COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
Centre for Information Resource Management (CIRM)

TEJASWI HPC CLUSTER USER GUIDE

Access Protocols • Software Environments • Job Submission

Official Documentation

Cochin University of Science and Technology, Centre for Information Resource Management (CIRM) Version 2.0 | 2026

For technical support: aniesh.mathew@staroneit.com | +91 7403030307

Cluster Deployed and Maintained by StarOne IT Solutions India Pvt Ltd

Welcome to Tejaswi HPC Cluster

Welcome to the Tejaswi High Performance Computing (HPC) Cluster at Cochin University of Science and Technology (CUSAT). This guide provides everything you need to connect, transfer files, load software, and submit computational jobs effectively.

Please read this document carefully before beginning work on the cluster. Improper use of shared resources — particularly running heavy computations on login nodes — will result in immediate job termination.

1. System Architecture & Access Protocol

The Tejaswi HPC Cluster uses a secure multi-tier access model. Direct external access to compute nodes is disabled. All connections must flow through the designated jump host before reaching the login or compute nodes.

Access Flow

```
Your PC / Laptop ──► Jump Host(10.0.8.142) ──► Login Node(10.20.0.203)

(University Network / VPN)  (Tier 1)  (Tier 2)
```

1.1 Tier 1 — Jump Host Access

The jump host is the entry point to the cluster. Accessible from the university network or via VPN from personal devices.

Host Type	IP Address	Username	Password
Jump Host	10.0.8.142	user_name	Shared Separately

SSH Command:

```
ssh user_name @10.0.8.142
```

Access Issues: For jump host authorization, contact administrator@cusat.ac.in or call +91 9656225162 during business hours.

1.2 Tier 2 — Login Node Access

After connecting to the jump host, SSH into the login node to access the cluster environment and job scheduler.

Host Type	IP Address	Username	Password
Login Node	10.20.0.203	user_name	Shared Separately

SSH Command (from Jump Host):

```
ssh user_name @10.20.0.203
```

CRITICAL: Never run compute jobs directly on the login node. Any process consuming high CPU or memory on the login node will be terminated immediately without warning.

2. Data Transfer & Storage

All user data must be stored in the Parallel File System (PFS). Use the Secure Copy Protocol (scp) or SFTP to transfer files. Data transfer follows the same two-hop path as SSH access.

2.1 Step 1 — Transfer to Jump Host

From your local PC or laptop, copy files to the jump host:

```
# Copy a single file
scp local_file.txt user_name @10.0.8.142:/home/user_name

# Copy an entire directory (recursive)
scp -r my_project/ user_name @10.0.8.142:/home/user_name
```

2.2 Step 2 — Transfer to Login Node (PFS)

From the jump host, transfer files to the parallel file system(PFS) on the login node:

```
# Copy file to your PFS home directory
scp jump_host_file.txt user_name @10.20.0.203:/pfs/home/user_name

# Copy directory to PFS
scp -r my_project/ user_name @10.20.0.203:/pfs/home/user_name
```

> **Tip:** Store all working data under /pfs/home/user_name . Only PFS directories persist across sessions. Data stored elsewhere (e.g. /tmp) may be deleted at any time.

3. Application Environment & Modules

Software on Tejaswi is managed through the Environment Modules system. Use module commands to load, unload, and list available applications. All application binaries are installed under /pfs/apps/.

3.1 Useful Module Commands

Command	Description
`module avail`	List all available modules on the cluster

`module load <name>`	Activate a software module (sets PATH, libs, etc.)
`module list`	Show currently loaded modules
`module unload <name>`	Deactivate a specific module
`module purge`	Unload all currently loaded modules
`module show <name>`	Show what a module sets up (paths, variables)

3.2 Installed Software

The following research applications are installed on Tejaswi. The module path base is /pfs/apps/modulefiles.

