Accelerating Research: NCI GADI GPU Insights

Research Seminar: Optimizing the Use of Computing Resources

Qixiang Chen[1]

Australian National University

December 10, 2023

[1]Qixiang Chen, honored with The Active Intelligence Research Challenge Award, serves as a Research Intern at Active Intelligence Corp. Additionally, he is an honours research student supervised by Lei Wang (ANU & Data61/CSIRO).

Table of Contents

Introduction

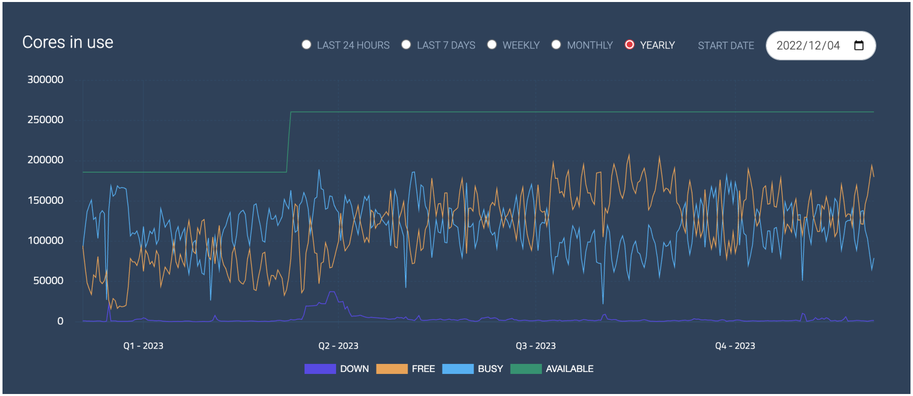
HPC system - Gadi

Gadi contains $>$ 250,000 CPU cores, 930 Terabytes of memory & 640 GPUs.

- 160 nodes: 4 Nvidia V100 GPUs, 2 Intel Xeon processors each.
- Mellanox Technologies' HDR InfiniBand for 200 Gb/s data transfer.
- More than 200 supported software packages.
- PBS (Portable Batch System) is responsible for optimising the job scheduling and managing the workload of the cluster.

HPC system - Gadi (cont.)

Make sure the efficient use of Gadi!



Access to GADI - Two Approaches

- **Prerequisites**

  Ensure OpenSSH is installed on your local machine.

1. **Standard Terminal Access**

   ```
   $ ssh yourAcc@gadi.nci.org.au
   ```

2. **SSH Config for Quick Access**

   1. Generate an SSH key on your local machine.
   2. Append the following lines to the SSH config file in `~/.ssh/config`

      ```
      Host gadi
        HostName gadi.nci.org.au
        User yourAcc
        IdentityFile ~/.ssh/id_ed25519
      ```

   3. Set up public key authentication on Gadi:

      ```
      $ ssh-copy-id -i ~/.ssh/id_ed25519.pub gadi
      ```

   4. Afterwards, access Gadi without entering a password:

      ```
      $ ssh gadi
      ```

Access to GADI - Login Nodes

```
<welcome message: Mesage of the Day>
============================================================
Last login: Wed Nov 22 17:00:04 2023 from 150.203.68.254
[qc2666@gadi-login-05 ~]$ pwd
/home/135/qc2666
[qc2666@gadi-login-05 ~]$ cat $HOME/.config/gadi-login.conf
PROJECT xj17
SHELL /bin/bash
```

- Processes exceeding 30 minutes of CPU usage or 4 GiB of memory will be terminated.
- Every user have 10 GiB in Home directory.
- With backups in `$HOME/.snapshot`

Environmental Settings

Environment Modules

Environment Modules[2] provide packages/licenses in multiple versions, allowing users to easily switch between them.

- **To look for specific Applications/Licenses:**
```
$ module avail <app_name>
```
- **To load/unload a module:**
```
$ module load <app_name>/<version>
$ module unload <app_name>/<version>
```
- **To list all modules:**
```
$ module list
```

[2] Check here for more commands.

Conda/Virtual Environment

Conda/Virtual environments facilitate the efficient management of specific dependencies, accommodating those with low data storage demands or those not included in the modules.

For Conda:

1 **Install a specific version of Anaconda/Miniconda:**

```
$ wget <URL_Here>
$ bash Anaconda3-xx.sh
```

2 **Create the conda environment and install necessary packages[3].**

* Export all packages from a conda environment.

```
  $ conda list --explicit > pkgs.txt
```

* Install packages in a new conda environment.

```
  $ conda create --name <NEWENV> -- file pkgs.txt
```

[3]check here for more information.

Conda/Virtual Environment (cont.)

