

# Comprehensive Guide to HPC and SLURM

This guide provides essential commands and workflows for working with High-Performance Computing (HPC) clusters and the SLURM scheduler.

## Table of Contents

- [Connecting to HPC Resources](#)
- [SLURM Basics](#)
- [Working with Modules](#)
- [File Transfer](#)
- [Environment Management](#)
- [SLURM Job Submission](#)
- [Job Monitoring and Management](#)
- [Interactive Sessions](#)
- [Common Workflows](#)
- [Neural Network Training \(NEquIP Examples\)](#)
- [Resource Management](#)
- [Additional Resources](#)

## Connecting to HPC Resources

Connect to HPC systems using SSH:

```
bash
```

```
ssh username@hpc.example.edu
```

For systems with non-standard SSH ports:

```
bash
```

```
ssh username@hpc.example.edu -p port_number
```

## SLURM Basics

SLURM (Simple Linux Utility for Resource Management) is a workload manager commonly used on HPC clusters.

Key commands:

- `sinfo` - Shows cluster status and available resources
- `squeue` - Displays job queue information
- `sbatch` - Submits a batch job script
- `srun` - Runs a command on allocated resources
- `scancel` - Cancels a submitted job

## Working with Modules

Many HPC systems use environment modules to manage software:

```
bash

# List available modules
ml
# or
module list

# Load a specific module
module load module_name

# Unload a module
module unload module_name
```

## File Transfer

Transfer files between local and remote systems:

```
bash

# From local to remote
scp /path/to/local/file username@hpc.example.edu:/path/to/remote/directory

# From remote to local
scp username@hpc.example.edu:/path/to/remote/file /path/to/local/destination

# Multiple files
scp file1.txt file2.yaml username@hpc.example.edu:/remote/directory/
```

## Environment Management

## Python Virtual Environments

```
bash

# Create a virtual environment
python -m venv my_project

# Activate a virtual environment
source /path/to/my_project/bin/activate

# Create with Conda
conda create --name env_name python=3.10

# Activate Conda environment
conda activate env_name

# List Conda environments
conda env list

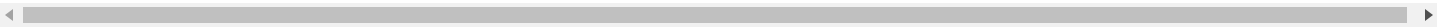
# Remove a Conda environment
conda env remove --name env_name
```

## Setting up Jupyter kernels

```
bash

# Install kernel packages
conda install -c conda-forge notebook ipykernel

# Register environment as a Jupyter kernel
python -m ipykernel install --user --name env_name --display-name "Python (env_name)"
```



## Environment Variables

```
bash

# Set environment variable
export VARIABLE_NAME="value"

# Unset environment variable
unset VARIABLE_NAME
```

## SLURM Job Submission

## Basic SLURM Script

bash

```
#!/bin/bash
#SBATCH -o job_output_%j.out      # Standard output file (%j is job ID)
#SBATCH -e job_error_%j.err      # Standard error file
#SBATCH -p partition_name        # Partition/queue name
#SBATCH -N 1                      # Number of nodes
#SBATCH --ntasks-per-node 4      # Number of tasks per node
#SBATCH -t 1-00:00:00            # Time limit (days-hh:mm:ss)
#SBATCH --mail-type=ALL          # Mail events (BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=your.email@example.com # Email address for notifications

# Load necessary modules
module load required_module

# Activate virtual environment if needed
source /path/to/venv/bin/activate

# Run commands
echo "Job started"
your_command --with --parameters
echo "Job ended"
```

## Submitting a Job

bash

```
sbatch job_script.sh
```

## Job Monitoring and Management

```
bash
```

```
# View all your running jobs
```

```
squeue -u $USER
```

```
# View all jobs in the system
```

```
squeue -a
```

```
# Get detailed information about a specific job
```

```
scontrol show job job_id
```

```
# View job history and statistics
```

```
sacct -u $USER --format=JobID,JobName%30,Partition,State,Elapsed,ExitCode -X
```

```
# Cancel a job
```

```
scancel job_id
```

## Interactive Sessions

Start an interactive session on compute nodes:

```
bash
```

```
# Basic interactive session
```

```
srun --pty /bin/bash
```

```
# Detailed request
```

```
srun -N 1 -n 1 --time=02:00:00 -p development --pty /bin/bash
```

```
# Interactive session with GPUs
```

```
idev -p gpu-a100-small -N 1 -n 1 -t 1:00:00
```

## Common Workflows

### Environment Setup and File Management

Working directory workflow:

```
bash
```

```
# Create and enter project directory
```

```
mkdir -p /work/username/my_project
```

```
cd /work/username/my_project
```

```
# Copy files from local to HPC
```

```
scp /local/path/file.txt username@hpc.example.edu:/work/username/my_project/
```

```
# Clean up results directory before new run
```

```
rm -rf ./results
```

## Neural Network Training (NEquIP Examples)

These examples use NEquIP, a neural equivariant interatomic potential framework:

```
bash
```

```
# Activate the environment
```

```
source /path/to/nequip_env/bin/activate
```

```
# Test model configuration
```

```
python -c "import yaml; yaml.safe_load(open('config.yaml'))"
```

```
# Run equivariance test
```

```
nequip-train /path/to/config.yaml --equivariance-test
```

```
# Train model
```

```
nequip-train /path/to/config.yaml
```

```
# Benchmark performance
```

```
nequip-benchmark /path/to/config.yaml
```

```
# Evaluate model
```

```
nequip-evaluate --train-dir /path/to/results/directory --batch-size 1
```

```
# Deploy model
```

```
nequip-deploy build --train-dir /path/to/results/directory output_model.pth
```

```
# Evaluate deployed model
```

```
nequip-evaluate --model model.pth --dataset-config dataset_config.yaml
```

```
nequip-evaluate --model model.pth --dataset-config config.yaml --metrics-config metric_
```

## Working with ASE and atomic data

python

```
from ase.io import read, write

# Read all configurations
atoms_list = read('data.extxyz', index=':')

# Modify atom properties
for atoms in atoms_list:
    atoms.info['property_name'] = value

# Write back to file
write('modified_data.extxyz', atoms_list)
```

## Resource Management

Check user and group permissions:

bash

```
# Check which groups you belong to
groups

# See members of a specific group
getent group group_name

# Check permissions on a directory
ls -ld /path/to/directory
```

## Additional Resources

### Learning Materials

- [Linux Journey](#) - Learn Linux fundamentals
- YouTube Tutorials:
  - "Three ways to use SLURM on a high-performance computer (HPC)"
  - "HPC NYU tutorials"
  - "HPC 1-hour guide"
  - "Neural networks on HPC systems"

## PyTorch Installation with CUDA

```
bash
```

```
# Install PyTorch Scatter with CUDA
```

```
pip install torch-scatter -f https://data.pyg.org/whl/torch-2.5.1+cu124.html
```

```
pip install torch-scatter -f https://data.pyg.org/whl/torch-2.6.0+cu124.html
```

---

**Note:** Always refer to your specific HPC system's documentation for system-specific commands and policies. Commands might vary slightly between different HPC environments.

Author: Chirag Sindhwani