Application	Version	Module Name	Domain
Weather Research & Forecasting (WRF)	`4.7.2`	`WRF-4.7.1`	Atmospheric Science
Regional Climate Model (RegCM)	`5.0.0`	`RegCM-5.0.0`	Climate Modeling
GROMACS	`2022.6`	`gromacs-2022.6`	Molecular Dynamics
LAMMPS	`22-Jul-25`	`lammps-2025`	Molecular Dynamics
Quantum ESPRESSO	`7.5`	`q-e-qe-7.5`	Materials Science

3.3 Supporting Libraries

The following supporting libraries are also available as modules:

Library	Version	Purpose
HDF5	`1.14.6`	Hierarchical Data Format for large scientific datasets
NetCDF	`4.3.1`	Network Common Data Form — climate and weather data
zlib	`1.3.1`	Data compression library
pwtk	`3.2`	PWscf Toolkit — scripting for Quantum ESPRESSO
WPS	`4.6.0`	WRF Preprocessing System

4. Job Submission with PBS

Tejaswi uses the Portable Batch System (PBS) scheduler. All compute jobs must be submitted through PBS — never run computations directly on the login node. Jobs are queued and dispatched to compute nodes based on available resources.

4.1 Node Specifications

Understanding node hardware is essential for writing correct PBS resource requests. The table below summarises each node type available on Tejaswi.

Node Type	CPUs × Cores	Total Cores	RAM	Accelerators

CPU Compute Node	2 × 32	64 cores	256 GB	—
GPU Compute Node	2 × 24	48 cores	256 GB	2 × NVIDIA A100 80GB

PBS Tip: Set ncpus to the total core count of the node (e.g. 64 for CPU nodes, 48 for GPU nodes). Set mem to match the node RAM. Set mpiprocs to match ngpus for GPU jobs so each MPI rank maps to exactly one GPU.

4.2 Common PBS Commands

Command	Description
`qsub myjob.sh`	Submit a job script to the scheduler
`qstat`	List all jobs in the queue
`qstat -u user_name`	List only your jobs
`qdel <job_id>`	Cancel (delete) a queued or running job
`qstat -f <job_id>`	Show detailed info for a specific job

4.3 CPU Job Script — Multi-node MPI

The following template submits a parallel MPI job across multiple CPU nodes using Intel MPI. Each CPU node has 2 CPUs × 32 cores = 64 cores and 256 GB RAM. Adjust select (node count) to scale your job.

```
vi job1.pbs
#!/bin/bash
#PBS -N myjob
#PBS -l select=1:ncpus=64:mpiprocs=64:mem=256gb
#PBS -l walltime=04:00:00
#PBS -o myjob.out
#PBS -e myjob.err

cd "$PBS_O_WORKDIR"
ulimit -s unlimited

# ── MPI Environment ──────────────────────────────────────
source /pfs/softwares/intel/oneapi/setvars.sh
export PATH=/pfs/softwares/intel/oneapi/mpi/2021.17/bin:$PATH

which mpirun || { echo 'ERROR: mpirun not found'; exit 1; }
mpirun --version

# ── Application Module ──────────────────────────────────
```

```
module load <your-application-module>

# ── PBS Bootstrap for Intel MPI ──────────────────────────
export I_MPI_HYDRA_BOOTSTRAP=pbs
export I_MPI_HYDRA_HOST_FILE="$PBS_NODEFILE"

# ── Diagnostics ──────────────────────────────────────────
echo '=== Nodes assigned ===' && cat $PBS_NODEFILE | sort -u
mpirun -np 1 -ppn 1 hostname

# ── Main Run (1 node × 64 ranks = 64 total) ──────────────
echo "=== Starting at $(date) ==="
mpirun -np 64 /path/to/your/application [input_args]
echo "=== Done at $(date) ==="
```

To submit the job

```
qsub job1.pbs
```

CPU Resource Parameters (per node)

Parameter	Value	PBS Option	Notes
Nodes	1	select=1	Increase for multi-node jobs
CPUs × Cores	2 × 32	ncpus=64	Total logical cores per node
MPI ranks / node	64	mpiprocs=64	1 MPI rank per core
Memory / node	256 GB	mem=256gb	Full node memory

