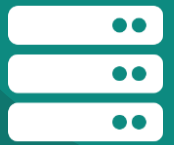
Good Practices Using HPC



Rāpoi Cluster — Victoria University of Wellington

Te Herenga Waka — Research Computing

Dr Muhammad Ali Hashmi
Research Computing Specialist

What We'll Cover

- Accessing the Cluster
- Storage & File Systems
- Environment Modules
- Running Jobs with Slurm
- Resource Requests
- GPU Computing
- Managing Jobs
- Python & R Environments
- Parallel Processing
- Containers
- Common Mistakes
- Best Practices

What is Rāpoi?

HPC Cluster

A collection of large servers (nodes) connected together with shared storage

Slurm Scheduler

Jobs are submitted to a scheduler that allocates CPU, memory, and time

Partitions

Different server/node groups: quicktest, parallel, gpu, bigmem, longrun

Network

Connected via 10G Ethernet for external connections and 52G InfiniBand for node-to-node communication

Tip: Most of the time you'll share a node with other users — request only what you need!

Cluster Hardware

AMD Parallel Nodes	GPU Nodes	Big Memory Nodes	Long Run Nodes
amd02n01-amd01n04 amd03n01-amd03n04 amd04n01-amd04n04 amd05n01-amd05n04 amd06n01-amd06n04 amd07n01-amd07n04	NVIDIA A100 GPUs gpu01, gpu02, gpu03 For CUDA/OpenCL workloads	high01–high04 Up to 1 TB RAM NVIDIA Tesla T4 gpu For memory-intensive work	bigtmp01–bigtmp02 30-day max runtime For extended computations

The CPU specs are more or less the same for all these nodes

2× AMD EPYC 7702
128 cores / 256 threads
512 GB RAM per node

Login Node: raapoi-login → Submit jobs → Scheduler assigns to compute nodes

Partitions Overview

Partition	Max Time	Nodes	Use Case
quicktest*	5 hours	amd01n01–04	Quick tests, builds, debugging, interactive sessions
parallel	10 days	amd02–amd07	General compute, most jobs go here
gpu	1 day	gpu01–03	GPU workloads (CUDA, ML)
bigmem	10 days	high01–04	Memory-intensive (up to 1 TB RAM)
longrun	30 days	bigtmp01–02	Extended computations

```
View partitions:  vuw-partitions     |     Node states: idle, mix, alloc, drain, down
```

Getting Started

1

Get an Account

Request via the Service Desk Portal
Provide: name, VUW username, faculty/school

2

Connect to VPN

Required from off-campus
Not required from Campus Staff WiFi
Use Cisco AnyConnect VPN client

3

SSH to Rāpoi

ssh username@raapoi.vuw.ac.nz
Port 22, VUW credentials

Tip: If hostname doesn't work, use IP address: 130.195.19.126

SSH Clients by Platform

Linux	Windows	macOS
Built-in terminal (GNOME Terminal, Konsole, xTerm)	Windows Terminal / PowerShell Git Bash MobaXterm (for GUI apps), Built-in SFTP transfer	Terminal.app (built-in) iTerm2 XQuartz (for X forwarding)
`ssh user@raapoi.vuw.ac.nz`	`ssh user@raapoi.vuw.ac.nz`	`ssh user@raapoi.vuw.ac.nz`

What is Unix shell?
https://cac-staff.github.io/swc-bash-lesson/
A Beginner's Guide to SSH Basics
https://blog.devops.dev/a-beginners-guide-to-ssh-what-it-is-and-how-to-use-it-27c118fec3d4

Storage & File Systems

Path	Quota	Backed Up?	Speed	Best For
`/nfs/home/<user>`	50 GB	Yes — replicated & backed up	Slow	Scripts, code, configs
`/nfs/scratch/<user>`	5 TB	No backup!	Medium	Working data, temp results
`/nfs/scratch/noquota-volatile`	100 TB shared	No — periodically deleted	Medium	Very large temp datasets
`/tmp (per node)`	1.7 TB	No — local only	Fastest (NVMe)	`For reading writing temporary files during I/O-heavy jobs`
`SoLaR Storage`	-	Yes — replicated & backed up	Slow	Storage for Learning and Research used for large scale research data, shared project spaces

