Job Scheduling on Midway

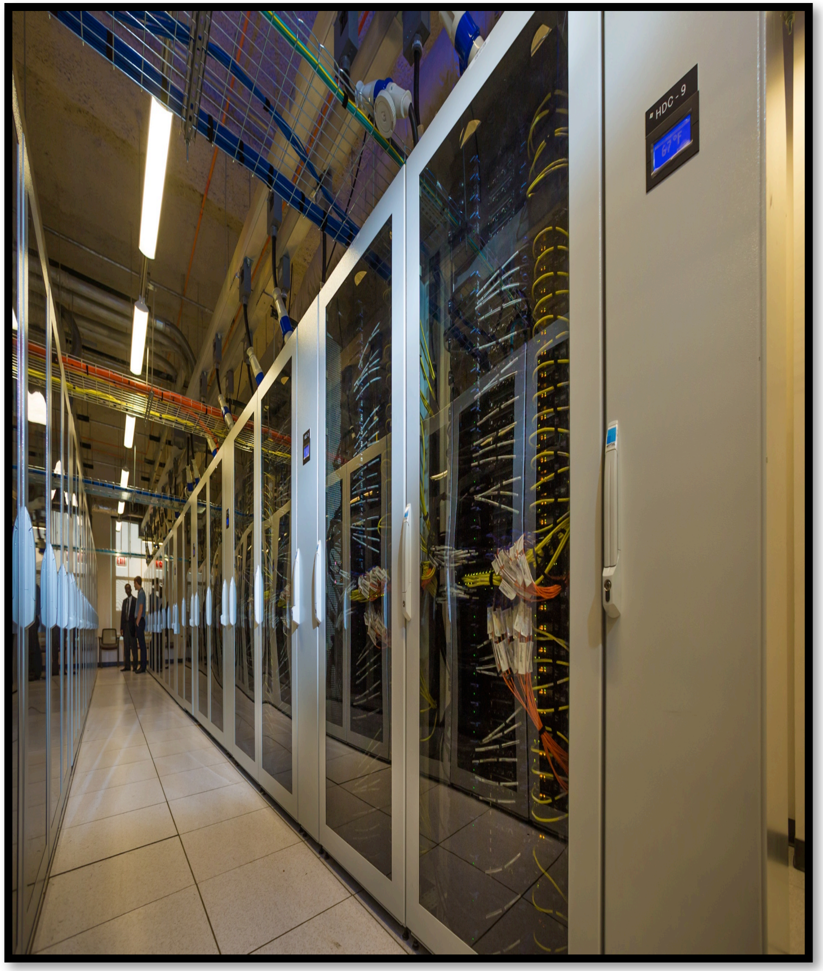
Hossein Pourreza

hpourreza@uchicago.edu

April 18, 2017

Midway

- UChicago's largest supercomputer managed by RCC
- A shared resource used by more than 2000 users
- Intended to be used non-interactively



Job, batch system, and scheduling

- A *job* is an executable code along with input data, parameters, environment variables, output files, and the description of required resources

- Jobs are defined using job scripts

- A batch system (SLURM on Midway) receives job scripts, puts them in a queue, and try to run them according to some rules

- The process of running jobs is called *scheduling*

Scheduling goals

- Maximize fairness
- Maximize throughput
- Minimize waiting time
- Maximize utilization

Before submitting a job

- Login to Midway1 or Midway2
  - Depending on where you eventually want to run your job
- Know your default account
  - `rcchelp user CNetID`
- Make sure you have enough service units
- Go to the folder from where you want to launch your job
- Write your job script

Different partitions on Midway

- A partition is a group of compute nodes sharing similar properties
  - Many partitions on Midway
  - Try the `rcchelp sinfo` command for the complete list

- Different partitions have different policies
  - Try the `rcchelp qos` command for the complete list

- Depending on your account type, some of partitions may not be available to run your job

Public partitions on Midway

Midway1	sandyb
	ivyb
	bigmem
	westmere
	gpu
	mic
	amd
Midway2	broadwl
	broadwl-lc
	bigmem2
	gpu2

Service Units

- Service Units (SUs) are the measure of computing resources

- In the simplest form, 1 SU is defined as using 1 core for 1 hour

- Each Principal Investigator (PI) requests for SUs to be used by her/his research group

- Use `rcchelp balance` and `rcchelp usage` commands to find out more your balance and usage

Storage types on Midway

- Home
  - A backed-up, private space usually used to keep your important files
  - Usually not a good place to start your jobs

- Project
  - A backed-up space usually shared between members of a research group
  - Jobs could be started from project unless they require heavy I/O

- Scratch
  - Non backed-up, private space tuned to run large I/O jobs
  - Accessible via the `SCRATCH` environment variable

Using software on Midway

- Midway uses the module software management tool
- You usually need to load a module before running your code
- The same set of modules should be specified in you job script
- Avoid loading modules in your login startup scripts

