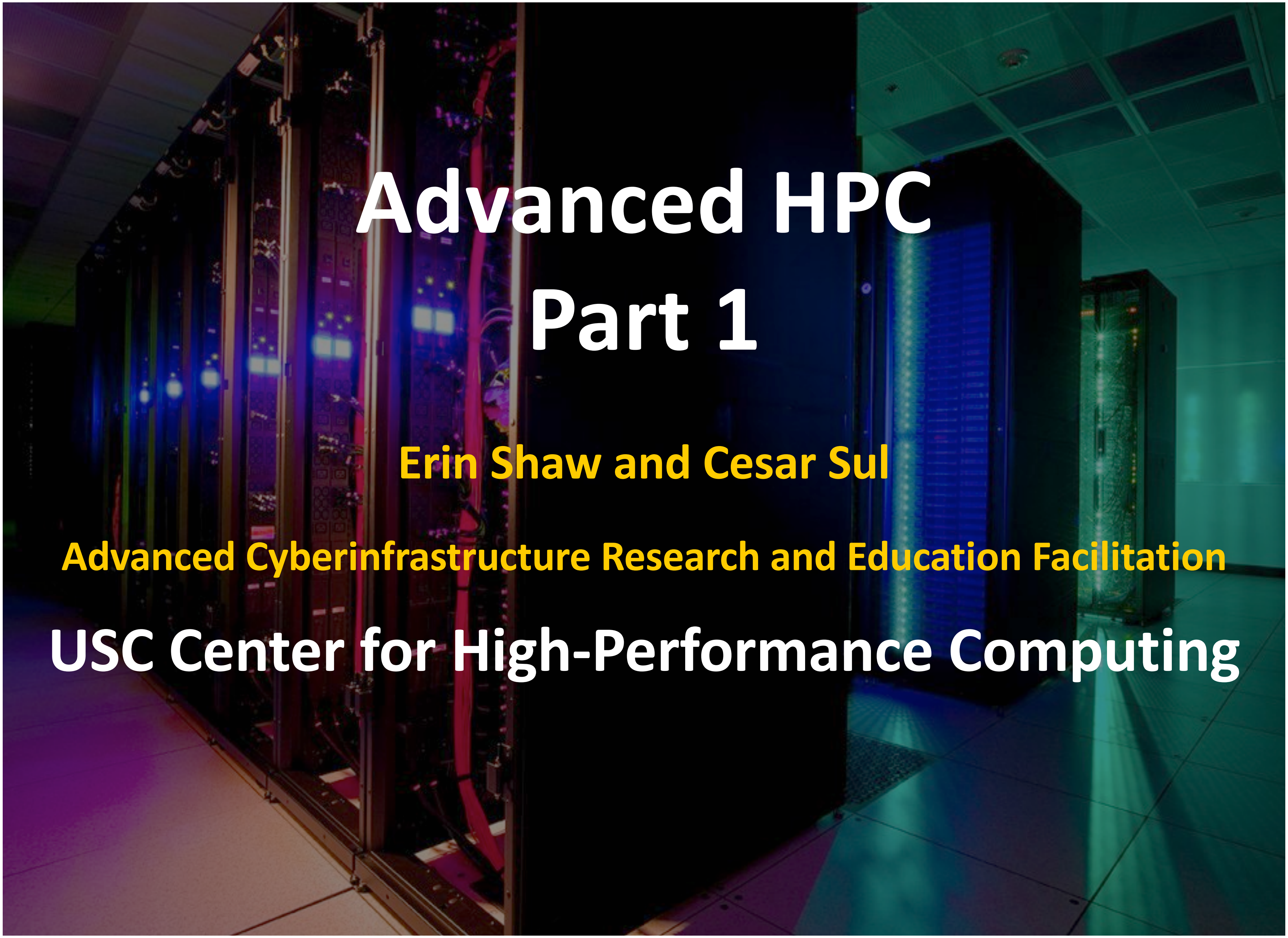


Advanced HPC
Part 1

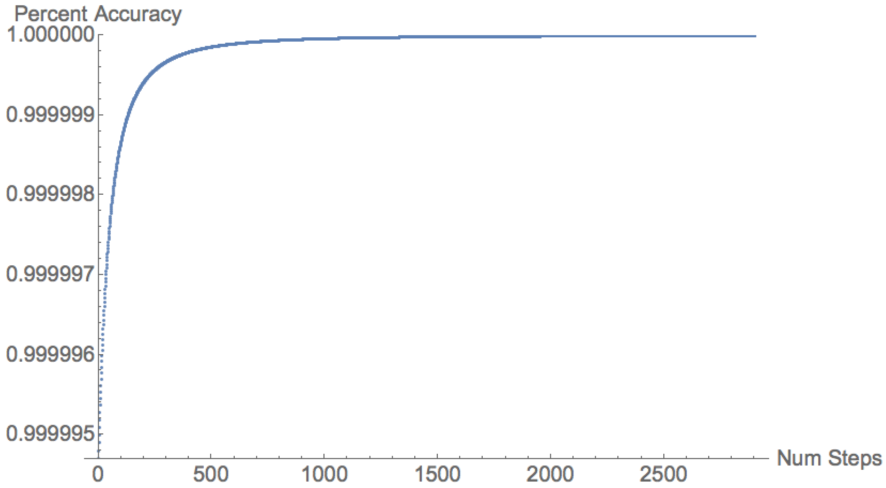
Erin Shaw and Cesar Sul

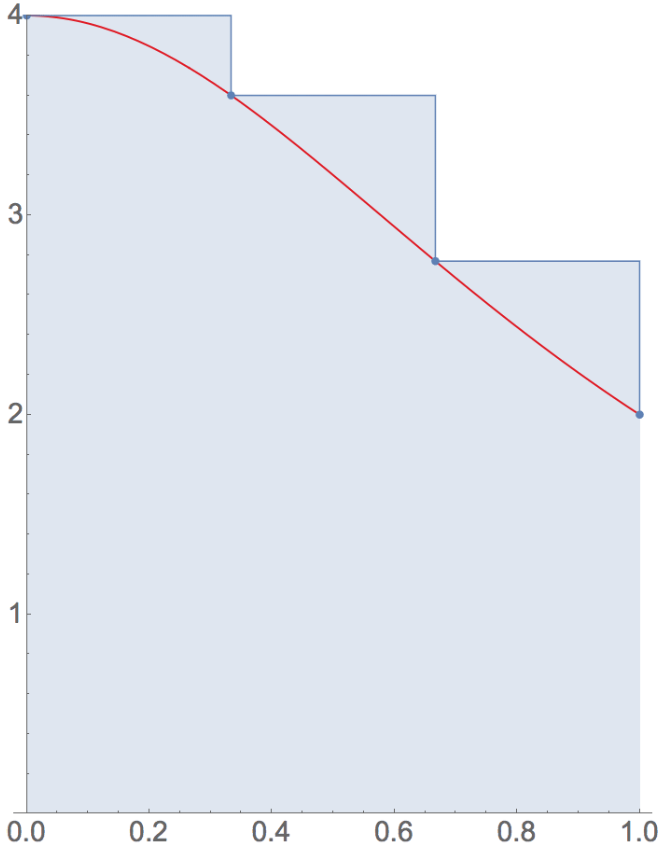
Advanced Cyberinfrastructure Research and Education Facilitation

USC Center for High-Performance Computing

Calculate Pi

- Numerically evaluate pi
  - $\int_0^1 \frac{4}{\sqrt{1+x^2}} dx = \pi$
  - Sample size vs accuracy?
- Have to run many times





Calculate pi

- How to run

```
./calc_pi -n 100
num_steps=100

pi=3.1515759869231288
```

- Want to run from n=1 to n=1000
- Want to collect results into one file
- Want to run statistics on the one file

Job Arrays on Slurm

- Job Arrays let you reuse a job script
- Each job does a similar task
- Read more on Slurm's Job Array documentation
- Example files:
  - /home/rcf-proj/workshop/adv_hpc/job_array

Job Arrays on Slurm

```
#!/bin/bash

#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --time=00:30:00
#SBATCH --export=none
#SBATCH --array=1-10

./pi_calc -n $SLURM_ARRAY_TASK_ID >
output${SLURM_ARRAY_TASK_ID}.txt
```

Job Arrays on Slurm

- Similar to normal job
- Include `#SBATCH --array=1-10`
- Job array elements differentiate themselves with `$SLURM_ARRAY_TASK_ID`

- [ttroj@hpc-login3 advanced_hpc]$ myqueue

```
                                                                           MIN_   START_        TIME_                NODELIST