Total MPI ranks	64	1 × 64	Pass to mpirun -np 64

4.4 GPU Job Script — CUDA-Accelerated

GPU nodes host 2 × NVIDIA A100 80GB GPUs with 2 CPUs × 24 cores = 48 cores. Set mpiprocs=2 to match ngpus=2, assigning one MPI rank per GPU.

```
vi job1.pbs
#!/bin/bash
#PBS -N myjob_gpu
#PBS -l select=1:ncpus=48:mpiprocs=2:ngpus=2:mem=256gb
#PBS -l walltime=02:00:00
#PBS -q gpu_queue
#PBS -o myjob_gpu.out
#PBS -e myjob_gpu.err

cd "$PBS_O_WORKDIR"
ulimit -s unlimited

# ── MPI Environment ──────────────────────────────────────
source /pfs/softwares/intel/oneapi/setvars.sh
export PATH=/pfs/softwares/intel/oneapi/mpi/2021.17/bin:$PATH

# ── CUDA Environment ─────────────────────────────────────
```

```
module load cuda/12.3
export CUDA_VISIBLE_DEVICES=0,1 # 2 x A100 80GB

# ── Application Module (GPU-enabled build) ──────────────
module load <your-gpu-application-module>

# ── PBS Bootstrap for Intel MPI ─────────────────────────
export I_MPI_HYDRA_BOOTSTRAP=pbs
export I_MPI_HYDRA_HOST_FILE="$PBS_NODEFILE"

# ── GPU Diagnostics ─────────────────────────────────────
echo '=== GPU devices on this node ==='
nvidia-smi --query-gpu=name,memory.total --format=csv,noheader

# ── Node Diagnostics ────────────────────────────────────
echo '=== Nodes assigned ===' && cat $PBS_NODEFILE | sort -u

# ── Main Run (1 node × 2 ranks = 2 total; 1 rank/GPU) ──
echo "=== Starting at $(date) ==="
mpirun -np 2 /path/to/your/gpu-application [input_args]
echo "=== Done at $(date) ==="
```

To submit the job

```
qsub job1.pbs
```

GPU Resource Parameters (per node)

Parameter	Value	PBS Option	Notes
Nodes	1	select=1	Increase for multi-node GPU jobs

CPUs × Cores	2 × 24	ncpus=48	Total logical cores per GPU node
MPI ranks / node	2	mpiprocs=2	Matches number of GPUs per node
GPUs / node	2	ngpus=2	2 × NVIDIA A100 80GB per node
Memory / node	256 GB	mem=256gb	Full node memory
Total MPI ranks	2	1 × 2	Pass to mpirun -np 2
Total GPU memory	160 GB	2 × 80 GB	Each A100 has 80 GB HBM2e

4.5 GPU Job Script — Single GPU

Use this script when your workload only needs one A100. Requesting a single GPU allocates half the node's cores (24) and half the memory (128 GB), leaving the other GPU free for another user's job.

```
vi job1.pbs
```

```
#!/bin/bash
#PBS -N myjob_gpu_single
#PBS -l select=1:ncpus=24:mpiprocs=1:ngpus=1:mem=128gb
#PBS -l walltime=02:00:00
#PBS -q gpu_queue
#PBS -o myjob_gpu_single.out
```

```
#PBS -e myjob_gpu_single.err

cd "$PBS_O_WORKDIR"
ulimit -s unlimited

# ── MPI Environment ──────────────────────────────────
source /pfs/softwares/intel/oneapi/setvars.sh
export PATH=/pfs/softwares/intel/oneapi/mpi/2021.17/bin:$PATH

# ── CUDA Environment ─────────────────────────────────
module load cuda/12.3
export CUDA_VISIBLE_DEVICES=0 # Single A100 80GB

# ── Application Module (GPU-enabled build) ───────────
module load <your-gpu-application-module>

# ── PBS Bootstrap for Intel MPI ──────────────────────
export I_MPI_HYDRA_BOOTSTRAP=pbs
export I_MPI_HYDRA_HOST_FILE="$PBS_NODEFILE"

# ── GPU Diagnostics ──────────────────────────────────
echo '=== GPU device on this node ==='
nvidia-smi --query-gpu=name,memory.total --format=csv,noheader

# ── Node Diagnostics ─────────────────────────────────
echo '=== Nodes assigned ===' && cat $PBS_NODEFILE | sort -u

# ── Main Run (1 node × 1 rank = 1 total; single A100) ──
echo "=== Starting at $(date) ==="
mpirun -np 1 /path/to/your/gpu-application [input args]
echo "=== Done at $(date) ==="
```

To submit the job

```
qsub job1.pbs
```


Single-GPU Resource Parameters

Parameter	Value	PBS Option	Notes
Nodes	1	`select=1`	Single GPU node
CPUs allocated	24	`ncpus=24`	Half the node cores (2 × 24 ÷ 2)
MPI ranks / node	1	`mpiprocs=1`	One rank for the single GPU
GPUs / node	1	`ngpus=1`	1 × NVIDIA A100 80GB
Memory	128 GB	`mem=128gb`	Half node RAM, proportional to 1 GPU
CUDA_VISIBLE_DEVICES	0	—	Exposes only the assigned A100
Total MPI ranks	1	1 × 1	Pass to mpirun -np 1

> **Tip:** Prefer the single-GPU script for development, testing, or workloads that do not scale across multiple GPUs. This keeps the second A100 available for other users and reduces your queue wait time.

