Introduction to SLURM

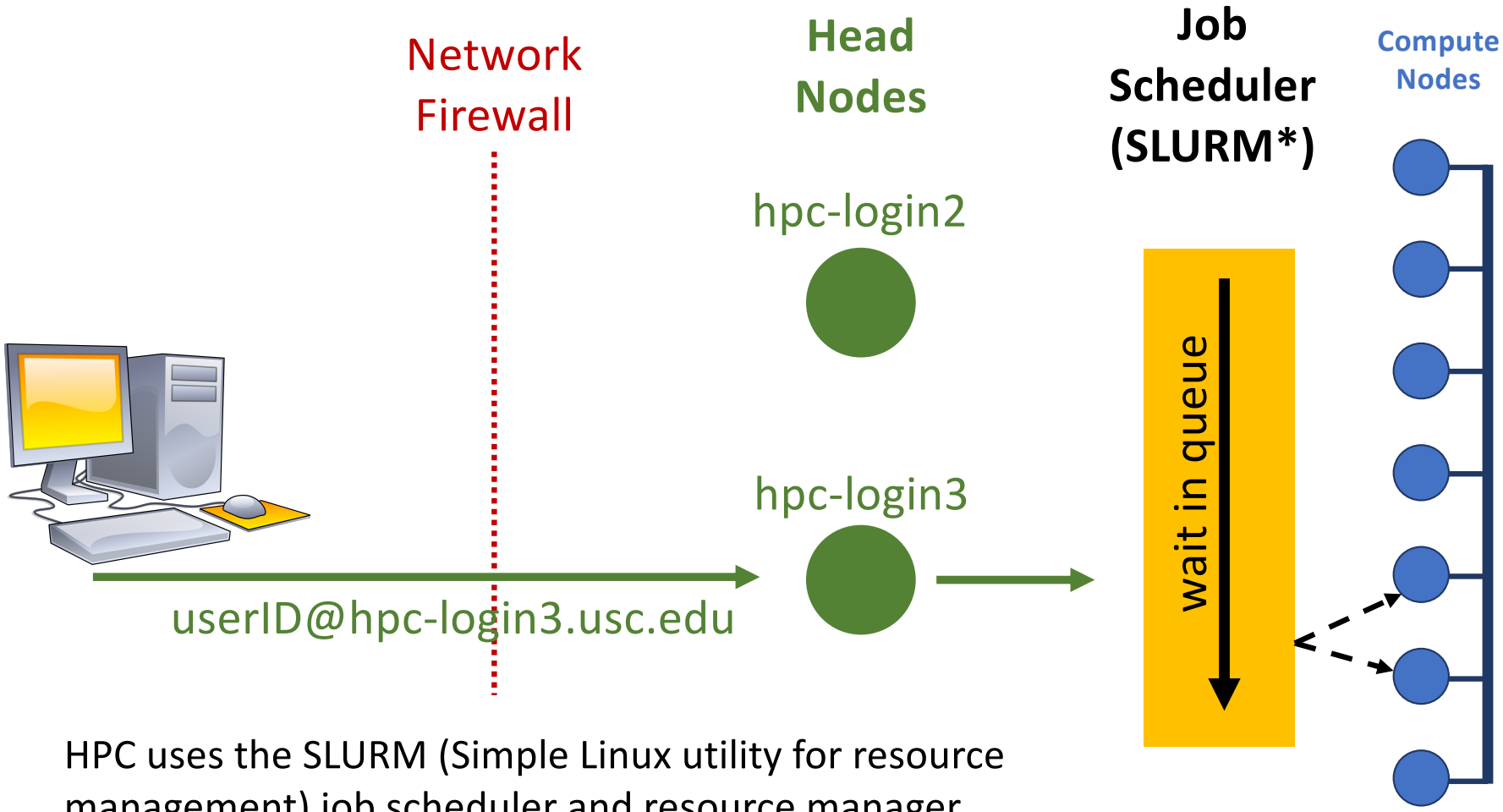
Minhui Chen, Ph.D.

minhuic@usc.edu

@chen_cmh

Material

- https://hpcc.usc.edu/gettingstarted/
- https://hpcc.usc.edu/support/documentation/slurm/
- HPC Computing Workshops presentations in /home/rcf-proj/workshop/handouts
  - HPCNewUser_20190531.pdf
  - HPCIntroHPC_20190607.pdf
  - handy_slurm_commands_201907.pdf
  - HPCAdvancedP1-20190726.pdf



HPC uses the SLURM (Simple Linux utility for resource management) job scheduler and resource manager. Jobs are scheduled based on order submitted, number & types of nodes requested and time required.