 JOBID   USER  ACCOUNT  PARTITION        NAME        TASKS CPUS_PER_TASK MEMORY  TIME   TIME   LIMIT      STATE     (REASON)
3959285 ttroj  lc_tt1    scavenge  job_array.slurm     1         1         1G     0     0:00 30:00:00 CONFIGURING hpc0682
3959286 ttroj  lc_tt1    scavenge  job_array.slurm     1         1         1G     0     0:00 30:00:00 CONFIGURING hpc0682
3959287 ttroj  lc_tt1    scavenge  job_array.slurm     1         1         1G     0     0:00 30:00:00 CONFIGURING hpc0682
3959288 ttroj  lc_tt1    scavenge  job_array.slurm     1         1         1G     0     0:00 30:00:00 CONFIGURING hpc0682
3959289 ttroj  lc_tt1    scavenge  job_array.slurm     1         1         1G     0     0:00 30:00:00 CONFIGURING hpc0682
3959290 ttroj  lc_tt1    scavenge  job_array.slurm     1         1         1G     0     0:00 30:00:00 CONFIGURING hpc0682
3959291 ttroj  lc_tt1    scavenge  job_array.slurm     1         1         1G     0     0:00 30:00:00 CONFIGURING hpc0682
3959292 ttroj  lc_tt1    scavenge  job_array.slurm     1         1         1G     0     0:00 30:00:00 CONFIGURING hpc0682
3959293 ttroj  lc_tt1    scavenge  job_array.slurm     1         1         1G     0     0:00 30:00:00 CONFIGURING hpc0683
3959284 ttroj  lc_tt1    scavenge  job_array.slurm     1         1         1G     0     0:00 30:00:00 CONFIGURING hpc0683
```