4.6 CPU vs GPU — Key Differences

Aspect	CPU Job	GPU Job (Both GPUs)	GPU Job (Single GPU)
Resource Request	`select=1:ncpus=64 mpiprocs=64:mem= 256gb`	`select=1:ncpus=48 mpiprocs=2:ngpus=2: mem =256gb`	`select=1:ncpus=24 mpiprocs=1:ngpus=1: mem =128gb`
Queue	`(default)`	`gpu_queue`	`gpu_queue`
CUDA Module	`Not required`	`module load cuda/12.3`	`module load cuda/12.3`
CUDA_VISIBLE_DEVICES	`Not required`	`0,1`	`0`
GPUs Used	None	2 × A100 80GB	1 × A100 80GB
GPU Memory	—	160 GB total	80 GB
Cores Allocated	64	48	24
RAM Allocated	256 GB	256 GB	128 GB
Total MPI Ranks	64	2	1
Best For	CPU-parallel workloads	Large GPU jobs / production runs	Dev, testing, single GPU apps

5. PBS Directives Reference

PBS directives are special comments (beginning with #PBS) that the scheduler reads before executing the script. They must appear before any shell commands.

Directive	Purpose & Example
`#PBS -N jobname`	Sets the job name shown in qstat. Use descriptive names.
`#PBS -l select=N:ncpus=M:mpiprocs=M`	Resource allocation: N nodes, M cores and MPI ranks each.
`#PBS -l select=...:ngpus=G`	Add GPU reservation: G GPU devices per node.
`#PBS -l walltime=HH:MM:SS`	Maximum runtime. Job is killed if this is exceeded.

`#PBS -q queue_name`	Target queue (e.g., gpu_queue for GPU nodes).
`#PBS -o filename.out`	File to capture standard output (stdout).
`#PBS -e filename.err`	File to capture standard error (stderr).

> **Note:** All #PBS directives must appear before any shell commands. The scheduler stops reading directives at the first non-comment, non-directive line.

6. Cluster Usage Policy

- **Do not run compute jobs on the login node.** Any unauthorized process consuming significant CPU or memory will be terminated immediately without notice.
- **All compute jobs must be submitted via PBS** using qsub, with a valid walltime directive. • **Store data only in your allocated PFS directory** (/pfs/home/user_name). Do not use shared system directories for persistent storage.
- **Never load both a CPU-only and a GPU application module** in the same job script — mixed builds can produce silent errors or incorrect results.
- **Always include ulimit -s unlimited** in your job scripts to prevent segmentation faults in applications with deep call stacks.
- **Set a realistic walltime.** Too short kills your job; excessively long walltimes may delay scheduling.
- **When using GPU nodes, set mpiprocs equal to ngpus per node** so each MPI rank maps to exactly one GPU device.

7. Support & Contact Information

For technical assistance with the Tejaswi HPC Cluster, please use the contacts below. Support is available during regular business hours.

Technical Support

Contact Person: Aniesh Mathew
Email: aniesh.mathew@staroneit.com
Phone: +91 7403030307

Jump Host / Access Administration
Email: administrator@cusat.ac.in
Phone: +91 9656225162

Quick Reference Cheatsheet

CONNECTION	
Jump Host	`ssh user_name @10.0.8.142`
Login Node	`ssh user_name @10.20.0.203`
FILE TRANSFER	
To Jump Host	`scp file.txt user_name @10.0.8.142:/home/user_name`
To Login Node	`scp file.txt user_name @10.20.0.203:/pfs/home/user_name`
PBS SCHEDULER	
Submit Job	`qsub myjob.sh`
Check Job Status	`qstat -u user_name`
Cancel Job	`qdel <job_id>`
List Modules	`module avail`
Load Module	`module load <name>`

Tejaswi HPC Cluster • Cochin University of Science and Technology • CIRM