4 MPI tasks in total.
#SBATCH --cpus-per-task=2
...
mpirun -ppn 16 -np 64 $application $options
...
```

- SLURM-aware MPI launches remote tasks via SLURM.
- The template script uses $SLURM_TASKS_PER_NODE to set PPN.

Job Submission: Hybrid Jobs

- Parallel jobs using both MPI and OpenMP.

```
#!/bin/bash
...
#SBATCH --nodes=4
#SBATCH --ntasks=64     # i.e. 16x4 MPI tasks in total.
#SBATCH --cpus-per-task=2
...
export OMP_NUM_THREADS=2   # i.e. 2 threads per MPI task.
mpirun -ppn 16 -np 64 $application $options
...
```

- This job uses 128 CPUs (each MPI task splits into 2 OpenMP threads).

Job Submission: High Throughput Jobs

- Multiple serial jobs across multiple nodes.
- Use srun to launch tasks (job steps) within a job.

```
#!/bin/bash
...
#SBATCH --nodes=2
...
cd directory_for_job1
srun --exclusive -N 1 -n 1 $application $options_for_job1 > output 2> err &
cd directory_for_job2
srun --exclusive -N 1 -n 1 $application $options_for_job2 > output 2> err &
...
cd directory_for_job64
srun --exclusive -N 1 -n 1 $application $options_for_job64 > output 2> err &
wait
```

- Exercise 6 - Submitting Jobs.

Job Submission: Interactive

- Compute nodes are accessible via SSH while you have a job running on them.
- Alternatively, submit an interactive job:

  `sintr -A NPL-GENERAL-CPU -N1 -n8 -t 2:0:0`

- Within the window (screen session):
  - Launches a shell on the first node (when the job starts).
  - Graphical applications should display correctly.
  - Create new shells with ctrl-a c, navigate with ctrl-a n and ctrl-a p.
  - ssh or srun can be used to start processes on any nodes in the job.
  - SLURM-aware MPI will do this automatically.

Job Submission: Array Jobs

- *http : //slurm.schedmd.com/job_array.html*
- Used for submitting and managing large sets of similar jobs.
- Each job in the array has the same initial options.
- SLURM

```
[abc123@login-a-1]$ sbatch --array=1-7:21,3,5,7 -A NPL-GENERAL-CPU submit_script
Submitted batch job 791609
```

```
[abc123@login-a-1]$ squeue -u abc123
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          791609_1 skylake      hpl    abc123  R       0:06      1 cpu-a-6
          791609_3 skylake      hpl    abc123  R       0:06      1 cpu-a-16
          791609_5 skylake      hpl    abc123  R       0:06      1 cpu-a-7
          791609_7 skylake      hpl    abc123  R       0:06      1 cpu-a-7
```

791609_1, 791609_3, 791609_5, 791609_7

i.e. ${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}

SLURM_ARRAY_JOB_ID = SLURM_JOBID for the first element.

Job Submission: Array Jobs (ctd)

- Updates can be applied to specific array elements using ${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}
- Alternatively operate on the entire array via ${SLURM_ARRAY_JOB_ID}.
- Some commands still require the SLURM_JOB_ID (sacct, sreport, sshare, sstat and a few others).
- Exercise 7 - Array Jobs.

Scheduling

- SLURM scheduling is multifactor:
  - QoS — payer or non-payer?
  - Age — how long has the job waited?
    Don't cancel jobs that seem to wait too long.
  - Fair Share — how much recent usage?
    Payers with little recent usage receive boost (not implemented yet).
  - sprio -j jobid
- Backfilling
  - Promote lower priority jobs into gaps left by higher priority jobs.
  - Demands that the higher priority jobs not be delayed.
  - Relies on reasonably accurate wall time requests for this to work.
  - Jobs of default length will not backfill readily.

Wait Times

- 36 hour job walltimes are permitted.
- This sets the timescale at busy times (*without* backfilling).
- Use backfilling when possible.
- Short (1 hour or less) jobs have higher throughput.

Checkpointing

- Insurance against failures during long jobs.
- Restart from checkpoints to work around finite job length.
- Application native methods are best. Failing that . . .

Job Submission: Scheduling Top Dos & Don’ts

- **Do . . .**
  - Give reasonably accurate wall times (allows backfilling).
  - Check your balance occasionally (mybalance).
  - Test on a small scale first.
  - Implement checkpointing if possible (reduces resource wastage).
- **Don’t . . .**
  - Request more than you need
    — you will wait longer and use more credits.
  - Cancel jobs unnecessarily
    — priority increases over time.