Job Arrays on Slurm

- This works but it generates messy output
- Output files are organized alphabetically, not numerically

```
$grep pi output*.txt

output10.txt:pi=3.2399259889071588
output1.txt:pi=4.0000000000000000
output2.txt:pi=3.6000000000000001
output3.txt:pi=3.4564102564102557
output4.txt:pi=3.3811764705882354
output5.txt:pi=3.3492611381098898
output6.txt:pi=3.3036297331379298
output7.txt:pi=3.2810484527641703
output8.txt:pi=3.2639884944910889
output9.txt:pi=3.2506461552653931
```

Job Arrays on Slurm

- We can "zero pad" our files
  - output1.txt -> output01.txt
  - Alphabetic sorting also becomes numeric sorting
- Change two lines in script

```
outfile=$(printf "output%02d.txt" $SLURM_ARRAY_TASK_ID)

./pi_calc -n $SLURM_ARRAY_TASK_ID > $outfile
```

- %02d means take the variable `$SLURM_ARRAY_TASK_ID` and put as many zeros as required to make it 2 characters wide

Job Arrays on Slurm

```
#!/bin/bash

#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --time=00:30:00
#SBATCH --export=none
#SBATCH --array=1-10

outfile=$(printf "output%02d.txt" $SLURM_ARRAY_TASK_ID)

./pi_calc -n $SLURM_ARRAY_TASK_ID > $outfile
```

Job Arrays on Slurm

- Organized files makes our lives easier down the line

```
$grep pi output*.txt

output01.txt:pi=4.0000000000000000
output02.txt:pi=3.6000000000000001
output03.txt:pi=3.4564102564102557
output04.txt:pi=3.3811764705882354
output05.txt:pi=3.3349261138109898
output06.txt:pi=3.3036297331379298
output07.txt:pi=3.2810484527641703
output08.txt:pi=3.2639884944910889
output09.txt:pi=3.2506461552653931
output10.txt:pi=3.2399259889071588
```