CPU, memory, and walltime

- Serial jobs cannot run faster by asking for more CPUs
  - Ask for as many CPUs as your program can utilize
- Know your job’s maximum amount of memory usage
- Know how long it will take for your program to finish
- Asking extra CPUs or memory will unnecessarily burn your SUs without getting your code run faster

Question 1

- Which of the following statements is correct?
  - My program reads gigabytes of data and writes few megabytes per second. I should start this job from home?
  - I login to Midway1 and then decide which partition to run my job on
  - I always ask 32GB of memory to make sure my job has enough memory to finish
  - I load required modules before run my code on Midway

Question 2

- True or False?
  - I submit my jobs in midnight to be ahead of other users
  - When there are many jobs on Midway, my program runs slower
  - The more CPU I request, the faster my job will run
  - The Midway scheduler will run my job as soon as possible respecting other jobs' priority

Before submitting a job

- Login to Midway1 or Midway2
  - Depending on where you eventually want to run your job

- Know your default account
  - rcchelp user CNetID

- Make sure you have enough service units

- Go to the folder from where you want to launch your job

- Write your job script

A sample job script

```
#!/bin/bash
#SBATCH --job-name=example_sbatch
#SBATCH --output=example_sbatch_%j.out
#SBATCH --error=example_sbatch_%j.err
#SBATCH --account=you_pi_account
#SBATCH --time=00:05:00
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --mail-user=your_email_address
#SBATCH --partition=broadwl
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=2000

module load python
cd your_working_directory
echo “job started at `date`”

python my_script.py

echo “job finished at `date`”
```

Submitting a job

- Once you have a job script (saved as sample_job.sbatch, for example), you could submit the job as:
  - `sbatch sample_job.sbatch`

- To see if your job has been accepted, run:
  - `squeue -u $USER`

Multi-threading vs message passing

- Multi-threading is a parallel programing mode utilizing available compute cores of one node
  - Use `--nodes=1` along with `--ntasks=xx` if you are running multi-threaded code
  - A multi-threaded code can NOT use cores from multiple nodes

- Message passing is a parallel programming mode utilizing compute cores from multiple nodes
  - Use `--ntasks=xx` if you running message passing code

Monitoring jobs

- While your job is running
  - `squeue –u $USER`
    - A useful variable in your login startup script
      - `export SQUEUE_FORMAT="%.8i %10P %.8j %10u %.15a %.3t %9r %.10l %.10L %.5D %.4C %N"`
  - `scontrol show job` *`job_id`*
  - To cancel your submitted job
    - `scancel` *`job_id`*
- After your job is finished
  - `sacct –j job_id`
    - A useful variable in your login startup script
      - `export SACCT_FORMAT="jobid,partition,user,account%16,alloccpus,node%20,elapsed,totalcpu,maxRSS,state"`

Interactive jobs

- It is sometimes necessary to test your code interactively before submitting your batch job

- To create an interactive session, use the sinteractive command:
  - `sinteractive -p sandyb --ntasks=1 --mem-per-cpu=2000 --time=00:30:00`

- Once you get a session, you could load modules (if applicable) and run your code interactively

GPU-enabled jobs

- Midway has some special purpose nodes dedicated to run jobs using Graphical Processing Units (GPUs)

- You program should be GPU enabled to take advantage of these nodes

- To request a GPU node with one GPU card, use:

```
#SBATCH    --partition=gpu
#SBATCH    --gres=gpu:1
```

  - The `--partition=gpu and --gres=gpu:1` tags can also be used with the `sinteractive` command

Job arrays

- The job array in SLURM is an efficient way to submit many similar jobs

```
#!/bin/bash
#SBATCH --job-name=arrayJob
#SBATCH --output=arrayJob_%A_%a.out
#SBATCH --error=arrayJob_%A_%a.err
#SBATCH --array=1-6
#SBATCH --time=01:00:00
#SBATCH --partition=sandyb
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=4000

module load python
# Print this sub-job's task ID
echo "My SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID

# Do some work based on the SLURM_ARRAY_TASK_ID
python my_script.py $SLURM_ARRAY_TASK_ID
```

Reducing the number of serial jobs

```
#!/bin/sh
#SBATCH --time=00:05:00
#SBATCH --ntasks=4

module load python
module load parallel

srun="srun --exclusive -N1 -n1"

parallel="parallel --delay .2 -j $SLURM_NTASKS --joblog
runtask.log --resume"

$parallel "$srun python my_script.py {1} > runtask.{1}" :::
{1..16}
```

Conclusion

- Midway is a world-class super computer accessible to all UChicago researchers

- To be able to run your code on Midway, you need to submit a job

- Selecting accurate parameters in your job submission script can reduce your wait time and SU usage

Useful links

- RCC official website
  - https://rcc.uchicago.edu
- Midway user documentation
  - https://rcc.uchicago.edu/docs/using-midway/index.html
- Official SLURM commands documentation:
  - https://slurm.schedmd.com/sbatch.html
  - https://slurm.schedmd.com/scontrol.html
  - https://slurm.schedmd.com/squeue.html
  - https://slurm.schedmd.com/sacct.html