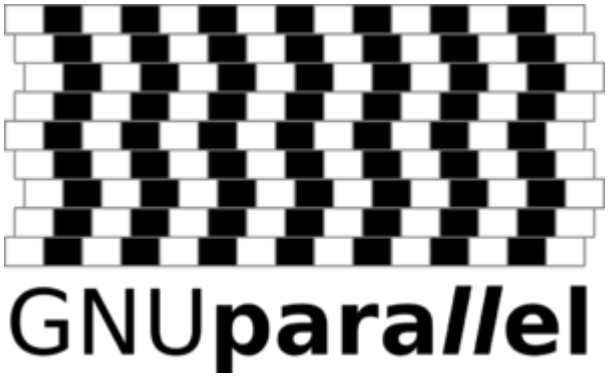
Efficient Submission of Serial Jobs

Jonathon Anderson
jonathon.anderson@colorado.edu

www.rc.colorado.edu

Slides: https://github.com/ResearchComputing/Final_Tutorials/



CURC loadbalancer

Batch job with one serial task

```
#!/bin/bash

module purge
module load python

time python matrix-multiply.py \
    data/input_0.csv
```

https://github.com/ResearchComputing/Parallelization_Workshop/blob/master/Day2-HTC/examples/bash-single-task.sh

Batch job with multiple tasks

Serial scripts run in sequence

```
#!/bin/bash

module purge
module load python

time (
    for input in data/input_{0..9}.csv
    do
        python matrix-multiply.py ${input}
    done
)
```

https://github.com/ResearchComputing/Parallelization_Workshop/blob/master/Day2-HTC/examples/bash-multiple-tasks.sh

Batch job with multiple tasks

Serial scripts run in parallel

```
#!/bin/bash

module purge
module load python

time (
    for input in data/input_{0..9}.csv
    do
        python matrix-multiply.py ${input} &
    done
    wait
)
```

https://github.com/ResearchComputing/Parallelization_Workshop/blob/master/Day2-HTC/examples/bash-multiple-background-tasks.sh

Exercise 1

Background bash jobs

In a terminal, background two `matrix-multiply.py` processes. Use the `wait` command to wait for all background commands to finish, and use the `time` command to time the total execution.

https://github.com/ResearchComputing/Parallelization_Workshop/blob/master/Day2-HTC/exercises/exercise1.md

Exercise 1 (solution)

Background bash jobs

```
$ module purge
$ module load python
$ time (
> python matrix-multiply.py \
>   data/input_0.csv &
> python matrix-multiply.py \
>   data/input_1.csv &
> wait)
```

Exercise 2

Background bash jobs in a Slurm job

Submit `bash-multiple-background-tasks.sh` as a Slurm job. Reduce the requested `--ntasks` and observe the execution time increase.

https://github.com/ResearchComputing/Parallelization_Workshop/blob/master/Day2-HTC/exercises/exercise2.md

Exercise 2 (solution)

Background bash jobs in a Slurm job

```
#!/bin/bash
#SBATCH --job-name exercise2-background-bash-jobs
#SBATCH --nodes 1
#SBATCH --ntasks 10
#SBATCH --output exercise2-%j.out
#SBATCH --time 00:02:00

module purge
module load python

time (
    for input in data/input_{0..9}.csv
    do
        python matrix-multiply.py ${input} &
    done
    wait
)
```

Exercise 2 (solution)

Background bash jobs in a Slurm job

```
$ sbatch --reservation tutorial1 \
> solutions/exercise2.sh

$ sbatch --reservation tutorial1 \
> --ntasks 5 \
> solutions/exercise2.sh
```

Bash scripting summary

- You don't need a special tool
- Bash is available almost anywhere
- More complex scripts require more advanced bash scripting experience
- Not great for managing large numbers of tasks

GNU parallel

- A shell tool for executing tasks in parallel using one or more computers
- In it's simplest form, a parallel replacement of a for loop
- Options to specify how many tasks should run in parallel, display output in order, limit resources and more!

