

## Contact Information

- web:** <http://www.nesi.org.nz>
- wiki:** <https://wiki.auckland.ac.nz/display/CER/>
- ganglia:** <http://ganglia.uoa.nesi.org.nz>
- support :** [support@nesi.org.nz](mailto:support@nesi.org.nz)



## Recommended Best Practices

- Use the **SubmitScripts Templates** that are in /share/SubmitScripts.
- In order to improve the scalability of your MPI code, we suggest to minimise the number of nodes to be used. The `--nodes=N-M` Slurm option, will define the minimum (N) and the maximum (M) number of nodes.

## 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

## NeSI Pan Cluster

Architecture	Westmere	SandyBridge	LargeMem
Model	X5660	E5-2680	E7-4870
Clock Speed	2.8 GHz	2.7 GHz	2.4GHz
Cache	12MB	20MB	30MB
Intel QPI speed	6.4GT/s	8 GT/s	6.4GT/
Cores/socket	6	8	10
Cores/node	12	16	40
Mem/node	96GB	128GB	512GB
GFLOPS/node	134.4	345.6	384.0
# nodes	76	194	4

## NeSI Pan Cluster - Co-Processors

Architecture	Nvidia Fermi	Nvidia Kepler	Intel Phi
Main CPU	X5660/E5-2680	E5-2680	E5-2680
Model	M2090	K20X	5110P
Clock Speed	1.3GHz	0.732GHz	1.053GHz
Cores/Dev.	512	2688	60 (240)
Dev./node	2	2	2
Mem/Dev.	6GB	6GB	8GB
TFLOPS/Dev	1.33	1.17	1.01
# nodes	16	5	2

## Disk Spaces + Default Quota

FileSystem	Space	Quota	ACL	Backup	Type	Usage
\$HOME	120TB	2GB	rw	yes	GPFS	Archive
/project	120TB	30GB	rw	yes	GPFS	Archive
/share	120TB	-	ro	yes	GPFS	Archive
\$TMP_DIR	240GB	-	rw	NO	EXT4	io
\$SCRATCH_DIR	13TB	-	rw	NO	GPFS	io
\$SHM_DIR	Mem	-	rw	NO	RamFS	io

## Available Compilers

Language	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	4.1.0.4.1.1
OpenMPI	1.4.1.6.1.8
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 <sup>1</sup>	-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`



We would like to encourage you to send suggestions and feedback to the NeSI Team ([support@nesi.org.nz](mailto:support@nesi.org.nz)).

## SSH Access : Login Node

The login node is not for running jobs, it is only for file management and job submission.

### Parameters

- host: [login-01.uoa.nesi.org.nz](http://login-01.uoa.nesi.org.nz)
- port: 22

## Suggested Software

- mobaxterm** (Windows) : <http://mobaxterm.mobatek.net>
- Putty** (Windows) : <http://www.chiark.greenend.org.uk/~sgtatham/putty/>
- Terminal** (MacOSX) : (Included in the OS)
- iTerm2** (MacOSX) : <http://www.iterm2.com>
- Konsole** (Linux) : <http://konsole.kde.org>
- GnomeTerminal** (Linux) : <https://wiki.gnome.org/Apps/Terminal>
- yakuake** (Linux) : <http://yakuake.kde.org>

## 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>
- 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/`

## Bash Shortcuts

- Ctrl+c** - halts the current command.
- Ctrl+z** - stops the current command.
- Ctrl+d** - log out of current session.
- Ctrl+k** - Delete from the cursor to the end of the line.
- Ctrl+w** - Delete from the cursor to the start of the word.
- Ctrl+u** - Delete from the cursor to the beginning of the line.
- Ctrl+r** - reverse search to bash history.
- !!** - repeats the last command.

## Process management

- ps** - display all active processes.
- top** - display all running processes.
- kill pid** - kill process id pid.
- killall proc** - kill all processes named proc.
- bg** - lists stopped or background jobs.
- bg n** - resume job n in the background.
- fg** - brings the most recent job to foreground.
- fg n** - brings job n to the foreground.

## Slurm Commands

- ▶ **sbatch** - submits a script job.
- ▶ **scancel** - cancels a running or pending job.
- ▶ **sinfo** - provides information on partitions and nodes.
- ▶ **sview** - GUI to view job, node and partition information (ssh -X option required).
- ▶ **smap** - CLI to view job, node and partition information.
- ▶ **squeue** - shows the status of jobs.
  - ▶ `squeue -l -j <job id>` gives extended job record information.
  - ▶ `squeue -u <user name>` lists only jobs initiated by the user.
- ▶ **sbcast** - transfer file to a compute nodes allocated to a job.
- ▶ **interactive** - opens an interactive job session.
- ▶ **sattach** - connects stdin/out/err for an existing job or job step.

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

## Slurm Submit Script Syntax

- ▶ `-A uoa99999` User account (i.e. nesiXXXXX, landcareXXXXX, uoaXXXXX,uocXXXXX)
- ▶ `--mem-per-cpu=8132` = memory/cpu (in MB)
- ▶ `-J My_Job_Name` Sets the job name
- ▶ `-o test-%j.%N.out` name of output file (default is slurm-jobid.out)
- ▶ `-e test-%j.%N.err` name of error file (default is slurm-jobid.err)
- ▶ `--mail-type=ALL` Specifies under which conditions Slurm will send out email notifications to the specified address about the state of the job (ALL, BEGIN, END, FAIL, REQUEUE).
- ▶ `--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` - Specifies the number of GPU devices requested.
- ▶ `--gres=mic:1` - Specifies the number of Intel Xeon Phi Co-Processor devices requested.
- ▶ `--ntasks-per-node=16` specifies the total number of tasks 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
- ▶ `--nodes=N-M` Defines the minimum (N) and the maximum (M) number of nodes.
- ▶ `-C sb` Requires a specific hardware architecture (sb=Sandybridge,wm=Westmere)
- ▶ `--exclusive` Requires exclusive execution

## 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] [-e]`

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

`-e`: binary that you want to run interactively

example : `interactive -A nesi99999`

## 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)
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
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)
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 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
srun binary_array $SLURM_ARRAY_TASK_ID
```

## User Environment

**LMOD** 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. The command line is fully compatible with the previous **Environment Modules**, and it provides simple short-cuts and advanced features.

Syntax: `module [options] sub-command [args ...]`

### Loading/Unloading sub-commands

- ▶ **load** | **add** load module(s)
- ▶ **del** | **unload** Remove module(s), do not complain if not found
- ▶ **purge** unload all modules
- ▶ **update** reload all currently loaded modules.

### Listing / Searching sub-commands

- ▶ **list** List loaded modules
- ▶ **avail** | **av** List available modules
- ▶ **avail** | **av string** List available modules that contain "string".
- ▶ **spider** List all possible modules
- ▶ **spider module** List all possible version of that module file
- ▶ **spiderstring** List all module that contain the "string".

### Short-cuts

- ▶ **ml** - means: module list
- ▶ **ml foo bar** - means: module load foo bar
- ▶ **ml -foo -bar baz goo-** means: module unload foo bar; module load baz goo;

More information at <http://www.tacc.utexas.edu/tacc-projects/lmod>