Quick Reference Guide to the High Performance Computer Environments @ Pan Cluster

New Zealand eScience Infrastructure

New Zealand eScience Infrastructure (NeSI)
a partnership for all New Zealand researchers delivered by
The University of Auckland, NIWA, University of Canterbury,
Landcare Research, Otago University and MBIE

Contact Information

- **web**: `http://www.nesi.org.nz`
- **wiki**: `https://wiki.auckland.ac.nz/display/CERES/`
- **ganglia**: `http://ganglia.uoa.nesi.org.nz`
- **support** : support@nesi.org.nz

Useful Quick References

- VI Quick Reference
- BASH Quick Reference
- Linux Quick Reference
- OpenMP Fortran Syntax
- OpenMP 3.1 API C/C++ Syntax
- MPI Quick Reference

Hardware Information

Architecture	Westmere	SandyBridge	LargeMem	GPU
Model	X5660	E5-2680	E7-4870	X5660 (M2090)
Clock Speed	2.8 GHz	2.7 GHz	2.4GHz	2.8 (1.3)GHz
Cache	12MB	20MB	30MB	12MB
Intel QPI speed	6.4GT/s	8 GT/s	6.4GT/	6.4GT/s
Cores/socket	6	8	10	6(512)
Cores/node	12	16	40	12(1024)
Mem/node	96GB	128GB	512GB	96GB(6GB)
GFLOPS/node	134.4	172.8	384.0	134.4 (1330DP)

Disk Spaces + Default Quota

FileSystem	Space	Quota	ACL	Backup	Type	Usage
$HOME	120TB	30GB	rw	yes	GPFS	Archive
/share	120TB	-	ro	yes	GPFS	Archive
$TMP_DIR	240GB	-	rw	NO	EXT4	io
$SCRATCH_DIR	8.8TB	-	rw	NO	GPFS	io

Environment Modules

Environment Modules is very useful to manage environment variables for each application and it is very easy to use. It loads the needed environment by a certain application and its dependencies automatically.

- **module avail** - lists available modules
- **module show module_name** - displays full information about the module with name *module_name*.
- **module load module_name** - loads the module with name *module_name* and its dependencies.
- **module unload module_name** - unload the module with name *module_name* and its dependencies.
- **module list** - list all modules currently loaded.

Available Compilers

Compiler	Intel	GNU	PGI
Fortran77	ifort	g77	pgf77
Fortran90	ifort	gfortran	pgf90
Fortran95	ifort	gfortran	pgf95
C	icc	gcc	pgcc
C++	icpc	g++	pgCC
Debug	idb	gdb	pgdbg
Profile	vtune	gprof	gpprof

Available MPIs

MPI	version
Intel MPI	4.1.0.024
OpenMPI	1.4,1.6
MPICH2	1.5,3.0.4
PlatformMPI	08.02
MVAPICH2	1.4.1p1

Optimization flags

Compiler	Intel	GNU	PGI
High Opt.	-fast	-O3 -ffast-math	-fast -Mipa=fast,inline
OpenMP	-openmp	-fopenmp	-mp=nonuma
Debug	-g	-g	-g
Profile	-p	-pg	-p
Westmere	-mtarget	-march=corei7	-tp=nehalem-64
SandyBridge	-mtarget	-march=corei7-avx	-tp=sandybridge-64
SSE	-xsse4.2	-msse4.2	-Mvect=[prefetch,sse]
AVX[1]	-xavx	-mavx	-fast

1. Advanced Vector Extension (AVX) streaming SIMD instructions. Sandy Bridge processor only.

Link MKL with OpenMPI, CDFT, ScaLAPACK, BLACS and Intel Compilers

Link line:
```
-L${MKLROOT}/lib/intel64 -lmkl_scalapack_ilp64 \
-lmkl_cdft_core -lmkl_intel_ilp64 -lmkl_sequential \
-lmkl_core -lmkl_blacs_intelmpi_ilp64 -lpthread -lm
```
Compiler options: `-DMKL_ILP64 -I${MKLROOT}/include`
More information at `http://software.intel.com/sites/products/mkl/`

Link MKL with OpenMP and Intel compilers

Link line:
```
-L${MKLROOT}/lib/intel64 -lmkl_intel_ilp64 \
-lmkl_intel_thread -lmkl_core -lpthread -lm
```
Compiler options: `-openmp -DMKL_ILP64 -I${MKLROOT}/include`