Job Arrays on Slurm

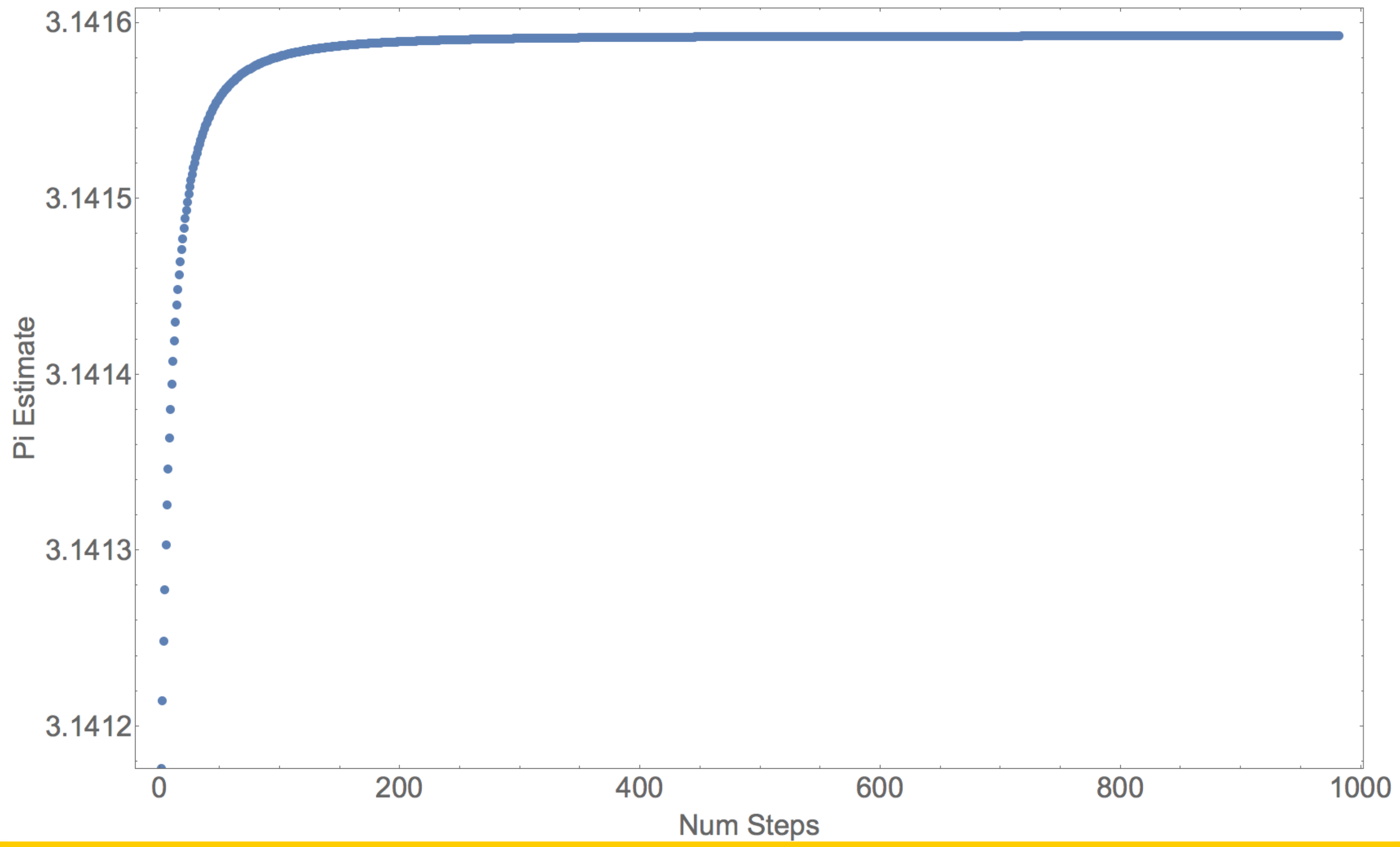
- Organized files makes our lives easier down the line

```
$grep pi output*.txt | sed 's/^.*=//'

4.0000000000000000
3.6000000000000001
3.4564102564102557
3.3811764705882354
3.3349261138109898
3.3036297331379298
3.2810484527641703
3.2639884944910889
3.2506461552653931
3.2399259889071588
```

Look for lines with 'pi' in them. Display only text after '=' character

- We don't have to parse this later

Summary of results



Job Arrays on Slurm

- Pros
  - Only need 1 job script
  - See all steps in 1 file
  - (Can be) Easy to set up
  - Many jobs can be queued up

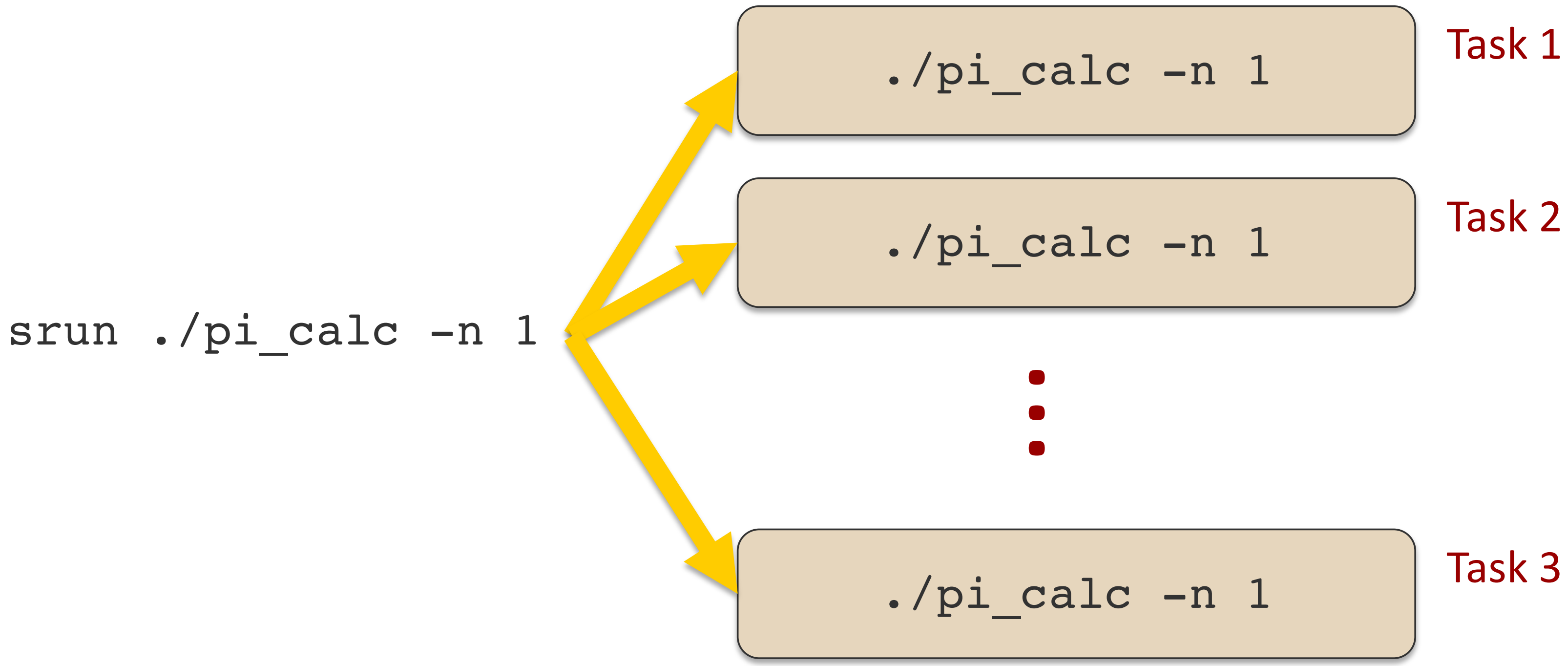
- Cons
  - 10 jobs running limit in main partition
  - Node resources may sit idle (depending on program)

