

HPC and cluster computing best practices



RICE | CTBP
Center for Theoretical Biological Physics

<https://github.com/Whitford/ctbp-techtalks>

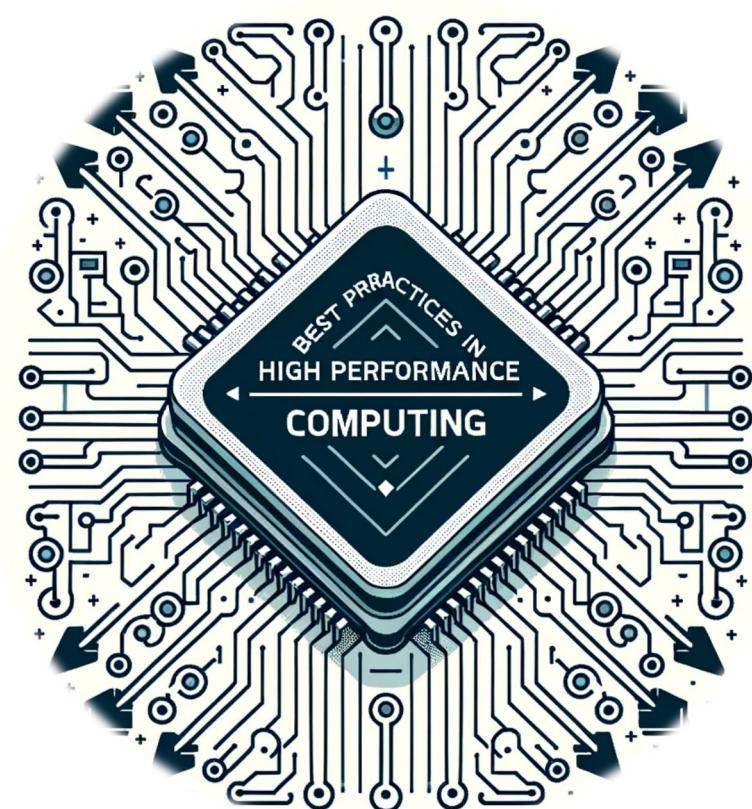
Vinícius Contessoto

contessoto@rice.edu – Nov/2023



Outline

- HPC - 5W and 1H
 - What, Who, Where, When, Why, and How
- Software Stack
- Accessing the Cluster
 - Vscode and GitHub Desktop
- Slurm – Job Management
 - Run a few examples
 - Job array, Job dependency



HPC – High-Performance Computing

W1 - What is HPC?

High-Performance Computing involves the usage of supercomputers/clusters and parallel processing techniques for solving complex computational problems at high speeds.



HPC – High-Performance Computing

W2 - Who uses HPC?

Scientists, researchers, engineers, and professionals across various domains such as healthcare, finance, manufacturing, and academia.



W3 - Where is HPC used?

HPC is used in universities, research institutions, government labs, and industries worldwide.



HPC – High-Performance Computing

W4 – When is HPC needed?

HPC is required when the computational needs exceed the capabilities of standard computers, especially for tasks that involve large-scale simulations, data analysis, and complex calculations.



W5 – Why is HPC important?

HPC is crucial for advancing scientific research, optimizing industrial processes, improving product design, predicting natural disasters, and much more.



HPC – High-Performance Computing

H1 – How does HPC work?

HPC systems use parallel processing, where multiple processors work simultaneously to perform computations quickly. They also utilize advanced networking, high-speed storage systems, and optimized software to handle large datasets and perform complex simulations.