For Virtualenv:

1 **Install[4] the package and make directory.**
```
$ pip3 install virtualenvwrapper
$ mkdir ~/.virtualenvs
```
2 **Edit the** `.bashrc` **file by appending:**
```
export WORKON_HOME=$HOME/.virtualenvs
source $HOME/.local/bin/virtualenvwrapper.sh
```
3 **Run** `$ source ~/.bashrc`

4 **Load the desired python version and create the environment.**
```
$ module load python3/3.8.5
$ mkvirtualenv --system-site-packages <NEWENV>
```
5 **Run** `$ pip install <package>` **to install the necessary packages.**

6 **Run** `$ workon <NEWENV>` **to activate.**

[4] check here for more information.

Job Submission & Monitoring

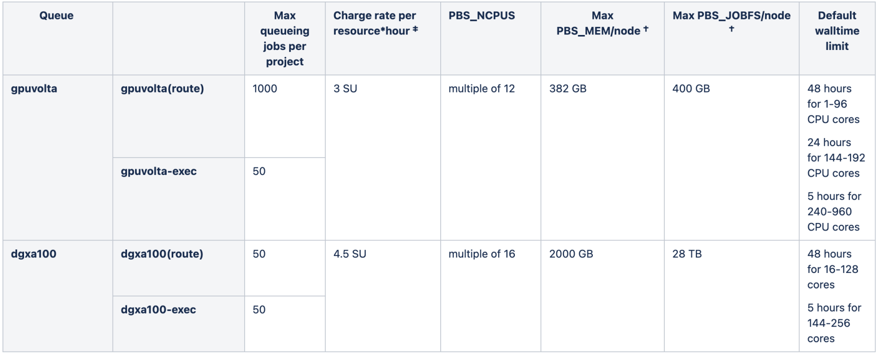
Submission Script

Sample Script[5]:

```
#!/bin/bash
#PBS -P xj17
#PBS -q gpuvolta
#PBS -l ngpus=1
#PBS -l ncpus=12
#PBS -l mem=16GB
#PBS -l walltime=00:05:00
#PBS -l wd
#PBS -l storage=gdata/xj17+scratch/xj17
cd /g/data/xj17/qc2666/demo
module load pytorch/1.10.0

# Activate Conda
# NOTE: Replace <ENV> with your actual conda environment name
#export CONDA_ENV='/home/135/qc2666/miniconda3/bin/activate'
#source $CONDA_ENV <ENV>

# Activate Virtualenv
# NOTE: Replace <ENV> with your actual virtualenv environment name
#export VIRTUAL_ENV='/home/135/qc2666/.virtualenvs/<ENV>/bin/activate'
#source $VIRTUAL_ENV

python3 main.py
```

[5] check here for more PBS Directives Explained.

Queue Types



Queue		Max queueing jobs per project	Charge rate per resource*hour ‡	PBS_NCPUS	Max PBS_MEM/node †	Max PBS_JOBFS/node †	Default walltime limit
gpuvolta	**gpuvolta(route)**	1000	3 SU	multiple of 12	382 GB	400 GB	48 hours for 1-96 CPU cores 24 hours for 144-192 CPU cores 5 hours for 240-960 CPU cores
	gpuvolta-exec	50					
dgxa100	**dgxa100(route)**	50	4.5 SU	multiple of 16	2000 GB	28 TB	48 hours for 16-128 cores 5 hours for 144-256 cores
	dgxa100-exec	50					

Submit jobs

- **Run** `$ nci_account` **to check the available resources and storage allocations.**
- **Run** `$ qsub <submit_job.sh>` **to submit the job.**
- **Run** `$ qdel <job_id>` **to delete the job.**
- **Run** `$ for file in submit_job*.sh; do qsub $file; done` **to submit multiple job scripts start with** `submit_job`**.**

* **Running** Array jobs.

Compute Grant and Job Debiting

$$\begin{aligned}\textbf{Job Cost (SU)} &= \textbf{Queue Charge Rate}^{6}\\ &\times \textbf{Max [NCPUs, Memory Proportion]}\\ &\times \textbf{Walltime Used (Hours)}\end{aligned}$$

Memory Proportion = Mem requested/Mem per core
Mem per core = Mem Per Node/NCPUs per node for queue

Queue	CPUs request	GPUs request	Mem request	Walltime Usage	Cost
gpuvolta	12	1*	90GB	5 hours	$3 \times 12 \times 5 = 180$ SU
gpuvolta	12	1*	380GB	5 hours	$3 \times 12 \times \max[1, (380/12) \times (48/382)] \times 5 = 3 \times 12 \times \max[1, 3.97905...] \times 5 = 716.23$ SU (rounded)

[6] Charge rate of **gpuvolta** is 3 SU per hour, check here for more information.