srun on Slurm

- Slurm's srun utility can launch parallel jobs
- `srun <command>` will launch `<command>` on all "tasks"

```
$ salloc --ntasks=4 --cpus-per-task=8
$ hostname
hpc0972
$ srun hostname
hpc0971
hpc0972
hpc0972
hpc0971
$ srun --ntasks=2 hostname
hpc0972
hpc0971
```

srun on Slurm



```
srun ./pi_calc -n 1
```

```
./pi_calc -n 1
```

Task 1

```
./pi_calc -n 1
```

Task 2

⋮

```
./pi_calc -n 1
```

Task 3

How to get unique behavior on each task?

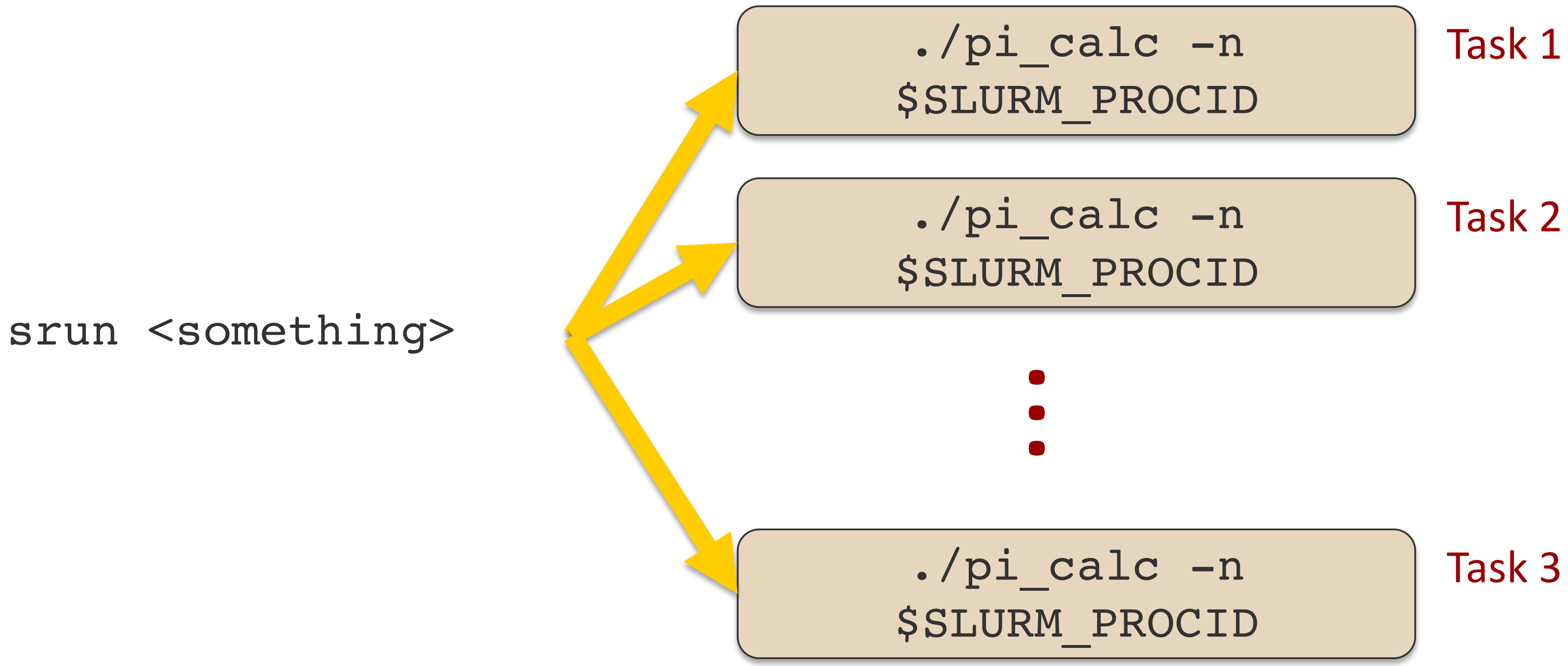
srun on Slurm

```
#!/bin/bash

#SBATCH --ntasks=10
#SBATCH --cpus-per-task=1
#SBATCH --time=00:30:00
#SBATCH --export=none

srun ./pi_calc -n 1
```