Supercomputer

By Jitze Couperus

By OLCF at ORNL



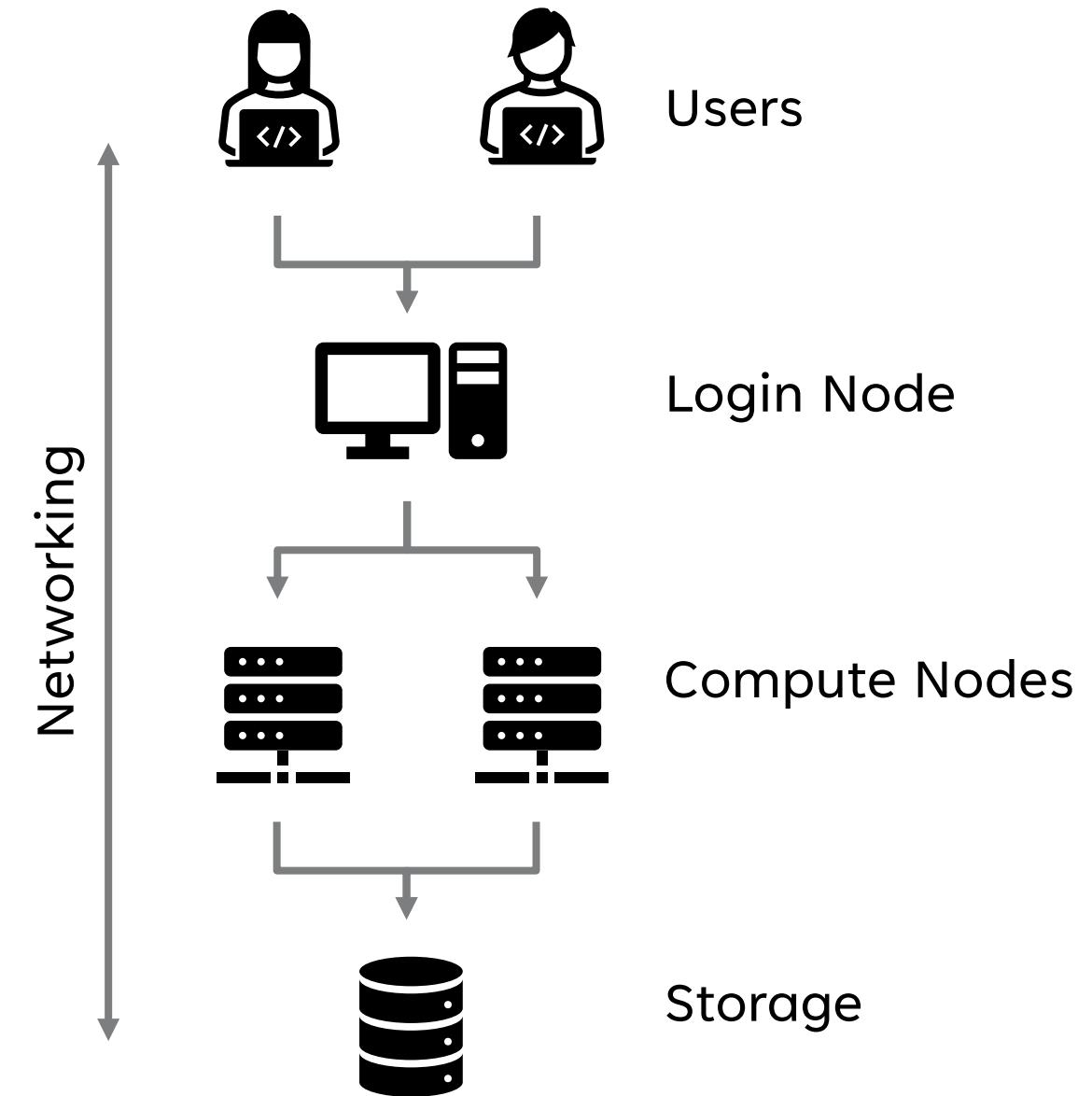
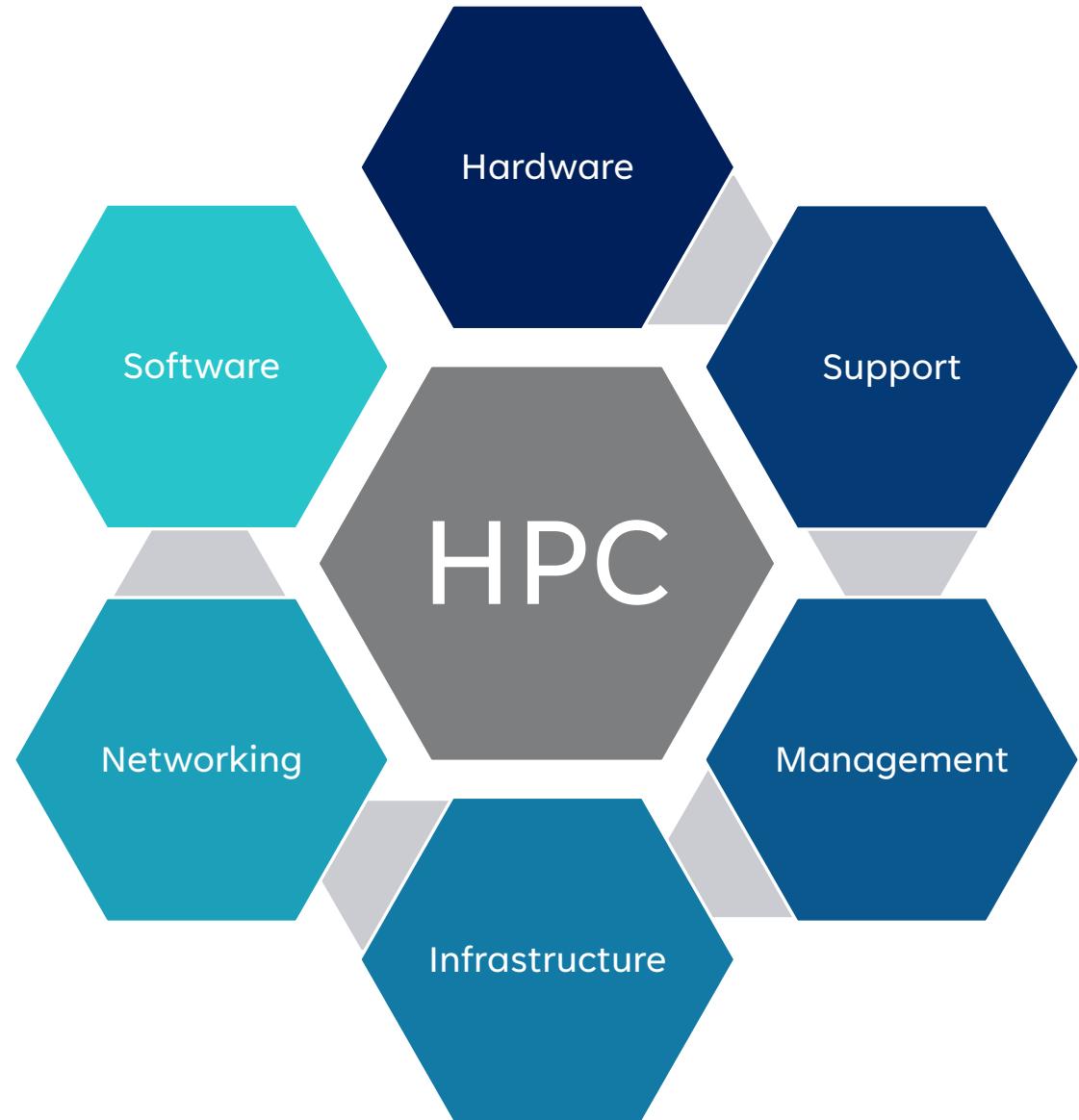
CDC 6600 – 1964 - 3 megaFLOPS



