Notes from NSCC Workshop

Shrinjana Ghosh

15$^{\text{th}}$ January, 2025

Link to slides: Slides

HPC Introduction

- HPC - High Performance Computing
- Measured in FLOPS (floating point operation/sec)

  No. of arithmetic operations/sec
- Parallel processing enabled
- Login node - access point to interact with HPC cluster
- Compute node - performs the actual computations inside the HPC cluster
- Usable in various domains - AI, ML, climate modelling, weather predictions, energy preserving simulations - anything that requires a lot of data.

Aspire 2A architecture

- Improved from Aspire 1 with more cores, more GPUs and more compact - hence more energy-friendly.
- Specs

  800 CPU Nodes

  16 high-freq nodes

  82 NVIDIA A100 GPU nodes

  476TB total system memory

  25Pbytes storage

  10Pbytes scratch disk
- Located in NUS i4.0 Lvl 2 (lol can i try visiting this)
- For me: always use the NUS login node and NUS VPN (using Ivanti Client) - try using WinSCP for FTP
- Slingshot HSN - proprietary system
- Purpose of login node

  Only for lightweight workload

  Only for accessing HPC cluster

  Installing software, transfering files, submitting jobs, monitoring the jobs

  Avoid performing any computational/analysis jobs on the login node - **always use the compute node for this purpose**

  Direct access nodes: aspire2a.nus.edu.sg

  Windows ssh client: MobaXterm, putty (the latter is currently in use)

Storage

- Lustre

  100 TB/person

  30 days purge policy

  Important data must be backed up to the project directory/folder.

- GFPS

  This is only for project folder - the retention depends on the project expiry date.

- Best Practices for using Scratch and Project directories

  Input data from project directory to scratch directory

  Execute HPC job for read/write on scratch directory; use Lustre I/O performance

  Transfer generated data back to scratch directory

- Check home quota

- Check project quota

File Transfer

- FileZilla for GUI transfer

- WinSCP: use SSH client for secure transfers

  scp /path/to/sourcefile username@destination: /path/to/destination

- Transferring within the file system

  Start screen session

  Submit interactive job:

  Start transfer:

- Screen - terminal multiplexer that allows us to start a terminal session and allows us to keep it running even when we disconnect.

- Screen commands - given in the slides.

- I'm too bored to take notes now - check slides for the rest of the notes. I'll only write anything noteworthy here - no more running notes. Instructor's damn boring.

Module Commands

- All given in the slides. Important ones below:

  module list

  module avail

  module load <package name>

Programming Environments

- Cray - default

- Preferably use GNU environment

- Cray Compiling Environment provides wrappers for Fortran, C, C++ via ftn, cc and CC.

- Use CC or cc to compile C++ code (ugh this language is back again).

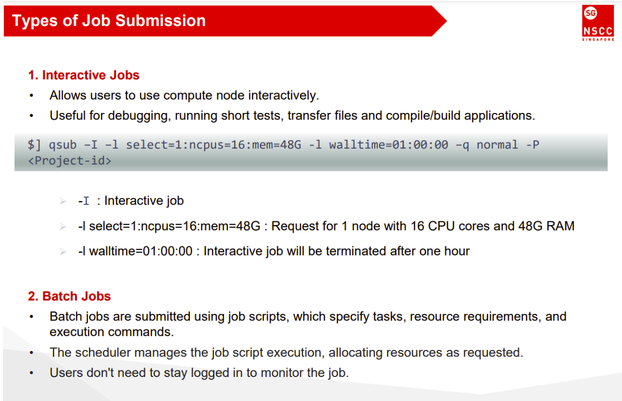

Figure 1: Types of job submission

Compute Nodes

- Cray EX CPU
- Large MEM
- Cray EX GPU
- AI nodes
- PBS queue for all except the last one is <normal>. The one for **AI nodes** is <ai>
- Register for additional course on their website for visualization purposes - VMD Training

Service Unit Allocation

- SU - currency to utilize resource on HPC
- Charged based on requested CPU and GPU resources

  1 SU/1 CPU core hour

  64 SU/1 GPU core hour
- One-time personal quota: stakeholders (like me) get 100k SU.
- Fixed, non-transferrable and cannot be topped up - after depletion, can submit job only through an approved project.
- SU utilization can be checked through <myusage>command.
- Rest commands can be checked via the commands on the slides.
- SU allocation after depletion based on accepted projects through **Call for projects** when NSCC does this.