srun and Tasks

- What is a "task"?
- According to Slurm it’s a “process”
  - pi_calc is a process
- You can assign resources per process
  - cpus-per-task
  - mem-per-cpu
- Slurm will not split up tasks across nodes
- Just like $SLURM_ARRAY_ID, tasks can have $SLURM_ PROCID
  - When launched with srun, $SLURM_PROCID ranges from 0 to N-1

srun on Slurm



```
srun <something>
```

```
./pi_calc -n $SLURM_PROCID
```

Task 1

```
./pi_calc -n $SLURM_PROCID
```

Task 2

⋮

```
./pi_calc -n $SLURM_PROCID
```

Task 3

srun on Slurm

```
#!/bin/bash

#SBATCH --ntasks=10
#SBATCH --cpus-per-task=1
#SBATCH --time=00:30:00
#SBATCH --export=none

srun wrapper.sh
```

- We can't directly use srun `./pi_calc -n $SLURM_PROCID`
- Outside of srun, `$SLURM_PROCID` is set to 0.
- Create "wrapper" script that determines unique work

srun on Slurm

```
#!/bin/bash
# All environment settings are initialized here


# stepsize of 0 is nonsensical, we must increment it
stepsize=$((SLURM_PROCID + 1))

# fancy zero padding like in the job array example
outfile=$(printf "output%02d.txt" $stepsize)

./pi_calc -n $stepsize  > $outfile
```

- Each wrapper script has unique `$SLURM_PROCID`
- We had to process `$SLURM_PROCID` a little

Job Arrays on Slurm

- Organized files makes our lives easier down the line

```
$grep pi output*.txt

output01.txt:pi=4.0000000000000000
output02.txt:pi=3.6000000000000001
output03.txt:pi=3.4564102564102557
output04.txt:pi=3.3811764705882354
output05.txt:pi=3.3349261138109898
output06.txt:pi=3.3036297331379298
output07.txt:pi=3.2810484527641703
output08.txt:pi=3.2639884944910889
output09.txt:pi=3.2506461552653931
output10.txt:pi=3.2399259889071588
```