Running on HPC: Easy as 1-2-3

1. Plan ahead

- Organize project directories
- Install software, transfer data
- Think about your job requirements
  - amount of memory
  - number of i/o files
  - size of data

2. Test interactively

- Use salloc/srun
  - test run commands
  - check paths, results
- Experiment with resource allocation
  - number cores (cpus)
  - amount of memory
- Start small!
  - *then* scale

3. Run remotely

- Create Slurm script
  e.g., myjob.slurm
- Submit job to queue
  $sbatch myjob.slurm
- Monitor your job
  $squeue --user <uname>
- Check results
  $less slurm-<jobid>.out

Test your job interactively

- Slurm has a special job submission mode that allows you to access computing resources interactively
  - Example: request 1 processor/cpu/core for one hour

    ```
    $ salloc --ntasks=1 --time=1:00:00
    ```
  - You can test your program until the requested time expires

    ```
    $ source /usr/usc/hello_usc/3.0/setup.sh
    $ hello_usc
    ```
  - Extremely useful for compiling/debugging/testing your code and preparing job scripts

1. Request and wait for resources

```
[minhuic@hpc-login3 ~]$ cd /home/rcf-proj/cc2/scripts/tutorial
[minhuic@hpc-login3 tutorial]$ salloc --ntasks=1 --time=1:00:00
salloc: Pending job allocation 5175112
salloc: job 5175112 queued and waiting for resources
```

2. Get resources

```
[minhuic@hpc-login3 tutorial]$ salloc --ntasks=1 --time=1:00:00
salloc: Pending job allocation 5175112
salloc: job 5175112 queued and waiting for resources
salloc: job 5175112 has been allocated resources
salloc: Granted job allocation 5175112
salloc: Waiting for resource configuration
salloc: Nodes hpc1119 are ready for job
---------- Begin SLURM Prolog ----------
Job ID:       5175112
Username:     minhuic
Accountname:  lc_cc2
Name:         sh
Partition:    quick
Nodelist:     hpc1119
TasksPerNode: 1
CPUsPerTask:  Default[1]
TMPDIR:       /tmp/5175112.quick
SCRATCHDIR:   /staging/scratch/5175112
Cluster:      uschpc
HSDA Account: false
---------- 2019-12-02 15:53:42 ---------
[minhuic@hpc1119 tutorial]$
```

3. Test our program

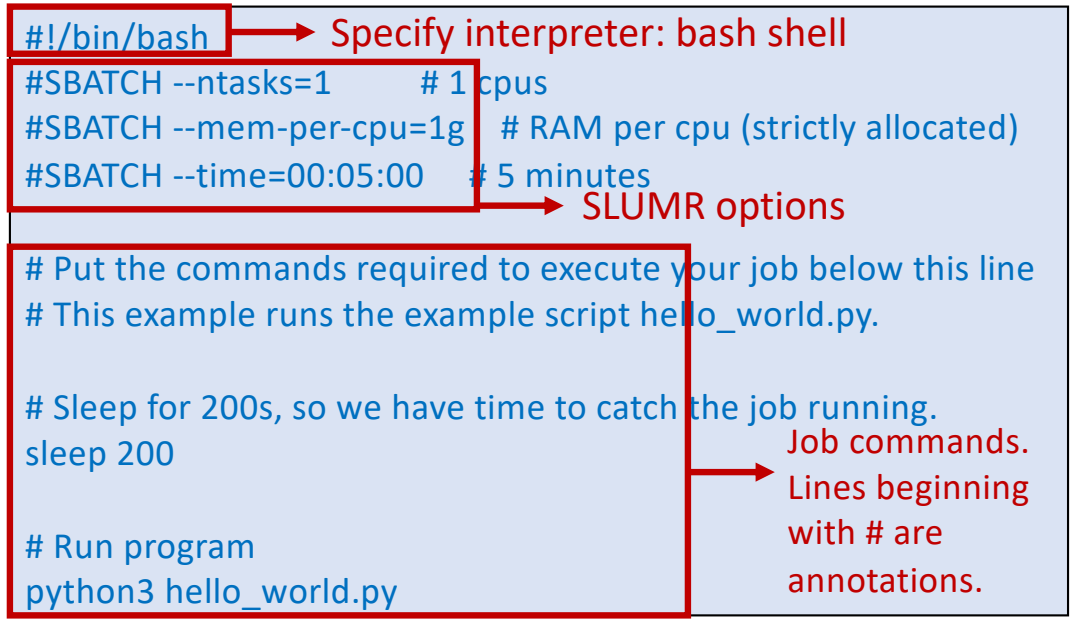
```
[minhuic@hpc1119 tutorial]$ python3 hello_world.py

      Hello world!
```