Slurm Commands

- **sbatch** sends a new job to the scheduler for execution
- **sinfo** show the current list of classes.
- **squeue** returns queue information
  - `squeue -l -j <job id>` gives extended job record information.
  - `squeue -u <user name>` lists only jobs initiated by the user.
- **scancel** kills a queued job: `scancel <job id>`

More information about Slurm in the NeSI Slurm User Guide :
`https://wiki.auckland.ac.nz/display/CER/Slurm+User+Guide`

Limits

Name	MaxJobCPU	Description
small1h	41	Jobs running for under 1 hour (high priority)
small6h	41	Jobs running for under 6 hours (high priority)
medium	12500	Jobs running for under 1 week (medium priority)
long	unlimited	Jobs running for longer than a week (low priority)
bigmem	unlimited	Jobs that needs more than 12GB/core

OpenSSH Access

Parameters

- host: login-01.uoa.nesi.org.nz
- port: 22

Limits

- CPU time: 60 minutes
- Memory : 2GB

Suggested Software

- Windows: mobaxterm, Putty, Bitvise Tunnelier
- MacOSX: Terminal(Included in the OS), iTerm2
- Linux: Konsole, GnomeTerminal, yakuake

interactive sessions

The interactive sessions will allow you to build the binaries for specific architecture. The binaries compiled in Westmere can run in Sandy Bridge, but it can **NOT** exploit all the Sandy bridge features. The binaries compiled in Sandy Bridge can **NOT** run in the Westmere nodes. The interactively usage is limited up to **24h** of **walltime**.

```
Usage: interactive [-A] [-a] [-c] [-m] [-J]
Mandatory arguments:
 -A: account
Optional arguments:
 -a: architecture (default: wm, values sb=SandyBridge wm=Westmere)
 -c: number of CPU cores (default: 1)
 -m: amount of memory (GB) per core (default: 1 [GB])
 -J: job name
example : interactive -A nesi99999 -a wm -c 4 -J MyInteractiveJob
```

Remote File System Access

In order to access the file system (/home) remotely from your machine, we recommend:

- **SSHFS** (MacOSX) : `http://code.google.com/p/macfuse/`
- **SSHFS** (Linux) : `http://fuse.sourceforge.net/sshfs.html`
- **SSHFS** (Windows) : `http://code.google.com/p/win-sshfs/`
- **Konqueror** (KDE) : type fish://user@host:port
- **Nautilus** (Gnome) : type sftp://user@host:port
- **WinSCP** (Windows) : `http://winscp.net`

Remote File System Transfer with RSYNC (Unix Only)

RSYNC over SSH protocol is the best choice to transfer big data volumes.

- Transfer data from your machine to the server:
  `rsync -avHl /path/origin/* sshserver:/path/destination/`
- Transfer data from the server to your machine:
  `rsync -avHl sshserver:/path/destination/* /path/origin/`

Remote File System Transfer with scp/sftp (Unix Only)

SCP and **SFTP** are the most popular software to transfer data across the SSH protocol.

- **SCP**: `scp -pr sshserver:/path/destination/* path/destination/`
- **SFTP**: `sftp sshserver:/path/destination/* path/destination/`

Get Involved!

Quick Reference Guide to the High Performance Computer Environments @ Pan Cluster

New Zealand eScience Infrastructure

New Zealand eScience Infrastructure (NeSI)
a partnership for all New Zealand researchers delivered by
The University of Auckland, NIWA, University of Canterbury,
Landcare Research, Otago University and MBIE

Submit Script Example : Serial

```
#!/bin/bash
#SBATCH -J OpenMP_JOB
#SBATCH -A uoa99999          # Project Account
#SBATCH --time=01:00:00      # Walltime
#SBATCH --mem-per-cpu=8132   # memory/cpu (in MB)
#SBATCH -C sb                # sb=Sandybridge,wm=Westmere
srun serial_binary
```