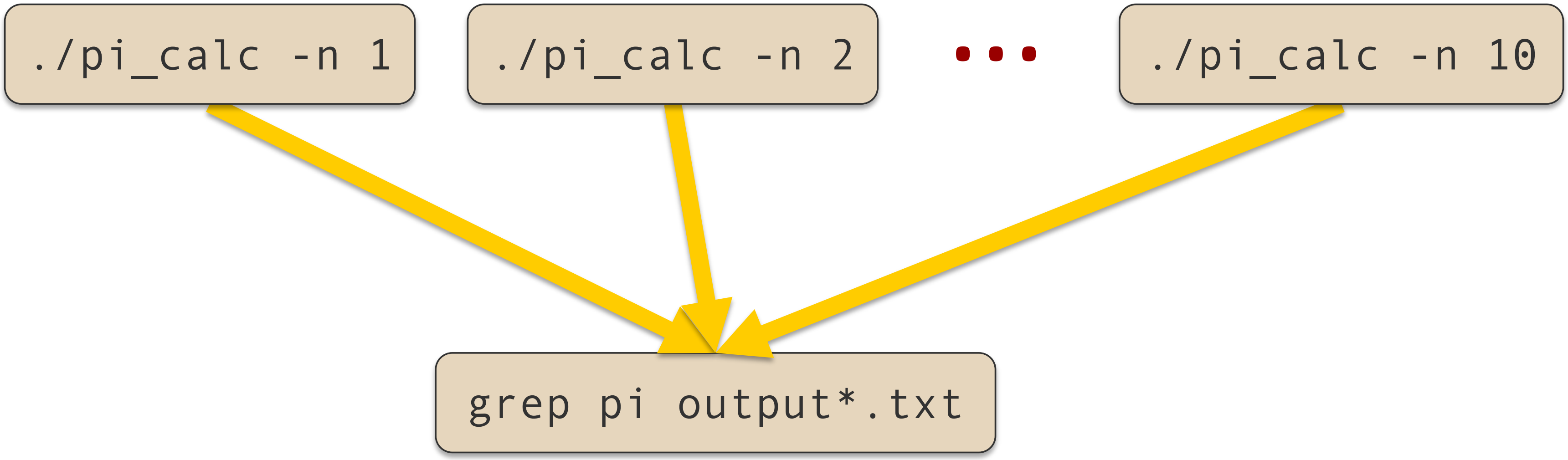
srun on Slurm

- Pros
  - Much more work per job

- Cons
  - **More complicated**
  - Not efficient for heterogeneous run times

Job Dependencies

- You can tell Slurm to put a job on hold until others finish
- Syntax is #SBATCH --dependency=<dependency_list>
- Read more on Slurm’s sbatch documentation



```
./pi_calc -n 1
./pi_calc -n 2
• • •
./pi_calc -n 10
grep pi output*.txt
```

Wait for all `pi_calc` jobs to complete, then run 'grep' stage

Job Dependencies

- Job script

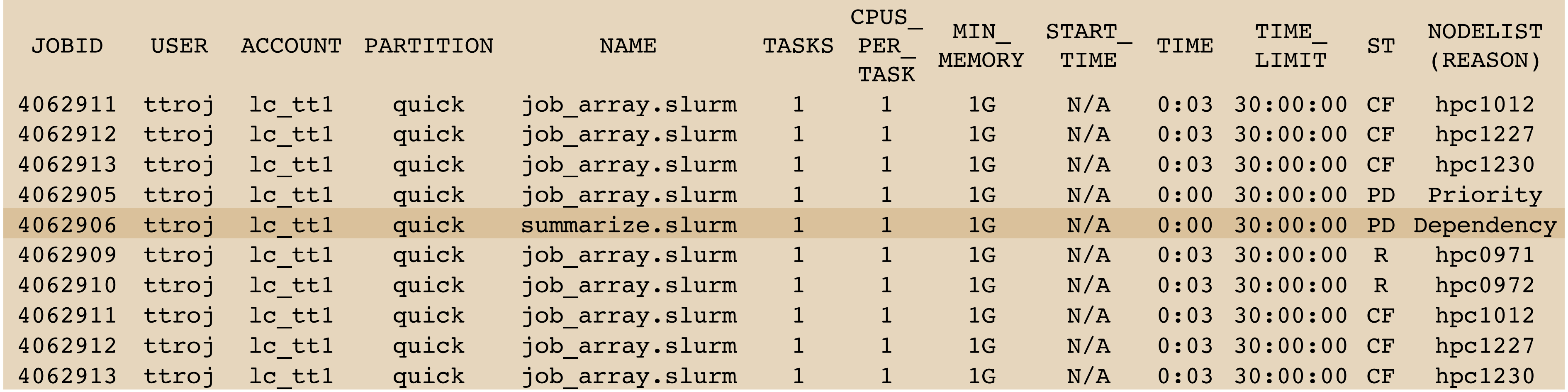
```
#!/bin/bash
#SBATCH --ntasks=1

grep pi output*.txt | sed 's/^.*=//' > summary.txt
```

- Dependency tracker - shell script

```
#!/bin/bash

# Launch first job
slurm_output=$(sbatch job_array.slurm)
# Get job id
dependency=$(echo $slurm_output | awk '{print $NF}')

# Assign dependency
sbatch --depend=afterok:${dependency} summarize.slurm
```