Monitoring Commands

- **Check the status of a job:**
  For a simple check: `$ qstat`
  For detailed information:
  ```
  $ qstat -swx <job_ID>
  $ qstat -Esw
  ```
- **Job state**
  - Q - job is queued, eligable to run or routed.
  - R - job is running.
  - E - Job is exiting after having run.
  - H - Job is held.
  - F - job is finished.

Monitoring Commands (cont.)

- **Check processes inside a job:**

  ```
  $ qps <job_ID>
  ```

- **Check the status of CPU and memory of a job:**

  ```
  $ nqstat_anu <job_ID>
  ```

- **Check the queue of the project:**

  ```
  $ nqstat
  ```

- **Access the GPU node for expanded monitoring capabilities.**

  `$ qstat -n1` to check the node.

  ```
  $ ssh gadi-gpu-v100-0097
  ```

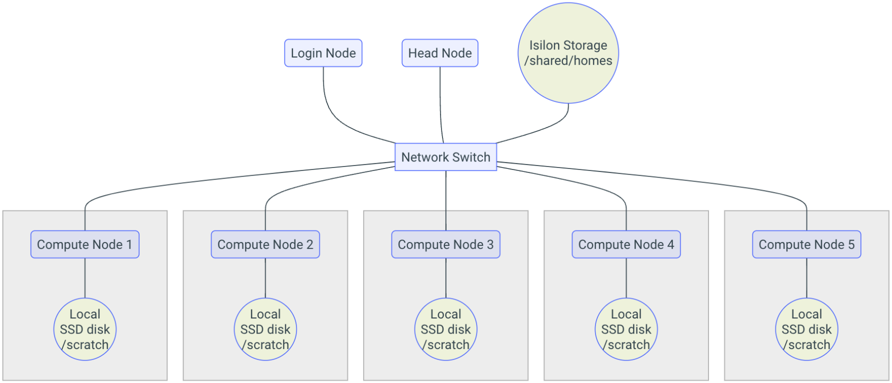
Interactive Jobs

```
[qc2666@gadi-login-03 qc2666]$ qsub -I -qgpuvolta -Pxj17
-lwalltime=01:00:00, ncpus=12,ngpus=1,mem=8GB,
storage=gdata/xj17+scratch/xj17
qsub: waiting for job 103392320.gadi-pbs to start
qsub: job 103392320.gadi-pbs ready

[qc2666@gadi-gpu-v100-0006 qc2666]$ module list
Currently Loaded Modulefiles:
1) pbs
[qc2666@gadi-gpu-v100-0006 qc2666]$ exit
logout
```

Interactive Login Using a Job Script

```
#!/bin/bash

# This is a simple PBS script that uses the "interactive" mode of PBS.
# See "Interactive-batch Jobs" in the "PBS User Guide".
#
# For interactive use you must submit this job with -I
# qsub -I this_script.sh

#PBS -P xj17
#PBS -q gpuvolta
#PBS -l ngpus=1
#PBS -l ncpus=12
#PBS -l mem=8GB
#PBS -l walltime=01:00:00
#PBS -l storage=gdata/xj17+scratch/xj17

# Note: don't have any other commands below here!
```

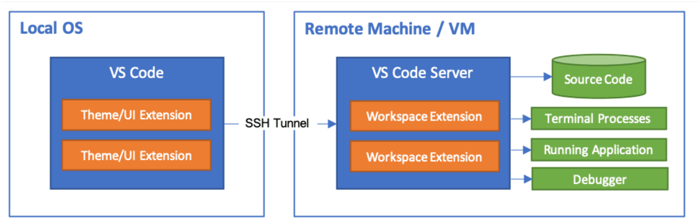
HPC Hardware Layout

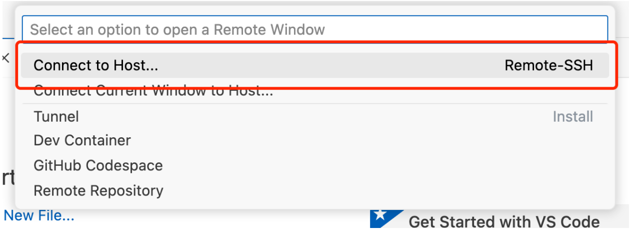


Operating/Debug Interface

Remote Development using SSH

VScode or PyCharm




Select an option to open a Remote Window
Connect to Host... Remote-SSH
Connect Current Window to Host...
Tunnel Install
Dev Container
GitHub Codespace
Remote Repository
New File...
Get Started with VS Code

Remote debug: Jupyter notebook

1 **Submit a interactive job to access GPU node.**

```
$ qsub -I access_gpu.sh
```

