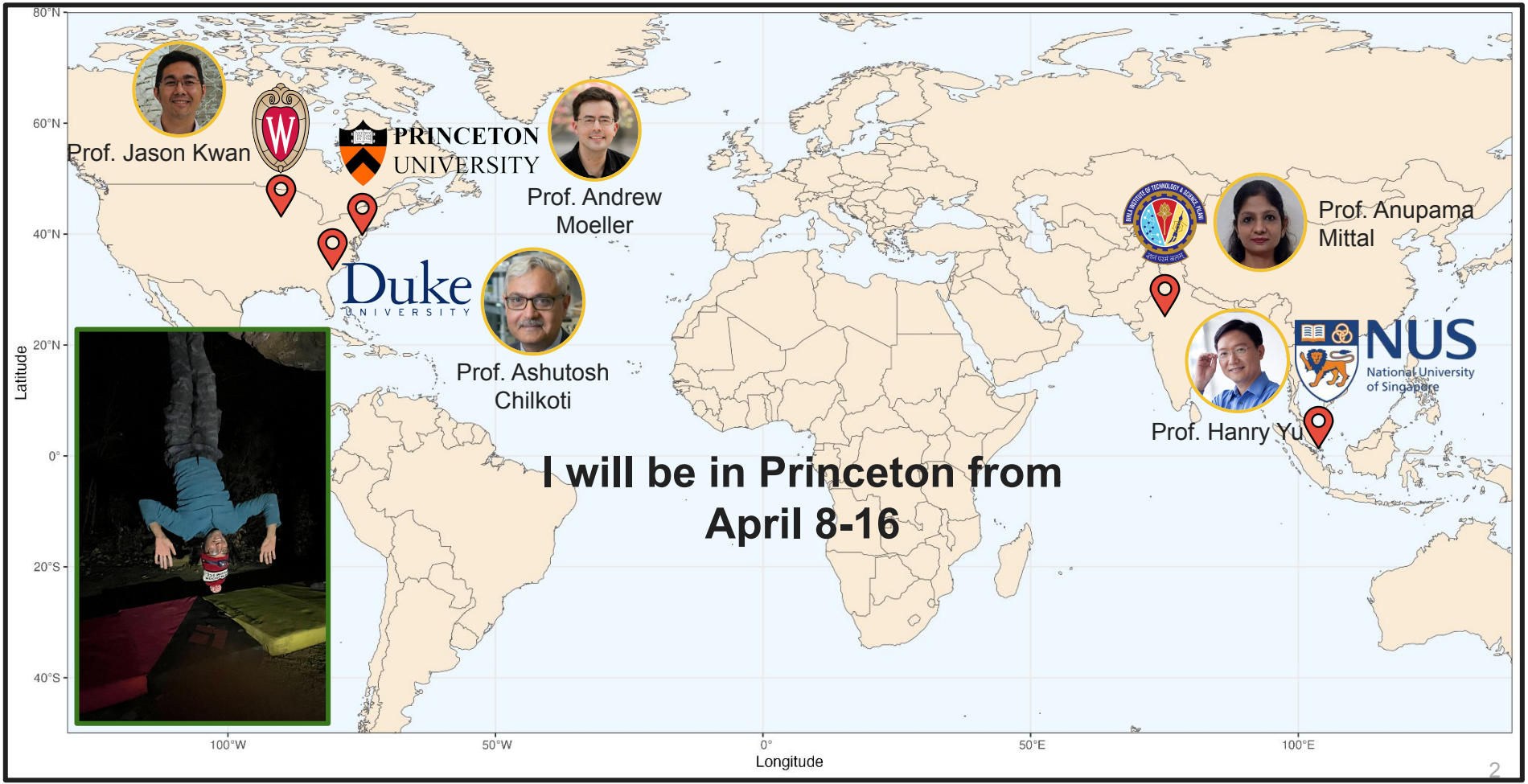
Slurm Tutorial

Siddharth Uppal
03/06/2025


Prof. Jason Kwan
PRINCETON UNIVERSITY
Prof. Andrew Moeller
Duke UNIVERSITY
Prof. Ashutosh Chilkoti
Prof. Anupama Mittal
NUS National University of Singapore
Prof. Hanry Yu
I will be in Princeton from April 8-16
Latitude
Longitude
80°N
60°N
40°N
20°N
0°
20°S
40°S
100°W
50°W
0°
50°E
100°E

Housekeeping

- **My goal**: Give a basic understanding of SLURM and inspire people to start using it
- All materials are available at: https://github.com/MoellerLabPU/tutorials
- Feel free to interrupt me at anytime during the talk
- If you have any issue DO NOT ~~hesitate to~~ contact me

Housekeeping

- **My goal**: Give a basic understanding of SLURM and inspire people to start using it
- All materials are available at: https://github.com/MoellerLabPU/tutorials
- Feel free to interrupt me at anytime during the talk
- If you have any issue DO NOT hesitate to contact me

What is SLURM ?

- Job scheduler workload manager for HPCs
- You submit a job to SLURM and it submits it to the server where the job will run

Why Should You Use SLURM Instead of Just Bash?