Job Scheduling and Package Management

- PBS - portable batch system
- Job scheduler - software to manage the workload - for allocating resources and budget it accordingly
- Allows user to submit batch and interactive jobs
- This scheduler manages the most optimized usage of the submitted jobs at a given point in time.
- Process flow given in the slides.
- Types of job scheduling given in Figure 1 above.
- Syntax for batch job given below in Figure 2.

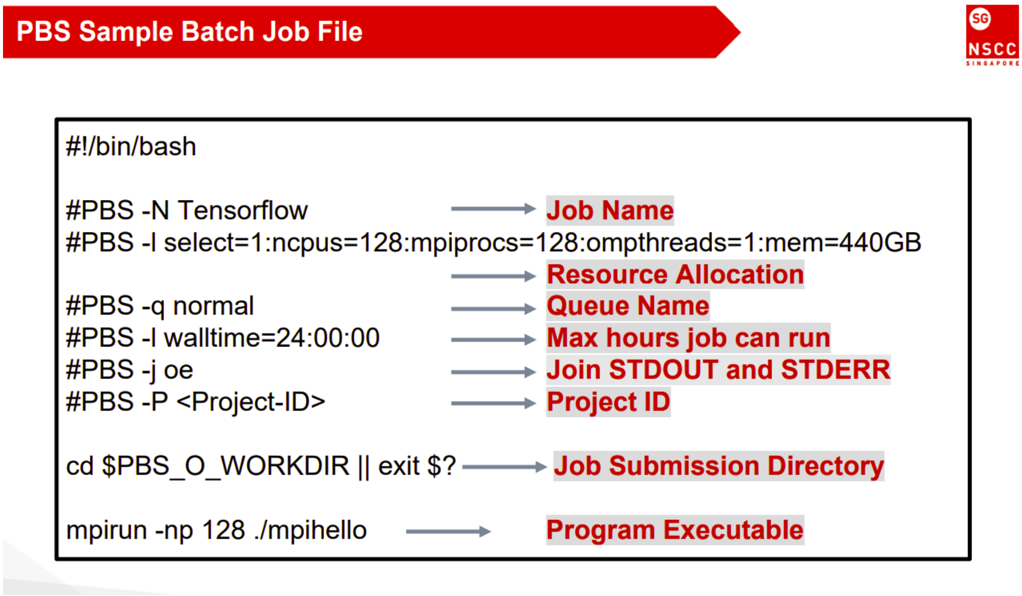

Figure 2: Syntax for batch job

- **Containers** - only supports <Singularity>containers.
- **Miniforge** - creating venvs inside the HPC clusters - it is a lightweight Conda installer.

Usage Best Practices

- **Optimising and Scaling your workloads**

  Start with a small job - allows efficient utilization

  Always monitor your jobs using <top>and <nvidia-smi>for CPU and GPU workload respectively.

  Try different configs and figure out the best parallel efficiency for future workloads - by incremental scaling.

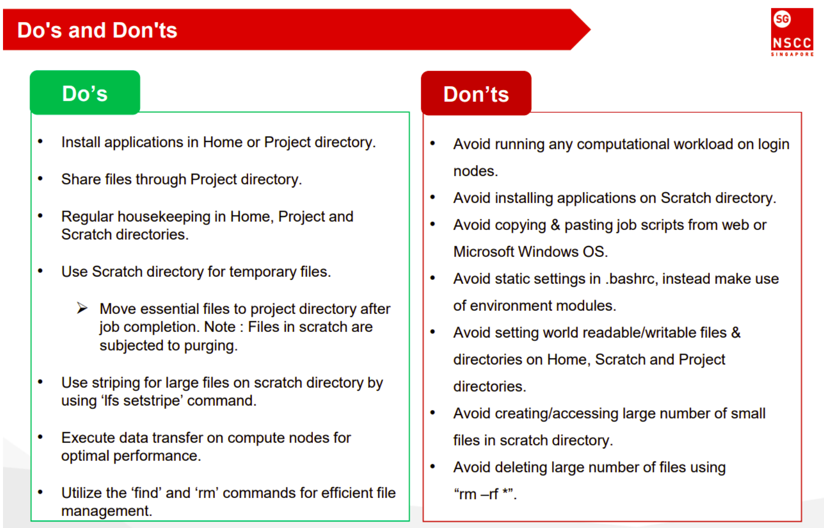

Figure 3: Best practices