2 **Load or activate the necessary dependencies.**

3 `cd` **to the working directory.**

4 **Check the GPU node and Run**

```
$ jupyter-notebook --no-browser --ip=gadi-gpu-v100-0097 >
notebook-output 2>&1 &
```

5 **Run** `$ jupyter notebook list` **to check the port & token.**

6 **Run**

```
$ ssh -N -f -L 127.0.0.1:8888:gadi-gpu-v100-0097:<port>
yourAcc@gadi.nci.org.au
```
[7] **on your local machine.**

7 **Type** `127.0.0.1:8888` **in the browser**[8] **and enter the token.**

[7] You can replace the ssh address with the configured hostname, for example `gadi` on page 6.
[8] Please verify the GPU node, port, and token on the remote node, and ensure that a corresponding port is available on your local machine

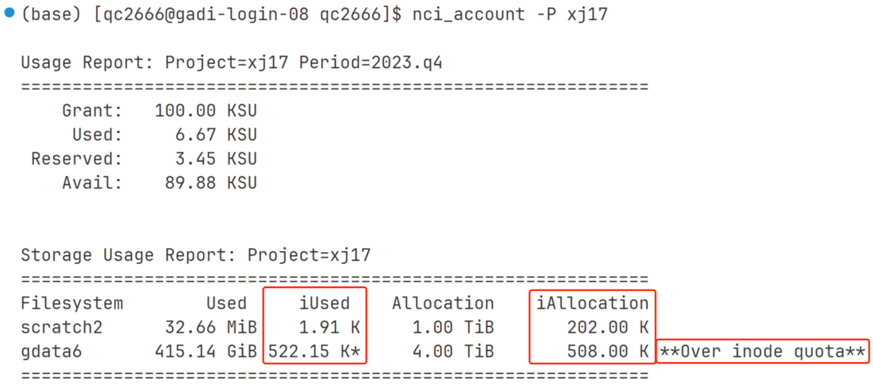
Code & Data Management

File Transfer to/from Gadi

1. **Standard SCP Transfer:**
   - Command: `$ scp <source> <destination>`
   - Example: `$ scp input.data gadi:/home/777/aaa777`
2. **Resumable Transfer with Rsync:**
   - Command: `$ rsync -avPS <source> <destination>`
   - Example: `$ rsync -avPS gadi:/scratch/a00/aaa777/test_dir ./`
3. **For instructions on transferring files larger than 50 GB, please refer to the detailed job submission guide.**
4. **For effective code management, it is recommended to use GitHub.**

Dataset Management

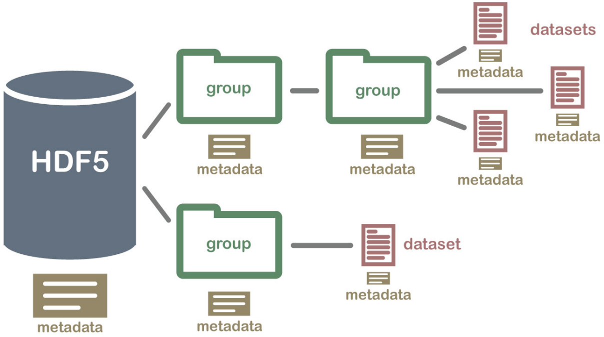
- **To view the storage quota details:**
  `$ lquota`
- **For detailed information on the quotas of the group & users:**
  `$ nci-files-report -g <Project_ID>`



```
(base) [qc2666@gadi-login-08 qc2666]$ nci_account -P xj17

Usage Report: Project=xj17 Period=2023.q4
==============================================================
    Grant:   100.00 KSU
     Used:     6.67 KSU
 Reserved:     3.45 KSU
    Avail:    89.88 KSU


Storage Usage Report: Project=xj17
==============================================================
Filesystem        Used      iUsed    Allocation    iAllocation
scratch2     32.66 MiB     1.91 K      1.00 TiB       202.00 K
gdata6      415.14 GiB   522.15 K*     4.00 TiB       508.00 K **Over inode quota**
==============================================================
```

Dataset Management (cont.)

HDF5 (Hierarchical Data Format, Version 5), a versatile library and file format designed for storing scientific data, is recommended as a solution to address inode limits issues.



Our AI flagship HPC funding scheme has been successful.

- **Research grant:** The NCI National AI Flagship Merit Allocation Scheme
- **Project title:** Robust anomaly detection in human-centric videos
- **Investigators:**
  - Lead CI: Lei Wang
  - Researchers: Qixiang Chen, Xiuyuan Yuan, Liyun Zhu, Arjun Raj, Liwen Luo
- **Research period:** 2024/01/01-2024/06/30

Thank you!