Job Dependencies

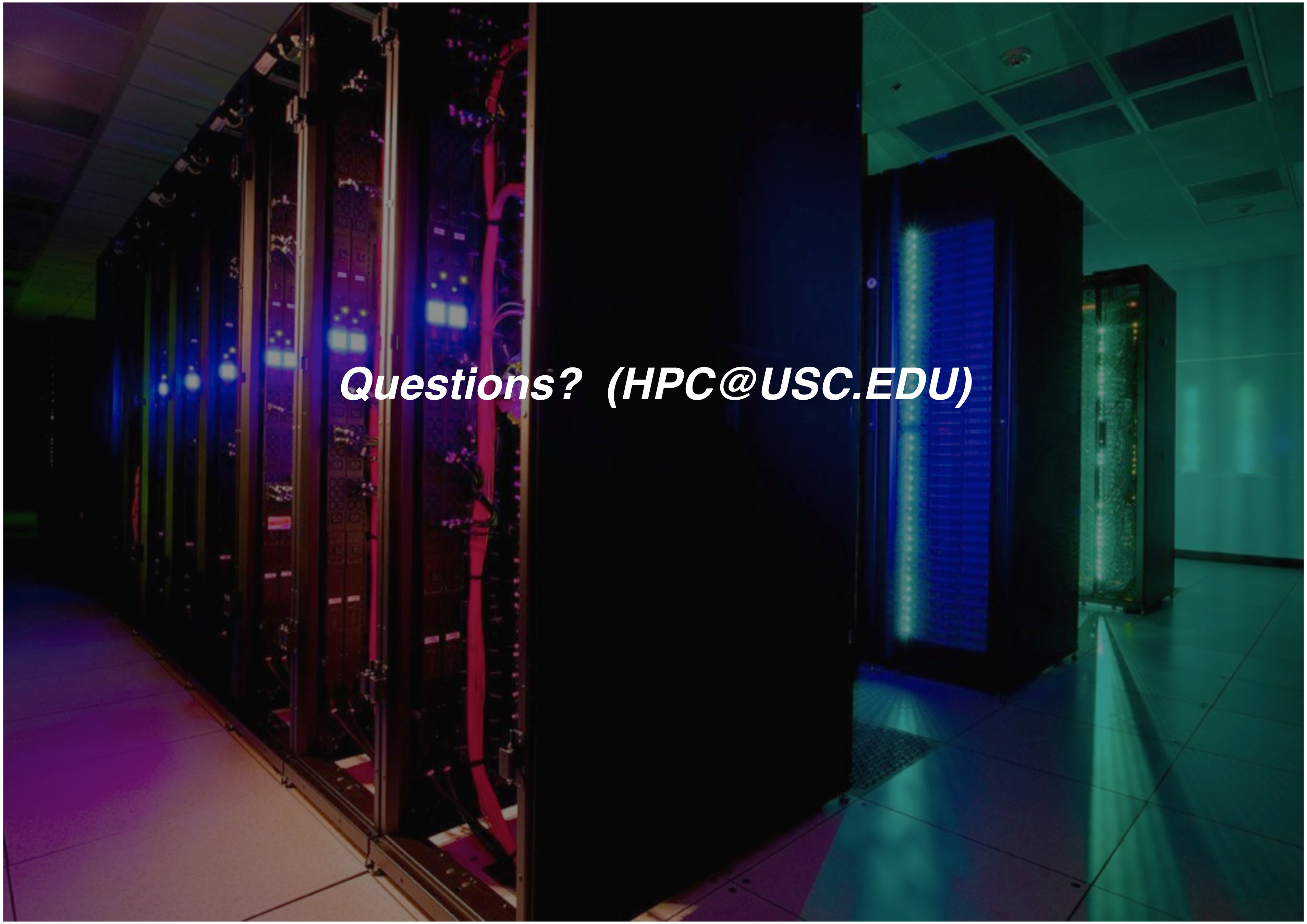
- Use dependency.sh to launch ALL jobs in workflow
- [ttroj@hpc-login3 advanced_hpc]$ myqueue



```
                                                     CPUS_
                                                             MIN_   START_          TIME_       NODELIST
 JOBID   USER   ACCOUNT  PARTITION       NAME        TASKS   PER_                                ST
                                                                    MEMORY   TIME   TIME   LIMIT    (REASON)
                                                      TASK
4062911  ttroj  lc_tt1     quick   job_array.slurm     1      1      1G     N/A    0:03  30:00:00  CF   hpc1012
4062912  ttroj  lc_tt1     quick   job_array.slurm     1      1      1G     N/A    0:03  30:00:00  CF   hpc1227
4062913  ttroj  lc_tt1     quick   job_array.slurm     1      1      1G     N/A    0:03  30:00:00  CF   hpc1230
4062905  ttroj  lc_tt1     quick   job_array.slurm     1      1      1G     N/A    0:00  30:00:00  PD  Priority
4062906  ttroj  lc_tt1     quick   summarize.slurm     1      1      1G     N/A    0:00  30:00:00  PD Dependency
4062909  ttroj  lc_tt1     quick   job_array.slurm     1      1      1G     N/A    0:03  30:00:00   R   hpc0971
4062910  ttroj  lc_tt1     quick   job_array.slurm     1      1      1G     N/A    0:03  30:00:00   R   hpc0972
4062911  ttroj  lc_tt1     quick   job_array.slurm     1      1      1G     N/A    0:03  30:00:00  CF   hpc1012
4062912  ttroj  lc_tt1     quick   job_array.slurm     1      1      1G     N/A    0:03  30:00:00  CF   hpc1227
4062913  ttroj  lc_tt1     quick   job_array.slurm     1      1      1G     N/A    0:03  30:00:00  CF   hpc1230
```

Job Dependencies

```
[ttroj@hpc-login3 job_dependency]$ cat summary.txt
4.0000000000000000
3.6000000000000001
3.4564102564102557
3.3811764705882354
3.3349261138109898
3.3036297331379298
3.2810484527641703
3.2639884944910889
3.2506461552653931
3.2399259889071588
```



Questions? (HPC@USC.EDU)