Submit a batch job

- Use a job script to submit a batch job to the cluster
  1. Add SLURM computing resource requests
  2. Add the commands you need to run your program
  3. Submit your job to the queue by typing:

     $ sbatch hello_world.slurm

Example job script: hello_world.slurm



Job monitoring

- squeue: view information about queued jobs

  $ squeue –u <user>

```
[minhuic@hpc-login3 gprs]$ squeue -u minhuic
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           5179183    conti gprs.Qx.  minhuic  R    1:17:39      1 hpc3936
           5179186    conti gprs.Qx.  minhuic  R    1:17:39      1 hpc3936
```

  Column ST displays the job status

  - PD (pending) job is queued and waiting to be executed
  - R (running) job is currently running
  - CD (completed) job has completed

- scancel: cancel jobs

  $ scancel <job_id>

1. Submit batch job

```
[minhuic@hpc-login3 ~]$ cd /home/rcf-proj/cc2/scripts/tutorial
[minhuic@hpc-login3 tutorial]$ sbatch hello_world.slurm
Submitted batch job 5179893
```

2. View job

```
[minhuic@hpc-login3 tutorial]$ squeue -u minhuic
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
5179893     quick hello_wo  minhuic PD       0:00      1 (Resources)
```

```
[minhuic@hpc-login3 tutorial]$ squeue -u minhuic
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
5179893     quick hello_wo  minhuic R       0:17      1 hpc3688
```

3. Check results

```
[minhuic@hpc-login3 tutorial]$ cat slurm-5179893.out
---------- Begin SLURM Prolog ----------
Job ID:       5179893
Username:     minhuic
Accountname:  lc_cc2
Name:         helloUSC.slurm
Partition:    quick
Nodelist:     hpc3688
TasksPerNode: 8
CPUsPerTask:  Default[1]
TMPDIR:       /tmp/5179893.quick
SCRATCHDIR:   /staging/scratch/5179893
Cluster:      uschpc
HSDA Account: false
---------- 2019-12-02 22:48:43 ---------

      Hello world!
```

Slurm sbatch options

Syntax	Meaning
--account=<account>	Which account to charge cpu time
--partition=<partition_name>	Which partition to submit to, for condo owners
--ntasks=<amount>	Number of CPUs to use
--mem_per_cpu=<amount>	RAM per CPU
--gres=gpu:<GPU_TYPE>:<amount>	Type and number of GPU to request
--time=<amount>	How long job will run
--constraint=<attribute>	Node property to request (avx, IB, xeon)
--mail-user=<email>	Where to send email alerts
--mail-type=<BEGIN| END| FAIL| REQUEUE| ALL>	When to send email alerts
--output=<out_file>	Name of output file
--error=<error_file>	Name of error file
--job-name=<job_name>	Job name
For more details see https://slurm.schedmd.com/sbatch.html	

Also check --cpus-per-task

Should be --mem-per-cpu

Slurm queues/partitions

- There are 4 default queues in the general (public) partition
  - A partition is automatically selected for you based on wall time and node count
  - Each partition has different constraints

Queue Name	Maximum Wall Time	Max Node Count	Max Jobs/ User
main	24 hours	99	10
quick	2 hours	4	10
large	24 hours	256	1
long	336 hours	1	1
largemem (must request)	336 hours (14 days)	1	1
scavenge (must request)	168 hours (7 days)	500 cores	200

- Partition conti: privately owned nodes

```
#!/bin/bash
#SBATCH --partition=conti
#SBATCH --account=lc_dvc
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=10G
#SBATCH --time=100:00:00
```

Parallel jobs

- Job arrays
  - Job Arrays let you reuse a job script
- srun
  - Slurm's srun utility can launch parallel processes
  - srun <command> will launch <command> on all "tasks“
- Example script: count # of SNPs in a file
  - The dataset itself is split to 22 chromosomes, and our script will count the number in each chromosome simultaneously using job arrays or srun