Frontier – 2022 - 1.102 exaFLOPS

9,472 [AMD Epyc 7453s](#) (606,208 cores) and
37,888 [Radeon Instinct MI250X](#) GPUs (8,335,360 cores)

HPC – High-Performance Computing



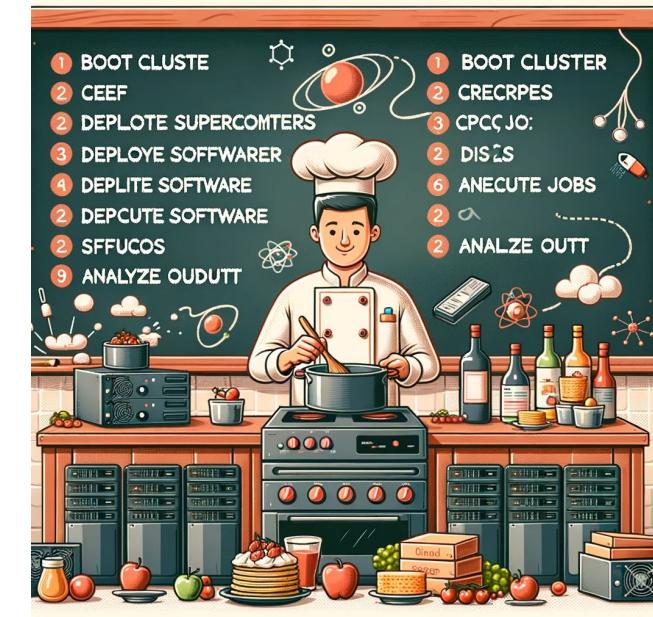
HPC Software Stack

- **Operating Systems:** Predominantly, HPC clusters run on Linux.
- **Schedulers and Resource Managers:** Most commons are Slurm and Torque/PBS. Manages the job queue and ensuring equitable and efficient resource distribution.
- **Parallelization Tools:** MPI handles inter-node communication, and OpenMP takes care of intra-node parallelism.