Note: try to avoid writing thousands of very small files where possible

Check your usage: **`vuw-quota`**

Storage Best Practices

Home Directory

- Keep scripts and code here — it's backed up
- 50 GB limit cannot be increased
- Accidentally deleted files can be recovered via Service Desk

Scratch Space

- Use for active working data — not long-term storage
- Not backed up — your responsibility to maintain copies
- Clean up when done — shared resource for all users

Local /tmp

- Fastest storage (NVMe) — great for I/O-heavy jobs
- Only visible to the node running your job
- Copy data in at start, copy results out at end, clean up

Speed hierarchy: /tmp (NVMe) ▸ Scratch ▸ Home (slowest)

Environment Modules

Rāpoi uses lmod to manage software versions. The module command loads applications into your environment.

`module avail`	List all available software
`module spider <name>`	Search for a package & show prerequisites. You must spider on a specific version to see prerequisites
`module load <name>/<ver>`	Load a specific software version
`module list`	Show currently loaded modules
`module unload <name>`	Unload a module
`module purge`	Remove all loaded modules

Module Prerequisites & Tips

Prerequisite Chain Example

```
# Find what's needed
 module spider Python

# Then specifically
$ module spider Python/3.9.5


# Load prerequisites first

$ module load GCCcore/10.3.0

$ module load Python/3.9.5

$ module list
```

Key Tips

- Use module spider to find prerequisites — not just module avail
- Loading a toolchain (e.g., GCC/10.3.0) makes available additional modules
- Always include module purge, then module loads in your batch scripts for reproducibility

Running Jobs with Slurm

Interactive sessions • Batch jobs • Resource management

Interactive Jobs with srun

Use srun to get an interactive session on a compute node — great for testing, debugging, and development.

Basic Interactive Session

```
$ srun --pty bash
username@amd01n01:~$
```

With Custom Resources

```
$ srun -J MyTest \
  --cpus-per-task=2 --mem=4G \
  --time=0-05:00:00 --pty bash
```

- **Never run programs directly on the login node** — always request an interactive session
- Good for: writing programs, debugging, data transfer, quick tests, running anything involving a GUI
- Default resources: 2 CPUs, 2 GB RAM, 1 hour runtime
- Use --constraint="nodename" to request a specific node, or --nodelist=<nodename>

Batch Jobs with sbatch

Batch jobs run in the background — submit and forget. They start when resources are available.

```
#!/bin/bash

#SBATCH --job-name=my_job
#SBATCH -o output.out
#SBATCH -e error.err
#SBATCH --partition=parallel
#SBATCH --cpus-per-task=4
#SBATCH --mem=8G
#SBATCH --time=0-24:00:00

module purge
Module load GCCcore/10.3.0
module load Python/3.9.5

python my_script.py
```

← Job name for squeue

← stdout/stderr files.

these are not needed and that an output script will be created in the format slurm-<jobid>.out

← Target partition

← CPU & memory requests

← Max wall time (D-HH:MM:SS)

← Load modules with pre-requisites

← Your actual commands

Explanation of CPU related terms

`--ntasks` `(-n)`	The total number of independent processes (tasks) to launch across the entire job allocation. Commonly used for MPI parallel jobs.
`--cpus-per-task` `(-c)`	The number of CPU cores allocated to *each* individual task. Critical for multithreaded applications like OpenMP or pthreads.
`--nodes` `(-N)`	The total number of distinct physical server blocks (nodes) requested for the job.
`--ntasks-per-node`	Specifies exactly how many tasks should run on each allocated node.
`--cpus-per-node`	Directly requests a fixed number of CPU cores on each allocated node, regardless of task distribution.

Resource Requests

Other users are prevented from using resources you request — even if you don't use them!

CPUs

```
--ntasks=N
```

Default: 2

If you use more CPUs than requested, your job shares them and runs extremely slowly

Memory

```
--mem=NG
```

Default: 2 GB

If your job exceeds requested memory, Slurm kills it immediately

Time

```
--time=D-HH:MM:SS
```

Default: 1 hour

If your job exceeds wall time, Slurm kills it — request ~20% extra as buffer

Tip: Run a short test job first, then check actual usage with vuw-job-report <jobID>

GPU Computing

GPU Partition

- 3 GPU nodes: gpu01, gpu02, gpu03
- Max runtime: 1 day
- NVIDIA GPUs for CUDA/OpenCL workloads
- `Request with: --gres=gpu:1 --partition=gpu`

Example GPU Job

```
#!/bin/bash

#SBATCH --partition=gpu
#SBATCH --gres=gpu:1
#SBATCH --cpus-per-task=4
#SBATCH --mem=16G
#SBATCH --time=0-24:00:00

module load fosscuda/2020b
module load PyTorch/1.7.1

python train_model.py
```

PyTorch needs ~4 GB just to import — allocate sufficient memory or you'll get cryptic errors.
It is probably better to create a local PyTorch installation in a virtual environment for the latest version.

