SLURM

General Guidelines for SLURM Usage

Table of Contents

1. General Guidelines for SLURM Usage

SLURM (Simple Linux Utility for Resource Management) is a just a job scheduler for the cluster. Below are some essential guidelines to ensure smooth job execution.

Key Points

- **Use the (base) Conda environment while submitting jobs**
  → Do **not** activate **other** Conda environments(job specific) **before** submitting the job. Othwise make sure, you've changed our **our deafult** slurm script, as you already activated that.
  → **Activate the required Conda environment inside the SLURM script.**

→ **Submit Job from the working directory for your code and slurm script is in the root folder for your project(if you use relative path). If you use full path instead of relative path, this is not necessary.**

- **Use Singularity instead of Docker**
  → Docker is unavailable in the SLURM job environment. It is only for predownload the images and store it locally for the computation. It is recommended that to use singularity for pull the images to store locally.
  → Singularity should be used for containerized workflows as well as job submission.
- **Avoid internet access during SLURM execution**
  → As like other HPC clusters, there is a restriction for internet access for SLURM jobs.
  → **Pre-download all required software, data, and Singularity images before submitting a job.**
- Check some documentations:
  Singularity: https://nfdi4ing.pages.rwth-aachen.de/knowledge-base/how-tos/all_articles/how_to_run_a_job_on_the_cluster_using_slurm_and_singularity/
  Nextflow: https://training.nextflow.io/basic_training/
  Conda: https://docs.conda.io/en/latest/\

2. Basic SLURM Commands

Submit a Job

```
sbatch job_script.sh
```

Check Job Status

```
squeue -u $USER
```

Cancel a Job

```
scancel <JOB_ID>
```

Check Job Output

```
cat slurm-<JOB_ID>.out
```

3. Writing an Effective SLURM Job Script

A **SLURM script** is a batch script containing directives (`#SBATCH`) for job submission.

Key Considerations

- Define **partition**, **memory**, **CPUs**, and **GPU** requirements.
- **Load required modules** (e.g., `singularity` , `nextflow` , `anaconda`).

- **Activate the correct Conda environment inside the script**.
- **Ensure software dependencies** (e.g., Singularity images) are accessible locally.
- Docs:
  https://docs.scibiome.nat.tu-bs.de/en/Slurm/compute
  https://docs.scibiome.nat.tu-bs.de/en/Slurm/GPU-Server

Example SLURM Job Script (`job_script.sh`)- GPU Based

```
#!/bin/bash
#SBATCH --partition=gpu                # Specify the GPU partition
#SBATCH --nodes=1                      # Request 1 node
#SBATCH --ntasks=1                     # Number of tasks (1 task)
#SBATCH --cpus-per-task=20             # Number of CPU cores per task
#SBATCH --gres=gpu:2                   # Request 2 GPUs
#SBATCH --mem=64GB                    # Amount of RAM
#SBATCH --time=24:00:00                 # Set a time limit
#SBATCH --job-name=basecalling_gpu      # Job name
#SBATCH --output=job_output.log         # Output log file
#SBATCH --mail-user=youremail@example.com # your email address

# **Step 1: Load Necessary Modules**
module load anaconda/3                 # Load Anaconda
source $(conda info --base)/etc/profile.d/conda.sh
conda activate base                     # Activate the base Conda environment

python test.py

# **Step 5: Print End Time**
echo "Job finished at: $(date)"
```

Example for a **CPU-only** computation **slurm** script is available here: https://docs.scibiome.nat.tu-bs.de/en/Slurm/compute

---

4. Example for Running Nextflow Workflows with SLURM

Key Considerations

- **Ensure Nextflow is loaded via a module.**
- **Use Singularity containers instead of Docker, as docker needs to be installed locally not cluster-wide. .**
- **Bind necessary directories for the containerized workflow.**
- **Your custom nextflow configuration.**
  Sample docs:
  https://carpentries-incubator.github.io/workflows-nextflow/08-configuration.html
  https://www.nextflow.io/docs/latest/container.html
  https://docs.sylabs.io/guides/3.2/user-guide/cli/singularity_pull.html
- Pull your Images first

```
From Sylabs cloud library
$ singularity pull alpine.sif library://alpine:latest

From Docker
$ singularity pull tensorflow.sif docker://tensorflow/tensorflow:latest

From Shub
$ singularity pull singularity-images.sif shub://vsoch/singularity-images
```