Recipe: Running a Job in the HPC Cluster Soup

- 1 Laptop, preferably lightweight, with a fresh sprinkle of SSH installed
- 1 Stable internet access, preferably high-speed.
- 1 Large Cluster, ripe and ready for computation
- A pinch of patience (sometimes more, depending on the queue)
- 1 Job script, marinated in correct syntax
- A dash of debugging skills (just in case)
- Optional: 1 cup of coffee or tea for those longer jobs



Accessing the Cluster – NOTS (Night Owls Time-Sharing)

Terminal (Rice Network or VPN)

```
ssh -Y (your_login_name)@nots.rice.edu
```

Gateway Access First

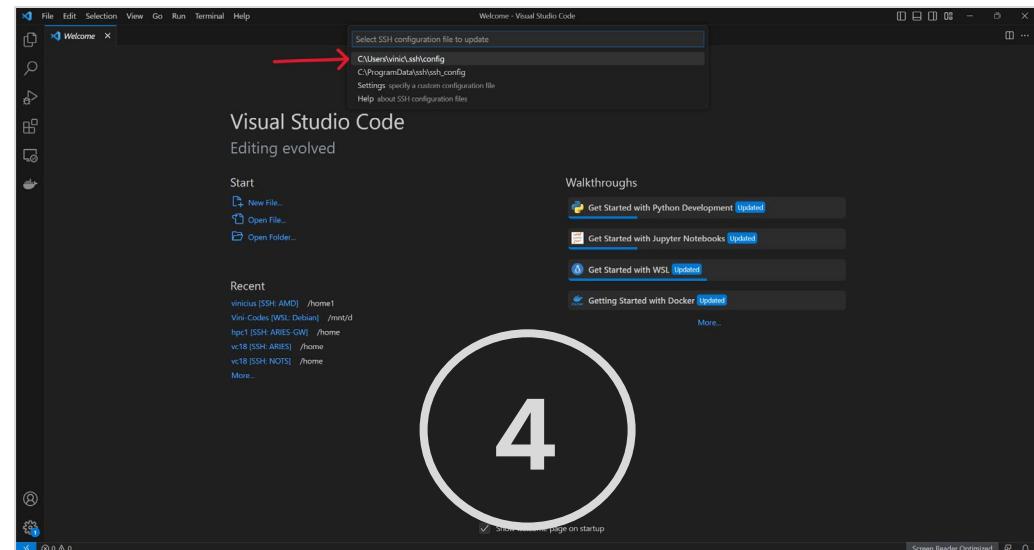
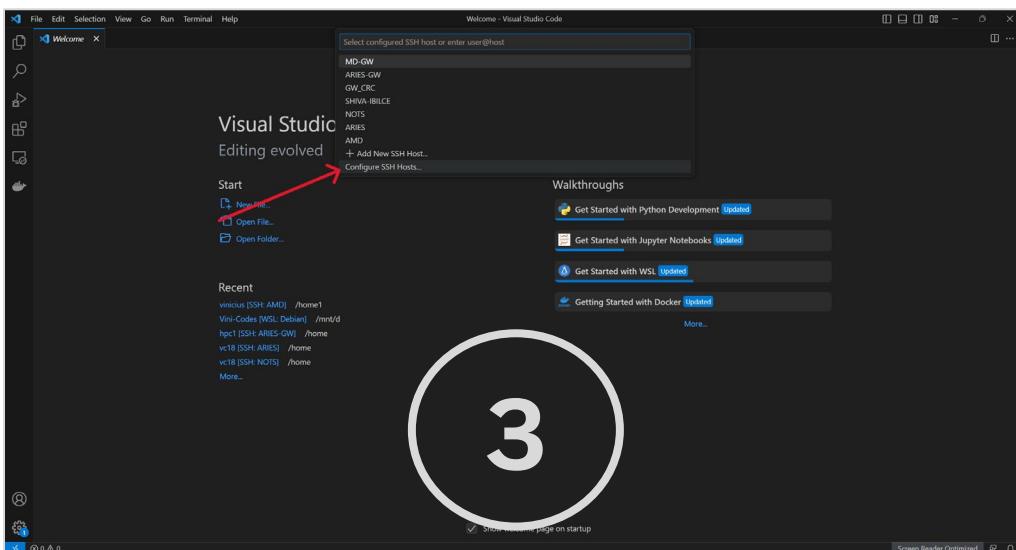
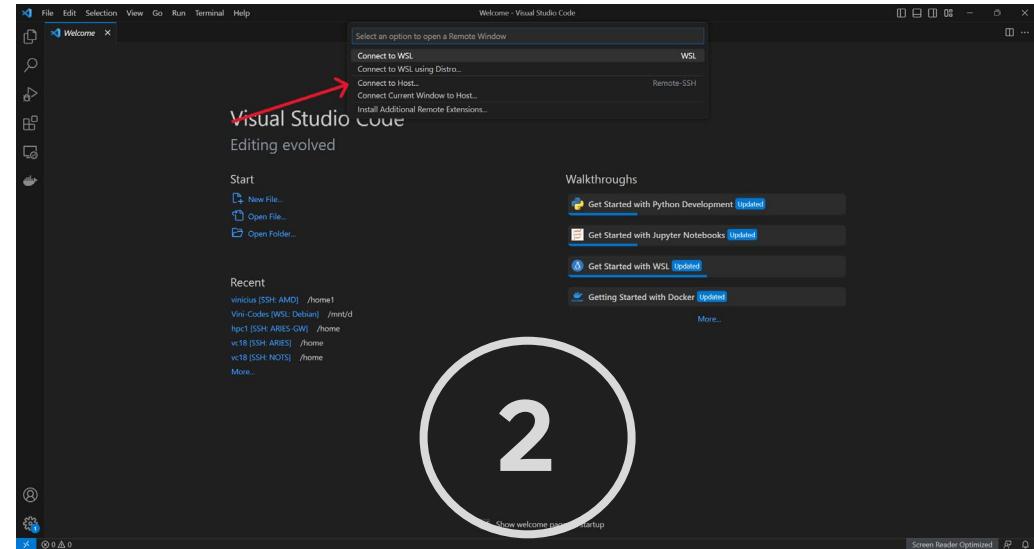
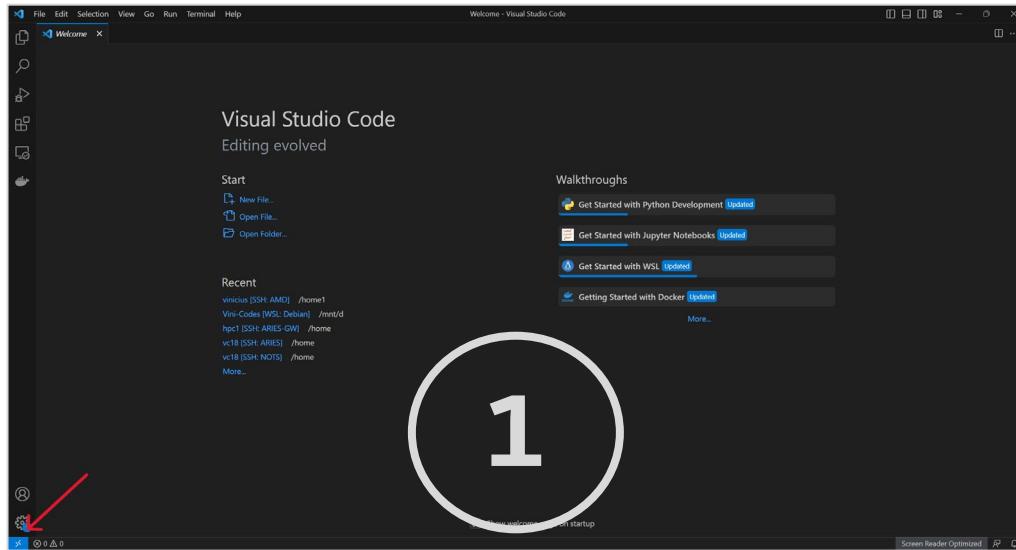
```
ssh -Y (your_login_name)@gw.crc.rice.edu
```