Submit Script Example : OpenMP

```
#!/bin/bash
#SBATCH -J OpenMP_JOB
#SBATCH -A uoa99999          # Project Account
#SBATCH --time=01:00:00      # Walltime
#SBATCH --mem-per-cpu=8132   # memory/cpu (in MB)
#SBATCH --cpus-per-task=8    # 8 OpenMP Threads
#SBATCH -C sb                # sb=Sandybridge,wm=Westmere
srun openmp_binary
```

Submit Script Example : MPI

```
#!/bin/bash
#SBATCH -J MPI_JOB
#SBATCH -A uoa99999          # Project Account
#SBATCH --time=01:00:00      # Walltime
#SBATCH --ntasks=2           # number of tasks
#SBATCH --mem-per-cpu=8132   # memory/cpu (in MB)
#SBATCH -C sb                # sb=Sandybridge,wm=Westmere
srun mpi_binary
```

Submit Script Example : Hybrid (MPI+OpenMP)

```
#!/bin/bash
#SBATCH -J Hybrid_JOB
#SBATCH -A uoa99999          # Project Account
#SBATCH --time=01:00:00      # Walltime
#SBATCH --ntasks=4           # number of tasks
#SBATCH --mem-per-cpu=8132   # memory/cpu (in MB)
#SBATCH --cpus-per-task=8    # 8 OpenMP Threads
#SBATCH --nodes=1            # number nodes
srun binary_hybrid
```

Submit Script Example : Hybrid (MPI+CUDA)

```
#!/bin/bash
#SBATCH -J GPU_JOB
#SBATCH --time=01:00:00      # Walltime
#SBATCH -A uoa99999          # Project Account
#SBATCH --ntasks=4           # number of tasks
#SBATCH --ntasks-per-node=2  # number of tasks per node
#SBATCH --mem-per-cpu=8132   # memory/cpu (in MB)
#SBATCH --cpus-per-task=4    # 4 OpenMP Threads
# The following line will request GPUs per node. In this particular
# example, it means 4 GPUs in total.
#SBATCH --gres=gpu:2
#SBATCH -C kepler
srun binary_cuda_mpi
```

Submit Script Syntax

- **A uoa99999** User account (uoaXXXXX, uocXXXXX, uooXXXXX, landcareXXXXX or nesiXXXX)
- **mem-per-cpu=8132** = memory/cpu (in MB)
- **-J My_Job_Name** Sets the job name
- **mail-type=ALL** Specifies under which conditions Slurm will send out email notifications to the specified address about the state of the job:
  - ALL - notifies about every change in the job status
  - START - notifies when the job is activated
  - END - notifies when the job ends
  - ERROR - notifies only if the job fails
  - NEVER - never send any notifications
- **–mail-user=username@nesi.org.nz** user email
- **–cpus-per-task=8** Sets the number of cores to be allocated for each task
- **–gres=gpu:1—2** Specifies what consumable resources the job requires to run per task:
  - gpu:1—2 - default 1 GPU per node
  - mic:1—2 - default 1 Intel Xeon Phi per node
- **ntasks-per-node=16** specifies the total number of tasks (cores) to be run on each available node.
- **ntasks=96** specifies how many tasks (cores) are to be run in total
- **time=01:00:00** Sets the limit for the elapsed time for which a job can run

Recommended Best Practices

- Use the **SubmitScripts Templates** that are in /share/SubmitScripts.
- Ask for the minimum nodes for the specified number of cores for the very large jobs. It will take more time to enter in execution but will finish with less walltime.

Submit Script Example : Job Array

```
#!/bin/bash
#SBATCH -J JobArray
#SBATCH --time=01:00:00      # Walltime
#SBATCH -A uoa99999          # Project Account
#SBATCH --ntasks=1           # number of tasks
#SBATCH --mem-per-cpu=8132   # memory/cpu (in MB)
#SBATCH --cpus-per-task=4    # 4 OpenMP Threads
#SBATCH --array=1-1000       # Array definition
#SBATCH -C sb                # sb=Sandybridge,wm=Westmere
srun binary_array $SLURM_ARRAY_TASK_ID
```

Grisu Command Line Interface (griclish)

- `help` - displays help menu
- `help <keyword>` - displays help about a keyword
- `attach <path_to_file>` - attaches a file to the next job
- `set <property> <value>` - sets a value for the next job
- `submit <command_to_run>` - submits a job
- `print jobs` - lists all jobs and their statuses
- `print job <jobname>` - displays details about a job