GNU parallel

Two ways to run

```
$ module load gnu_parallel

$ parallel 'wc -l {}' ::: data/input_*.csv

$ ls data/input_*.csv | parallel 'wc -l {}'
```

GNU parallel

Comparison to bash

```
$ ls data/input_*.csv | parallel 'wc -l {}'

$ for input in data/input_*.csv
> do
>   wc -l ${input} &
> done
$ wait
```

Exercise 3

Convert a bash loop to GNU parallel

Convert a serial bash loop to a a parallel-execution GNU parallel command.

```
for input in data/input_{0..9}.csv
do
    python matrix-multiply.py ${input}
done
```

https://github.com/ResearchComputing/Parallelization_Workshop/blob/master/Day2-HTC/exercises/exercise3.md

Exercise 3 (solution)

convert a bash loop to GNU parallel

```
#!/bin/bash

module purge
module load python gnu_parallel

time (
    ls data/input_*.csv \
    | parallel python python matrix-multiply.py {}
)
```

GNU parallel

Useful options

View what commands parallel will run without executing them:

```
$ seq 10 | parallel --dry-run echo {}
```

Limit number of tasks running at one time:

```
$ seq 10 | parallel -j 2 echo {}
```

Wait until enough memory is available to start next task:

```
$ seq 10 | parallel --memfree 2G echo {}
```

Exercise 4

GNU parallel with Slurm

Convert the solution from exercise 2 ("background bash jobs in a Slurm job") to use GNU parallel rather than background Bash jobs, and submit it as a Slurm job. Reduce the requested `--ntasks` and observe the execution time increase.

https://github.com/ResearchComputing/Parallelization_Workshop/blob/master/Day2-HTC/exercises/exercise4.md

Exercise 4 (solution)

GNU parallel with Slurm

```
#!/bin/bash
#SBATCH --job-name htc-exercise4-gnu-parallel
#SBATCH --nodes 1
#SBATCH --ntasks 10
#SBATCH --output exercise4-%j.out
#SBATCH --time 00:02:00

module purge
module load python gnu_parallel

time (
    ls data/input_*.csv \
      | parallel python matrix-multiply.py {}
)
```

Exercise 4 (solution)

GNU parallel with Slurm

```
$ sbatch --reservation tutorial1 \
> solutions/exercise4.sh

$ sbatch --reservation tutorial1 \
> --ntasks 5 \
> solutions/exercise4.sh
```

GNU parallel summary

- Great for replacing and speeding up simple loops
- Control how your tasks are run
- Can run on multiple computers as well (may take some effort to get working with Slurm)
- Lots of examples and documentation online
- Useful tool outside of compute nodes too
- Not always available

Slurm job arrays

- Submit multiple sub-jobs from a single job script
- Array indices specified by `--array`
  - For example, `--array 0-9`
- Array index is available as
  - `$SLURM_ARRAY_TASK_ID` (in the job)
  - `%a` (in `--output`)
- Master job id is available as
  - `$SLURM_ARRAY_JOB_ID` (in the job)
  - `%A` (in `--output`)

Slurm job arrays

Example

```
#!/bin/bash
#SBATCH --array 0-9
#SBATCH --output slurm-array-%A.%a.out

echo "Master job id: ${SLURM_ARRAY_JOB_ID}"
echo "Array index: ${SLURM_ARRAY_TASK_ID}"
```

Exercise 5

Slurm job arrays

Convert the solution from exercise 2 ("background bash jobs in a Slurm job") and exercise 4 ("GNU parallel with Slurm") to use Slurm job arrays rather than a Bash loop or GNU parallel. Each array task should execute a single `matrix-multiply.py` process.

https://github.com/ResearchComputing/Parallelization_Workshop/blob/master/Day2-HTC/exercises/exercise5.md

Exercise 5 (solution)

Slurm job arrays

```
#!/bin/bash
#SBATCH --job-name htc-exercise5-slurm-arrays
#SBATCH --ntasks 1
#SBATCH --array 0-9
#SBATCH --output htc-exercise5-%A.%a.out
#SBATCH --time 00:01:00

module purge
module load python

time python matrix-multiply.py \
  data/input_${SLURM_ARRAY_TASK_ID}.csv
```

Exercise 5 (solution)

Slurm job arrays

```
$ sbatch --reservation tutorial1 \
> solutions/exercise5.sh
```