Accessing the Cluster – NOTS (Night Owls Time-Sharing)

VScode – Visual Studio Code

- **Download at:** <https://code.visualstudio.com>
- **Install the Remote - SSH Extension:** Go to Extensions and search for “Remote - SSH” and install it.
- **Connect to Your Cluster:** Press Ctrl+Shift+P, type “Remote-SSH: Connect to Host...”, and then enter ssh –AY username@cluster-address.

VScode – Visual Studio Code – Cluster access



VScode – Config file Example

```
Host GW_CRC
  User vc18
  HostName gw.crc.rice.edu
  ForwardAgent yes
  IdentityFile C:\\\\Users\\\\vini\\\\.ssh\\\\id_rsa

Host NOTS
  HostName nots.rice.edu
  User vc18
  Port 22
  ForwardAgent yes
  ProxyJump GW_CRC
  IdentityFile C:\\\\Users\\\\vini\\\\.ssh\\\\id_rsa

Host ARIES
  HostName aries.rice.edu
  User vc18
  ForwardAgent yes
  ProxyJump GW_CRC
  Port 22
  IdentityFile C:\\\\Users\\\\vini\\\\.ssh\\\\id_rsa
```



GitHub Desktop - Download Files

GitHub Desktop

- **Download at:** <https://desktop.github.com>
- **Clone the Repository:** <https://github.com/Whitford/ctbp-techtalks>
 - Go to the green button “ <> Code” -> Open with GitHub Desktop
- **Transfer Data to the Cluster:** Copy & Paste the folder **Intro_HPC** to home folder in cluster

Some useful command lines

```
sacctmgr show associations user=username  
  
sinfo -p ctbp-onuchic  
  
squeue -p ctbp-onuchic  
  
quota -s  
  
module avail  
  
module spider Anaconda3/2022.10
```