- **Resource Management**: Avoid conflicts and crashes from too many jobs running at the same time.
- **Automated Logging**: Captures standard output (stdout) and standard error (stderr)
- **Scalability -** Only use the amount of resources we have
- **Reproducible Workflows**

Basic SLURM Commands

- `sbatch`: Submits a job script for execution
- `squeue`: Shows the status of submitted jobs (yours and possibly others)
- `scancel`: Cancels/ kill a job (or set of jobs)

sbatch

Submits a job script for execution

```
sbatch <myScript.sh>

# Upon submission, SLURM will return a Job ID—for example:

Submitted batch job 123456
```

squeue

Shows the status of submitted jobs

```
# Show the status of ALL jobs submitted by EVERYONE

squeue

# Filter jobs by user:

squeue -u suppal
```

scancel

Cancels/ kill a job

Get the <JOB_ID> for the job from `squeue`

```
scancel <JOB_ID>
```

Cancel all jobs of a user

```
scancel -u suppal
```

Basics of a running a script for SLURM

- A special `#SBATCH` directive is used before any command that is passed to SLURM, example `#SBATCH --mem=4000` or `#SBATCH --cpus-per-task=1`
- Are just **Bash** scripts. That means:
  - All bash commands, eg. `grep, echo, cat, sed`, etc work as normal
  - `PATH` can be exported before running a script
  - Command line arguments can be provided with code, eg. --cpus, --outDir
- If using a conda environment, activate the environment before running the SLURM script
- SAME script template can be used for both Cornell and Princeton servers

Example script 1

```
#!/bin/bash
#SBATCH --ntasks=1                    # Number of tasks to spawn
#SBATCH --nodes=1                     # Node count
#SBATCH --output=my_first_job.out     # Stdout file name
#SBATCH --error=my_first_job.err      # Stderr file name
#SBATCH --time=00:05:00     # Total runtime (HH:MM:SS). Kill the jobs if it runs longer
#SBATCH --mem=4000                    # Memory in Mb
#SBATCH --cpus-per-task=4             # CPUs to use

# Optional: Email notifications
#SBATCH --mail-user=<YourNetID>@princeton.edu
#SBATCH --mail-type=BEGIN,END,FAIL,ALL

# Print some info about the job
echo "Running job ID: $SLURM_JOB_ID on node(s): $SLURM_NODELIST"
echo "Current working directory is $(pwd)"

# Run your computational task
python my_script.py —-cpus 4 --outDir /workdir1/sidd/myDir
```

Example script 2

```
#!/bin/bash
#SBATCH --ntasks=1                    # Number of tasks to spawn
#SBATCH --nodes=1                     # Node count
#SBATCH --output=my_first_job.out     # Stdout file name
#SBATCH --error=my_first_job.err      # Stderr file name
#SBATCH --time=00:05:00     # Total runtime(HH:MM:SS). Kill the jobs if it runs longer
#SBATCH --mem=4000                    # Memory in Mb
#SBATCH --cpus-per-task=4             # CPUs to use

# Running a tool
export PROKKA_CMD="singularity run -C -B $PWD --pwd $PWD
/programs/prokka-1.14.5-r9/prokka.sif"

$PROKKA_CMD prokka -out myresult_dir --cpus 4myInputGenome.fna

# Export PATH and then run
export PATH=/programs/prodigal-2.6.3:$PATH
prodigal -i myInputGenome.fna -o my.genes -a my.proteins.faa
```

Time to get your hands dirty

```
# Clone the repository
git clone https://github.com/MoellerLabPU/tutorials.git
cd tutorials/slurm_tutorial

# Run prodigal.sh
nano prodigal.sh                          #  Change the outDir path
sbatch prodigal.sh

# Run square_multithread.sh
nano square_multithread.sh                #  Change the outDir path
sbatch square_multithread.sh
```

Snakemake + SLURM == Maximum efficiency, minimal chaos

- Snakemake can automatically submit jobs to SLURM
- A `yaml` config specifies the parameters to submit to SLURM
- `Yaml` should be named `config.yaml` and needs to be inside a directory
- Cookiecutter `yaml` profile: https://github.com/jdblischak/smk-simple-slurm?tab=readme-ov-file
- Need to install cluster-generic plugin: https://bioconda.github.io/recipes/snakemake-executor-plugin-cluster-generic/README.html

```
Snakemake --profile cornell_profile
```

profile/config.yaml

```
executor: cluster-generic

cluster-generic-submit-cmd:
  mkdir -p logs/`date +"%d-%m-%y"`/{rule} &&
  sbatch
    --cpus-per-task={threads}
    --mem={resources.mem_mb}
    --time={resources.time}
    --job-name=smk-{rule}-{wildcards}
    --output=logs/`date +"%d-%m-%y"`/{rule}/{rule}-{wildcards}-%j.out
    --parsable

default-resources:
  - mem_mb=4000
  - time="10:00:00"

restart-times: 0
max-jobs-per-second: 100
max-status-checks-per-second: 1
latency-wait: 60
jobs: 100
keep-going: True
rerun-incomplete: True
printshellcmds: True
use-conda: True
cluster-generic-cancel-cmd: scancel
```

General guidelines for lab & tips

- Best practice is to ALWAYS submit a job using SLURM

- Check Your Resource Requests:
  - Requesting too little can causes crashes (“Out of Memory”) or slow runs
  - Requesting too much might lead to long wait times or wasted resources

- Don’t forget to load any modules and/or export PATHS that you need within your SLURM script

Topics for future tutorials

- `ssh` to servers without using your password
- Connecting code editor (eg. VS Code, Atom) to remote server
- Using LLMs (eg. chatGPT, deepSeek) for debugging and coding
- Anything else???