Managing Jobs

`vuw-myjobs` `(squeue -u <user>)`	View your running/pending jobs with resource details
`vuw-job-history`	Quick view of jobs completed in the last 5 days
`vuw-job-report <ID>` `(sacct -j <ID>)`	Detailed report: requested vs. actual memory/CPU usage
`scancel <jobID>`	Cancel a specific job
`scancel -u <user>`	Cancel all your jobs

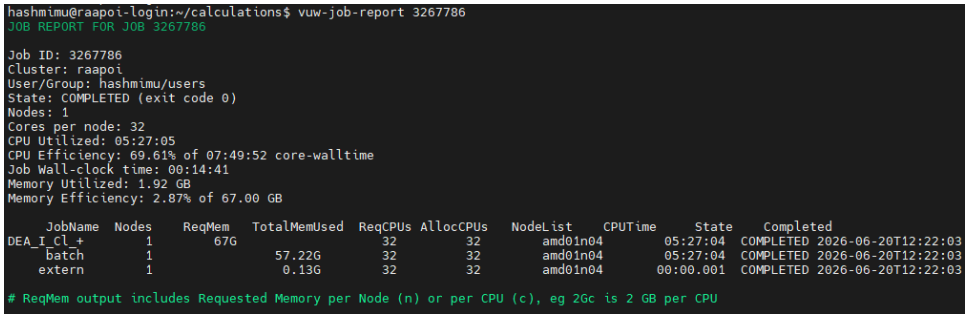
Checking Job Efficiency

Always review your job reports to avoid wasting resources!

```
$ vuw-job-report 162711

JobName      ReqMem    UsedMem    ReqCPUs   State

test-job     64 GB     0.15 GB    24        COMPLETED
```



The Problem

Requested 64 GB but only used 0.15 GB

= 63.85 GB wasted and unavailable to others!

* The UsedMem is not always very accurate. Take it as an estimate

The Fix

Run a short test first, check actual usage,

then request ~20% more than needed.

Use vuw-job-report after every job!

Python & Virtual Environments

Using Modules + venv

```
# Load Python via modules

module load GCCcore/10.3.0
module load Python/3.9.5

# Create a virtual environment

python3 -m venv myenv

source myenv/bin/activate

# Install packages
pip install numpy pandas
```

Key Considerations

- venv only works with the Python version it was created with
- Include module load commands in your batch scripts
- Activate the venv inside your submit script too
- Conda/Anaconda also available as modules
- Use scratch for large data — keep venv in home

R users: module spider R to find available versions. R notebooks and RStudio also supported.

Languages available: Python, R, Julia, MATLAB, Lua, and many more

Use of VSCode Server

Using VSCode Server on the Login Node is not allowed at all

Follow the instructions at Raapoi Docs to configure VSCode to run on a compute node

R users: module spider R to find available versions. R notebooks and RStudio also supported.

https://vuw-research-computing.github.io/raapoi-docs/usersub/#vscode

Parallel Processing & Advanced Topics

Job arrays • Multi-threading • MPI • GPU offloading

Job Arrays

Run many independent tasks simultaneously — perfect for parameter sweeps, processing multiple files, etc.

```
#!/bin/bash

#SBATCH -a 1-50

#SBATCH --cpus-per-task=1

#SBATCH --mem-per-cpu=2G

#SBATCH --time=0-00:10:00

#SBATCH --partition=parallel


module load fastqc/0.11.7

fastqc -o $TMPDIR/output \

  seqfile_${SLURM_ARRAY_TASK_ID}
```

How It Works

- -a 1-50 spawns 50 parallel tasks
- $SLURM_ARRAY_TASK_ID is 1, 2, ... 50
- Each task gets 1 CPU + 2 GB
- Total: 50 CPUs, 100 GB RAM
- Be mindful of cluster resource limits!