Running Our First Job

```
#!/bin/bash
#SBATCH --job-name=job_1
#SBATCH --account=ctbp-onuchic
#SBATCH --partition=ctbp-onuchic
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --time=00:05:00
#SBATCH --mem-per-cpu=100M
#SBATCH -e slurm-%j-%x.err
#SBATCH -o slurm-%j-%x.out

echo "Job ID: $SLURM_JOB_ID"
echo "Hostname: $(hostname)"
echo "Memory Usage: $(free -h)"
echo "Node: $SLURM_NODELIST"
echo "CPUs: $SLURM_CPUS_ON_NODE"
echo "Threads: $(nproc --all)"
echo "Number of Nodes: $SLURM_NNODES"
echo "Number of Tasks: $SLURM_NTASKS"
echo "Cores/CPUs per Node: $SLURM_CPUS_ON_NODE"
echo "Runtime Duration: $((($SECONDS / 60)) minutes and $($SECONDS % 60)) seconds."
```

Running Our First Job

```
sbatch script_job_1.sh
```

```
Job ID: 7053026
Hostname: bc6ulln6
Memory Usage:          total        used        free      shared  buff/cache   available
Mem:            31G       804M       28G       98M      1.8G        29G
Swap:           0B         0B         0B
Node: bc6ulln6
CPUs: 2
Threads: 32
Number of Nodes: 1
Number of Tasks: 1
Cores/CPUs per Node: 2
Runtime Duration: 0 minutes and 0 seconds.
```



Job Monitoring

sbatch script_job_2.sh

squeue -u username

```
(base) [vc18@login3 Workshop_Nov_2023]$ squeue -u vc18
      JOBID PARTITION      NAME      USER ST      TIME  NODES NODELIST(REASON)
 7054057  ctbp-onuc    job_2    vc18  R      0:06      1 bc6u1ln6
```

Job Monitoring

```
scancel job_ID
```

```
scancel 7054057
```

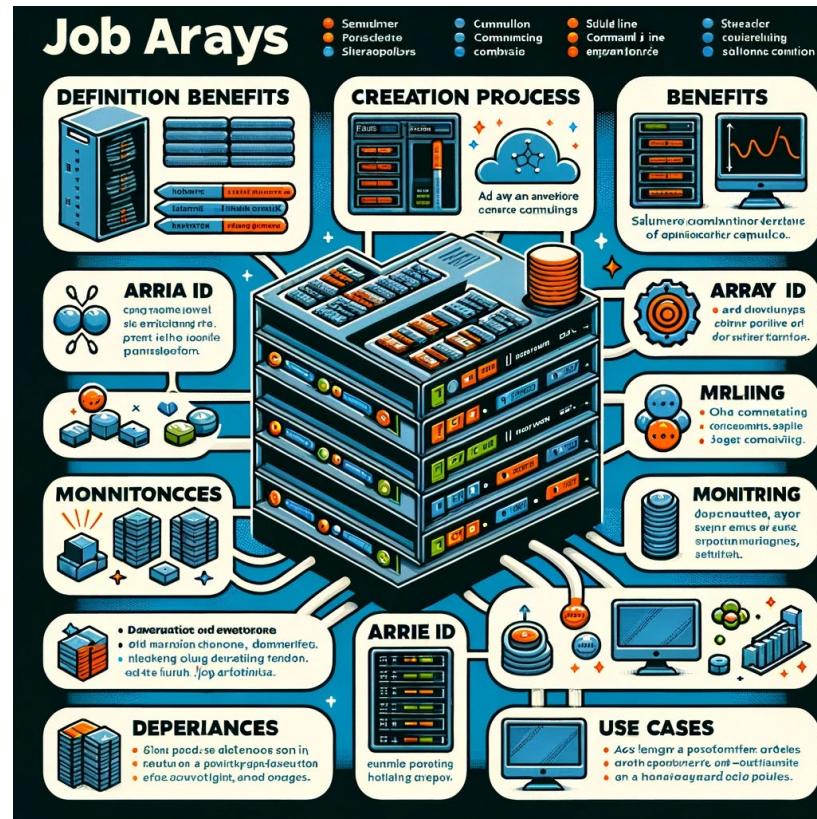
```
(base) [vc18@login3 Workshop_Nov_2023]$ cat slurm-7054057-job_2.err
slurmstepd: error: *** JOB 7054057 ON bc6u11n6 CANCELLED AT 2023-10-30T13:45:10 ***
```



Job Array

What is Job Array?

- A job array allows the submission of many similar jobs using a single script.
- Each job in the array is assigned a unique "array ID", which can be used within the script to vary the behavior of individual tasks.