Slurm job arrays summary

- Standard Slurm semantics (and similar features exist on other schedulers)
- Incurs additional scheduling overhead
- Runs on multiple nodes

CURC loadbalancer

- Submitting hundreds of Slurm jobs is inefficient (even using job arrays)
- Balances serial applications using MPI (without needing knowledge of MPI!)
- Schedules tasks across multiple nodes from one job
  - Choose how many tasks will run at a time
  - Starts tasks in order (no control over output order)
  - Automatically replaces finished tasks with new tasks
  - Straightforward input format

CURC loadbalancer

Input file format

- One task per line
- Each task may run multiple commands, each command separated by a semicolon

```
for input in data/input_{0..9}.csv
do
    echo >>lb_cmd_file \
      "wc -l ${input} >$(basename ${input})-lines"
done
```

CURC loadbalancer

Input file format

```
wc -l data/input_0.csv >input_0.csv-lines
wc -l data/input_1.csv >input_1.csv-lines
wc -l data/input_2.csv >input_2.csv-lines
wc -l data/input_3.csv >input_3.csv-lines
wc -l data/input_4.csv >input_4.csv-lines
wc -l data/input_5.csv >input_5.csv-lines
wc -l data/input_6.csv >input_6.csv-lines
wc -l data/input_7.csv >input_7.csv-lines
wc -l data/input_8.csv >input_8.csv-lines
wc -l data/input_9.csv >input_9.csv-lines
```

CURC loadbalancer

Execution

```
#!/bin/bash


module purge
module load intel impi
module load loadbalance

mpirun lb lb_cmd_file
```

Exercise 6

CURC loadbalancer with Slurm

- Convert the solution from exercise 2 ("background bash jobs in a Slurm job"), exercise 4 ("GNU parallel with Slurm"), and exercise 5 ("Slurm job arrays") to use the CURC loadbalancer a Bash loop, GNU parallel, or Slurm arrays, and submit it as a Slurm job.
- Reduce the requested `--ntasks` and observe the execution time increase.
- Each task / input file line should execute a single `matrix-multiply.py` process.

https://github.com/ResearchComputing/Parallelization_Workshop/blob/master/Day2-HTC/exercises/exercise6.md

Exercise 6 (solution)

CURC loadbalancer with Slurm

```
#!/bin/bash
#SBATCH --job-name htc-exercise6-lb
#SBATCH --ntasks 10
#SBATCH --output htc-exercise6-%j.out
#SBATCH --time 00:02:00

module purge
module load python intel impi loadbalance

(
  for input in data/input_{0..9}.csv
  do
    output="${SLURM_JOB_ID}-$(basename ${input%.csv}).out"
    echo "python matrix-multiply.py ${input} >${output}"
  done
) >lb_cmd_file-${SLURM_JOB_ID}

time mpirun lb lb_cmd_file-${SLURM_JOB_ID}
```

Exercise 6 (solution)

CURC loadbalancer with Slurm

```
$ sbatch --reservation tutorial1 \
> solutions/exercise6.sh

$ sbatch --reservation tutorial1 \
> --ntasks 5 \
> solutions/exercise4.sh
```

CURC loadbalancer summary

- No mpi knowledge required
- Saves time by reducing scheduling overhead
- Runs on multiple nodes
- Input file can be created in your favorite language
- Non-standard (but it is on github)
  - https://github.com/ResearchComputing/lb

References

- Bash scripting
  - https://www.rc.colorado.edu/blog/reducejanuswaittimes
- GNU parallel
  - https://www.gnu.org/software/parallel/parallel_tutorial.html
  - https://www.gnu.org/software/parallel/man.html
  - O. Tange (2011): GNU Parallel - The Command-Line Power Tool, ;login: The USENIX Magazine, February 2011:42-47.
- Slurm arrays
  - https://slurm.schedmd.com/job_array.html
- CURC loadbalancer
  - https://github.com/ResearchComputing/lb
  - https://www.rc.colorado.edu/support/examples-and-tutorials/load-balancer.html

Questions?

- Email rc-help@colorado.edu
- Twitter: CUBoulderRC

- Link to survey on this topic: http://tinyurl.com/curc-survey16

- Slides: https://github.com/ResearchComputing/Parallelization_Workshop