Job arrays

Example script: job_array.slurm

```
#!/bin/bash
#SBATCH --partition=conti
#SBATCH --account=lc_dvc
#SBATCH --job-name=CountingSNPsJobArray
#SBATCH --output=%x_%A_%a.out  # %x %A %a represent jobname jobid
jobarrayindex
#SBATCH --error=%x_%A_%a.err
#SBATCH --array=1-22        # specify array arrange

# script to count number of SNPs per chromosome in HapMap3 (/home/rcf-
proj/cc2/datasets/public/hapmap3/hapmap3_r1_hg19_fwd.MERGED.qc.poly
.bim)

# define variable chr(omosome)
chr=$SLURM_ARRAY_TASK_ID
# define output file name for each chromosome
outfile=$(printf "chr%02d.job_array.txt" $chr)

# Sleep for 200s, so we have time to catch the job running.
#sleep 200
# grep -P "^${chr}\t": extract out lines (SNPs) on chromosome $chr
# wc -l: count the number of lines (SNPs)
grep -P "^${chr}\t" /home/rcf-
proj/cc2/datasets/public/hapmap3/hapmap3_r1_hg19_fwd.MERGED.qc.poly
.bim | wc -l > $outfile
```

1. Submit batch job

```
[minhuic@hpc-login3 ~]$ cd /home/rcf-proj/cc2/scripts/tutorial
[minhuic@hpc-login3 tutorial]$ sbatch job_array.slurm
Submitted batch job 5273662
```

2. View job

```
[minhuic@hpc-login3 tutorial]$ squeue -u minhuic
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
5273662_1    conti Counting  minhuic  R       0:54      1 hpc3936
5273662_2    conti Counting  minhuic  R       0:54      1 hpc3937
5273662_3    conti Counting  minhuic  R       0:54      1 hpc3937
...
5273662_21    conti Counting  minhuic  R       0:54      1 hpc3803
5273662_22    conti Counting  minhuic  R       0:54      1 hpc3804
```

3. Check results

```
[minhuic@hpc-login3 tutorial]$ ls
chr01.job_array.txt ... chr22.job_array.txt
CountingSNPsJobArray_5273662_1.err
CountingSNPsJobArray_5273662_1.out
...
CountingSNPsJobArray_5273662_22.err
CountingSNPsJobArray_5273662_22.out
[minhuic@hpc-login3 tutorial]$ cat chr01.job_array.txt
130057
```

srun

Example script: srun.slurm

```
#!/bin/bash
#SBATCH --partition=conti
#SBATCH --account=lc_dvc
#SBATCH --job-name=CountingSNPsSRUN
#SBATCH --output=slurm.%x.%j.out  # %x %j represent jobname jobid
#SBATCH --error=slurm.%x.%j.err
#SBATCH --ntasks=22 # 22 tasks (processes)

# script to count number of SNPs per chromosome in HapMap3 (/home/rcf-
proj/cc2/datasets/public/hapmap3/hapmap3_r1_hg19_fwd.MERGED.qc.poly.bim)

# Sleep for 200s, so we have time to catch the job running.
#sleep 200
# use for loop to iterate 22 chromosomes
# grep -P "^${chr}\t": extract out lines (SNPs) on chromosome $chr
# wc -l: count the number of lines (SNPs)
# srun -n1 -N1 <command>: request for 1 node 1 task to run the <command>
# '&' character at the end of the srun command let the command run in the background
for chr in {1..22}
do
   outfile=$(printf "chr%02d.srun.txt" $chr)
   srun -n1 -N1 grep -P "^${chr}\t" /home/rcf-
proj/cc2/datasets/public/hapmap3/hapmap3_r1_hg19_fwd.MERGED.qc.poly.bim | wc -l > $outfile &
done
# wait for all running processes to finish
wait
```

1. Submit batch job

```
[minhuic@hpc-login3 ~]$ cd /home/rcf-proj/cc2/scripts/tutorial
[minhuic@hpc-login3 tutorial]$ sbatch srun.slurm
Submitted batch job 5274520
```

2. View job

```
[minhuic@hpc-login3 tutorial]$ squeue -u minhuic
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
5274520     conti Counting  minhuic  R       3:02      2 hpc[3559,3797]
```

3. Check results

```
[minhuic@hpc-login3 tutorial]$ ls
chr01.srun.txt ... chr22.srun.txt
slurm.CountingSNPsSRUN.5274520.err
slurm.CountingSNPsSRUN.5274520.out
[minhuic@hpc-login3 tutorial]$ cat chr01.srun.txt
130057
```