Job Array

```
#!/bin/bash
#SBATCH --job-name=job_3_array
#SBATCH --account=ctbp-onuchic
#SBATCH --partition=ctbp-onuchic
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --time=00:10:00
#SBATCH --mem-per-cpu=100M
#SBATCH -e slurm-%A_%a-%x.err
#SBATCH -o slurm-%A_%a-%x.out
#SBATCH --array=1-8

echo "Array Task ID: $SLURM_ARRAY_TASK_ID"
echo "Job ID: $SLURM_JOB_ID"
echo "Hostname: $(hostname)"
echo "Memory Usage: $(free -h)"
echo "Node: $SLURM_NODELIST"
echo "CPUs: $SLURM_CPUS_ON_NODE"
echo "Threads: $(nproc --all)"
echo "Number of Nodes: $SLURM_NNODES"
echo "Number of Tasks: $SLURM_NTASKS"
echo "Cores/CPUs per Node: $SLURM_CPUS_ON_NODE"
sleep 240 # time in seconds
echo "Runtime Duration: $((($SECONDS / 60)) minutes and $($SECONDS % 60)) seconds."
```



Job Array

```
sbatch script_job_3_array.sh
```

```
Submitted batch job 7054274
```

```
(base) [vc18@login3 Workshop_Nov_2023]$ squeue -u vc18
      JOBID PARTITION      NAME      USER ST      TIME  NODES NODELIST(REASON)
 7054274_[1-8]  ctbp-onuc  job_3_ar    vc18 PD      0:00      1 (Priority)
```

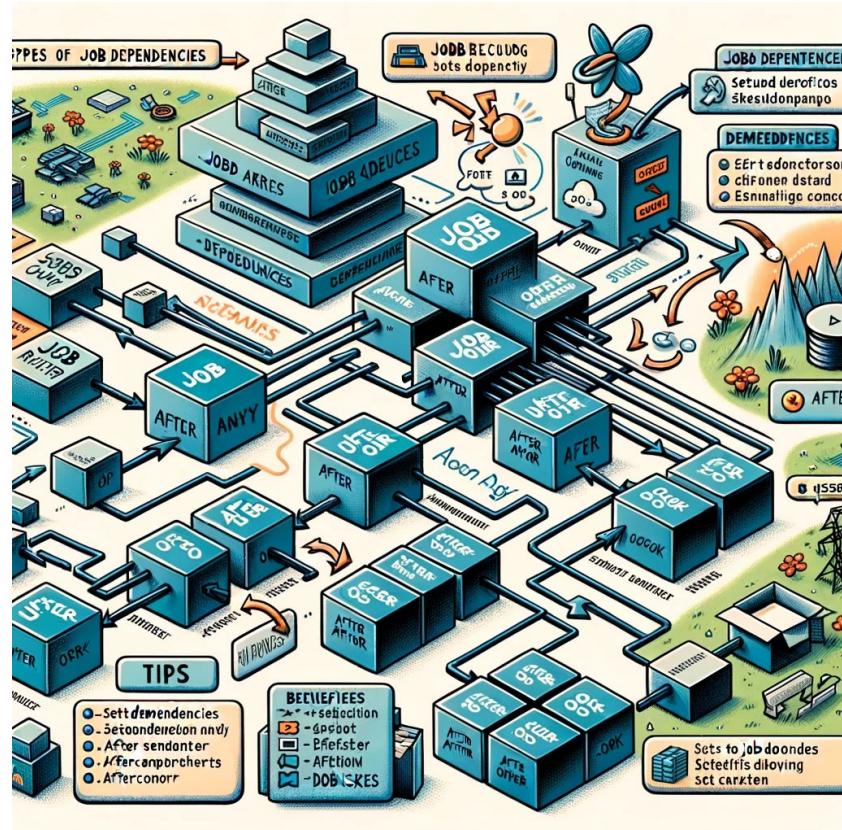
```
(base) [vc18@login3 Workshop_Nov_2023]$ squeue -u vc18
      JOBID PARTITION      NAME      USER ST      TIME  NODES NODELIST(REASON)
 7054274_1  ctbp-onuc  job_3_ar    vc18 R      0:36      1 bc6u15n1
 7054274_2  ctbp-onuc  job_3_ar    vc18 R      0:36      1 bc6u15n1
 7054274_3  ctbp-onuc  job_3_ar    vc18 R      0:36      1 bc6u15n1
 7054274_4  ctbp-onuc  job_3_ar    vc18 R      0:36      1 bc6u15n1
 7054274_5  ctbp-onuc  job_3_ar    vc18 R      0:36      1 bc6u15n1
 7054274_6  ctbp-onuc  job_3_ar    vc18 R      0:36      1 bc6u15n1
 7054274_7  ctbp-onuc  job_3_ar    vc18 R      0:36      1 bc6u15n1
 7054274_8  ctbp-onuc  job_3_ar    vc18 R      0:36      1 bc6u15n1
```



Job Dependency

What is Job Dependency?