- Our Sample example for nextflow.conf\

```
singularity {
   enabled = true
   autoMounts = true
   cacheDir = "/home/users/test/folder"
}

process {
   executor = 'local'
}

docker.enabled = false
singularity.enabled = true


process {
    withName: 'getParams' {
        container = "/path/to/container/name.sif"
    }
    withName: 'getParams2' {
        container = "/path/to/container/name.sif"
    }
  }
```

\

Example Nextflow-Specific SLURM Job Script

```
#!/bin/bash
#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=20
#SBATCH --gres=gpu:1
#SBATCH --mem=32GB
#SBATCH --time=24:00:00
#SBATCH --job-name=nextflow_job
#SBATCH --output=nextflow_output.log

# **Load required modules**
module load anaconda/3
source $(conda info --base)/etc/profile.d/conda.sh
conda activate base

module load singularity/4.2.1
module load nextflow/24.04

# **Set up Singularity to use local images**
export NXF_SINGULARITY_CACHEDIR="/home/users/<u_name>/folder/"
export NXF_SINGULARITY_PREPULL=false
export NXF_SINGULARITY_RUNOPTIONS="--bind /home/users/<u_name>/folder/"

# **Run Nextflow pipeline**
nextflow run <job> -profile singularity -c nextflow.config
```

---

5. Monitoring and Debugging SLURM Jobs

Check Job Status

```
squeue
```

Cancel a Running Job

```
scancel <JOB_ID>
```

Check Job Logs

```
cat job_output.log
```

\

7. Best Practices for Running Jobs on SLURM

- **Use Conda's `(base)` environment before submitting a job.**

- **Activate the required Conda environment inside the SLURM script, not before submission.**
- **Use Singularity instead of Docker for containerized workflows.**
- **Ensure all required software and images are pre-downloaded.**
- **Monitor job status and logs frequently to detect errors early.**

8. Conda specific

- Always check version dependencies, either it's for python or some specific package.
- Use **pypi** libraries when conda's library for the specific Python packges are not available
- Make sure you install all the libraries after you activate your specific environment.

9. Some General Advice for SLURM Usage

- If computation is **GPU heavy**, use low number of cpu cores, e.g: `cores: 12, ram-32 GB.`
- While using **multi gpu**, make sure your **code support** this. Currently we have: `dds8-2* A40, and dds9: 4* A40` gpus. Use **multi gpu iff** your model **parameters** actually need it. Check the doc below: https://www.oreilly.com/library/view/generative-ai-on/9781098159214/ch04.html
- No constraints for cpu specific tasks.
  \

10. Accounting Storage for slurm

It enables a **centralized database (via SlurmDBD)** to persist **all job-related metadata-** this is what's referred to as **"accounting storage"**.\

For every job submitted from now on, SlurmDBD logs:

Logged Data	Example

Basic Job Status with sacct:\

```
sacct -j <jobID>
```

Typical output:

```
JobID    JobName  Partition    User  State      ExitCode
------------ ---------- ---------- ------- ---------- ---------
2759         test-slurm computedfg dborman COMPLETED   0:0
2759.batch              computedfg dborman COMPLETED   0:0
```

If it's blank, the job may still be running or accounting delay (~30 sec). Use `squeue` to check live status:

```
squeue -j 2759
```

Live Job Details:

```
scontrol show job 2759
```

Job Details from SAACT

```
sacct -j 2759 --format=JobID,JobName,User,State,Elapsed,Submit,Start,End,Allo
```

2759 is just a Sample <JOB_ID>

```
(base) dborman@headnode:~/test$ sacct -j 2759 --format=JobID,JobName,User,Par
JobID           JobName      User  Partition      State ExitCode    Elapsed
------------ ---------- --------- ---------- ---------- -------- ---------- -
2759         test-slurm   dborman computedfg  COMPLETED      0:0   00:00:15 2
2759.batch        batch                       COMPLETED      0:0   00:00:15 2
```

For GPU:

```
(base) dborman@headnode:~$ sacct -j 2762 --format=JobID,JobName,AllocTRES%50
JobID           JobName                                          AllocTRES
------------ ---------- --------------------------------------------------
2762         test-slur+          billing=6,cpu=6,gres/gpu=1,mem=10G,node=1
2762.batch        batch                     cpu=6,gres/gpu=1,mem=10G,node=1
```

More info: Search online, https://docs.rc.fas.harvard.edu/kb/convenient-slurm-commands/

https://curc.readthedocs.io/en/latest/running-jobs/slurm-commands.html