Submit with: `sbatch array.sh`

Types of Parallel Computing

Multi-threaded (Shared Memory)

Multiple threads share memory

OpenMP, Posix Threads

Use: --cpus-per-task=N

`Single node only`

Distributed Memory (MPI)

Processes across multiple nodes

Explicit message passing

Use: --constraint="IB"

`InfiniBand recommended`

GPU Offloading (CUDA/OpenCL)

Offload computation to GPU

Massive parallelism

Use: --gres=gpu:N

`gpu partition`

Need help designing your parallel workflow? Ask on the raapoi-help Slack channel!

Using Containers

Singularity containers let you run complex software stacks without installation headaches.

Interactive Container

```
srun --pty -c 4 --mem=64G bash

module load Singularity/3.10.2

singularity pull docker://image

singularity shell image.sif
```

Batch Container Job

```
#SBATCH -c 4 --mem=64G

#SBATCH --time=0-12:00:00


module load Singularity/3.10.2

singularity exec app.sif <your script>
```

- singularity shell — interactive use and testing
- singularity exec — run a specific command (for batch jobs)
- singularity run — execute the container's default command
- Pull Docker images directly: singularity pull docker://name

Jupyter Notebooks on Rāpoi

1

Copy a submission script

Example scripts at:
/home/software/vuwrc/
examples/jupyter/

Options: bare, anaconda, R kernel

2

Submit the job

sbatch notebook.sh

The script starts a Jupyter server on a compute node and outputs connection details

3

SSH port forward

ssh -L <port>:localhost:<port> \
user@raapoi.vuw.ac.nz

Open browser at localhost:<port>

Scripts available: notebook-bare.sh, notebook-anaconda.sh, R-notebook.sh, R-notebook-anaconda.sh

Data Transfer & Research Storage

SCP & RSYNC

- scp file.dat user@raapoi:~/scratch/
- rsync -avz data/ user@raapoi:~/scratch/data/
- Use rsync for large transfers (resume on failure). MobaXterm has built-in SFTP.

Cloud (rclone)

- module load rclone/1.54.1
- Sync to AWS S3 or other cloud providers
- Use tmux to keep long transfers alive if you disconnect

SoLAR Research Storage

- Digital Solutions managed backup storage
- Back up scratch data here for safety
- Replicated off-site; ideal for long-term data archiving

Tip: Use tmux for long transfers. Do visualisation locally — transfer results to your machine.

Common Mistakes to Avoid

Don't Do This	Do This Instead
Running programs on the login node	Always use srun or sbatch
Requesting far more resources than needed	Test first, then request ~20% extra
Not checking job reports after completion	Use vuw-job-report to review usage
Storing important data only on scratch	Scratch is not backed up — copy to home or external
Forgetting to clean up /tmp after jobs	Copy results out, remove temp files
Not including module loads in batch scripts	Always load modules in your submit script
Install software locally that already exists	Check what is available using `module spider`

Best Practices Summary

Resources

- Test short jobs first to estimate needs
- Request ~20% more than actual usage
- Review vuw-job-report after every job

Storage

- Code in /home, data in /scratch
- Use /tmp for I/O-heavy operations
- Back up scratch data externally

Jobs

- Include all module loads in scripts
- Use job arrays for parallel tasks
- Set --mail-type for notifications

Etiquette

- Don't hog the login node
- Clean up scratch when done
- Ask on Slack before opening a ticket

Remember: Rāpoi is a shared resource — use it wisely and considerately!

Quick Reference: Key Commands

Access	`ssh -X user@raapoi.vuw.ac.nz`
Info	`vuw-partitions`
Storage	`vuw-quota`
Modules	`module spider <name>`
Interactive	`srun --pty bash`
Batch	`sbatch submit.sh`
Monitor	`vuw-myjobs`
Report	`vuw-job-report <ID>`
Cancel	`scancel <jobID>`
Disk	`df -h | grep scratch`

Full documentation: vuw-research-computing.github.io/raapoi-docs

Getting Help

Slack Channel (scan the QR code)

raapoi-help — first stop for questions

Documentation

vuw-research-computing.github.io/raapoi-docs

Service Desk

Account requests & file recovery

Please contact the Rāpoi support team — not Digital Solutions — for HPC issues