- Allows you to define conditions under which a job will start based on the state of one or more other jobs.
- Useful for workflows, where tasks need to be executed in a specific order or condition.



Job Dependency

```
#!/bin/bash
#SBATCH --job-name=job_4_dependency
#SBATCH --account=ctbp-onuchic
#SBATCH --partition=ctbp-onuchic
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --time=00:10:00
#SBATCH --mem-per-cpu=100M
#SBATCH -e slurm-%A_%a-%x.err
#SBATCH -o slurm-%A_%a-%x.out
#SBATCH --array=1-8
```

```
# If it's the last job in the array, submit the dependent job
if [ $SLURM_ARRAY_TASK_ID -eq 1 ]; then
    sbatch --dependency=afterany:$SLURM_ARRAY_JOB_ID script_job_5.sh
fi
```

```
echo "Number of Nodes: $SLURM_NNODES"
echo "Number of Tasks: $SLURM_NTASKS"
echo "Cores/CPUs per Node: $SLURM_CPUS_ON_NODE"

sleep $TOTAL_SLEEP_TIME

echo "Runtime Duration: $((($SECONDS / 60)) minutes and $((($SECONDS % 60))) seconds.)"
```



Job Dependency

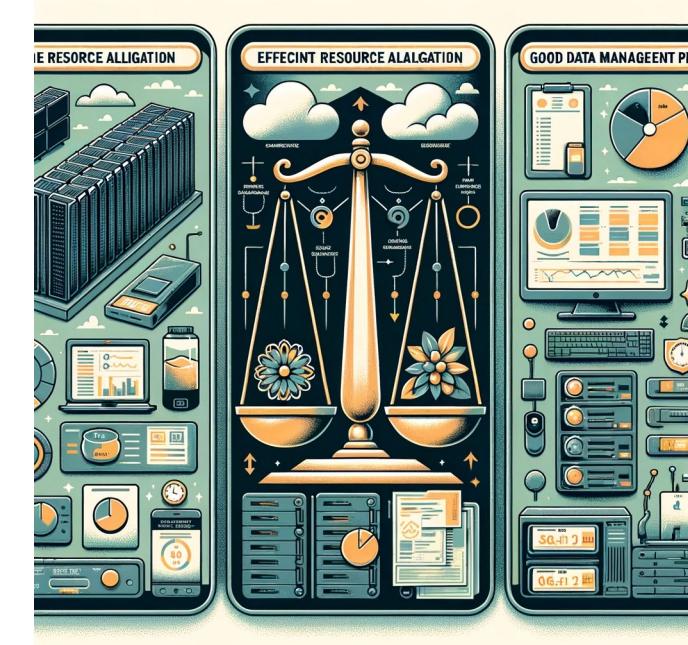
```
#!/bin/bash
#SBATCH --job-name=job_5_after_all_4
#SBATCH --account=ctbp-onuchic
#SBATCH --partition=ctbp-onuchic
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --time=00:10:00
#SBATCH --mem-per-cpu=100M
#SBATCH -e slurm-%j-%x.err
#SBATCH -o slurm-%j-%x.out

# Your job commands go here
echo "Running script_job_5.sh after the job array completed."
sleep 120
```



Best Practices for HPC Cluster Usage

- **Efficient Resource Allocation:** Overestimating can lead to wasteful usage of cluster resources, while underestimating can result in job failure or suboptimal performance.
- **Effective Job Monitoring and Management:** Keeping track of your jobs' status and performance is essential for catching errors early and ensuring efficient use of the cluster.
- **Practice Good Data Management:** Organize, back up, and manage your data efficiently to ensure smooth operations and avoid data loss during your computations.



BENCHMARKING

Intermediate SLURM Operations

Interactive Section and Node Allocation

```
run --pty --partition=ctbp-onuchic --account=ctbp-onuchic --ntasks=1 --mem=1G --time=00:30:00 $SHELL
```

GPU Request

```
#SBATCH --gres=gpu:1
```

OpenMP, MPI, and Hybrid Jobs

```
(base) [vc18@login3 advanced_slurm_parallel]$ ls
hybrid_mpi_openmp_example.py  mpi_example.py      result_hybridJob.txt  result_openmpJob.txt  slurm_mpi.sh
interactive_command.txt       openmp_example.py  result_mpiJob.txt   slurm_hybrid.sh    slurm_openmp.sh
```



CTBP – Resources & Useful links

- <https://kb.rice.edu/page.php?id=108237> (**NOTS**)
- <https://wiki.rice.edu/confluence/display/CTBP/CTBP+Computing+Activities> (**CTBP**)
- <https://researchcomputing.rice.edu> (**CRC**)
- <https://slurm.schedmd.com/tutorials.html> (**SLURM**)

