

HIGH-PERFORMANCE COMPUTING

Assignment-01

2303A51463

Batch-07

Section 1: Section A — SSH Access & Environment Sanity Check

(Login Node)

Aim

Access the HPC cluster via SSH, verify environment (Python version, CPU info), and run a trivial Python script to confirm your account setup. (No heavy compute—just a sanity check.)

Requirements

- An HPC account (username, password/SSH key)
- Hostname of the cluster login node (e.g., login.cluster.edu)
- SSH client (Linux/macOS Terminal or PuTTY on Windows)
- Python 3.x available on the cluster

[1]

```
#hello_cluster.py
import os
import platform
import sys
print("Hello from HPC cluster!")
print("User:", os.getenv("USER"))
print("Host:", platform.node())
print("Python:", sys.version)
print("cores(logical):", os.cpu_count())
```

```
... Hello from HPC cluster!
User: None
Host: 2ce418d41e92
Python: 3.12.12 (main, Oct 10 2025, 08:52:57) [GCC 11.4.0]
cores(logical): 2
```

Section 2: Section B — First Batch Job: Serial Python Script via Slurm/PBS


Aim

Submit a simple serial Python job to the scheduler, capture logs, check job status, and understand job metadata.

Requirements

- Slurm (sbatch) or PBS/Torque (qsub)
- Python 3.x available on compute nodes
- A short partition/queue (e.g., short) and project/account if

Required

```
[7]
✓ 0s  #serial_baseline.py
import time,os,platform,sys
def work(n=5_00_000):
    s=0.0
    for i in range(n):
        s+=(i%7)*0.123456
    return s
if __name__=='__main__':
    print("===Job Info===")
    print("Host",platform.node())
    print("User",os.getenv("USER"))
    print("Python",sys.version)
    print("===work===")
    t0=time.perf_counter()
    result=work()
    t1=time.perf_counter()
    print("result={result:.6f}|Time={(t1-t0):.3f}s")

...
===Job Info===
Host 2ce418d41e92
User None
Python 3.12.12 (main, Oct 10 2025, 08:52:57) [GCC 11.4.0]
===work===
result={result:.6f}|Time={(t1-t0):.3f}s
```

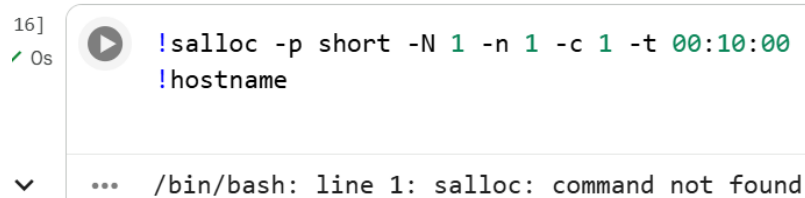
Section 3: Section C — Interactive Compute Session (salloc/srun or qsub -l)

Aim

Launch an interactive allocation on a compute node to run Python commands interactively (e.g., for debugging or quick experiments).

Requirements

- Slurm (salloc/srun) or PBS (qsub -l)
- Python 3.x available on compute nodes



A terminal window showing a failed command. The prompt is 16] and the command is !salloc -p short -N 1 -n 1 -c 1 -t 00:10:00. The output is !hostname. Below the command, there is a dropdown arrow and the error message ... /bin/bash: line 1: salloc: command not found.

```
16] !salloc -p short -N 1 -n 1 -c 1 -t 00:10:00
    !hostname
... /bin/bash: line 1: salloc: command not found
```

Section 4: Section D — Job Arrays for Parameter Sweep

(Multiple Serial Runs)

Aim

Use job arrays to submit multiple independent serial Python jobs with different parameters (e.g., Monte Carlo sample sizes), improving throughput and simplifying management.

Requirements

- Slurm (array jobs) or PBS (array jobs)
- Python 3.x
- Basic knowledge of environment variables

(SLURM_ARRAY_TASK_ID / PBS_ARRAYID)

[19]
✓ 0s

```
import os,time,random

def estimate_pi(n_samples,seed=42):
    random.seed(seed)
    inside=0
    for _ in range(n_samples):
        x=random.random()
        y=random.random()
        inside+=(x*x+y*y<=1.0)
    return 4.0*inside/n_samples

if __name__=="__main__":
    tid=int(os.getenv("SLURM_ARRAY_TASK_ID",os.getenv("PBS_ARRAYID","0")))
    sizes=[50_000,100_000,200_000,300_000,400_000,800_000]
    n=sizes[tid%len(sizes)]
    t0=time.perf_counter()
    pi=estimate_pi(n,seed=100+tid)
    t1=time.perf_counter()
    print(f"TaskID={tid} N={n} pi\u2248{pi:.6f} Time={{(t1-t0):.3f}s}")
```

... TaskID=0 N=50000 pi≈3.141360 Time=0.010s

Section 5: Section E — Modules/Venv, Scratch I/O, and Resource

Flags

Aim

Submit a serial Python job that uses modules or virtual environments, writes outputs to scratch, and uses explicit resource flags (CPU/time/partition/queue).

Requirements

- Environment Modules (module command) or Python venv
- Scratch directory path (e.g., \$SLURM_TMPDIR or /scratch/\$USER)
- Slurm or PBS
- Python 3.x

