Slurm in 5 min.

Robert Dyro
Stanford ASL

August 15, 2023

Outline

Introduction - Overview

- a cluster allows to run independent processes across many computers
- many computers usually means 2 to 20
- by all means run your computation on only 1 computer if you want!
- processes are independent, can communicate over network
- usage through shell through `ssh`
- a cluster is managed by a cluster software - Slurm
- computers in the cluster communicate via network (sockets)
- storage is shared, all computers see the same files
- sharing storage is managed by the shared storage software
- no `sudo`, install locally or load available packages

Introduction - Terminology

- *a node* - a computer on the network managed by Slurm [-N]
- *a task* - a single of the many processes you want to run [-n]
- *a CPU* - a single core on a computer (a physical core) [-c]
- *a job* - a set of independent processes to run
- *to submit* - to ask Slurm to run a job
- *to schedule* - what Slurm does to execute all your tasks
- *array job* - an alternative way of scheduling NOT covered here

Introduction - How it works

- you submit a job
  - your job consists of instruction how many and what processes to run
  - you specify strict requirements: nodes nb, memory per node, time limit
  - you specify other requirements and job hints
- your jobs waits in a queue until resources (nodes) are available
  - if it is rejected, it is rejected immediately
- your job is scheduled
  - you can get an email about it
  - you check on the status by examining a combined output file
- your job finishes or is cancelled (by you)

Outline

Important Commands

Slurm user commands (you're the user, as opposed to an admin)

- `sinfo` - query available computational resources
- `squeue` - check status of jobs (`squeue --user $USER`)
- `srun` - run a single process, low-level command
- `sbatch` - run a job, a job is a shell script (that calls `srun`)

`module` - allows to load packages, no `sudo apt` on a cluster

- `module avail` - available preinstalled packages
- `module list` - currently loaded packages
- `module load package` - load *package*
- `module unload package` - unload *package*

Example Job Script

run by issuing `sbatch my_job.sh`

```
#!/bin/bash
#SBATCH --output=out_%j.txt
#SBATCH --time=0-10:00
#SBATCH --mail-type=ALL
#SBATCH --ntasks=20
#SBATCH --cpus-per-task=2
module load julia # load needed packages
source environment # like a python virtualenv
echo ""; date; echo ""
srun -n1 -c2 python3 setup.py # 1 node, task and cpu
wait # barrier ########################################
for i in {0..100}; do # 101 times, more than ntasks is OK
  srun -n1 -c2 python3 exp.py $i & # notice &
done # each srun ends with & to run all in the background
wait # barrier ########################################
echo ""; date; echo ""
```

Scheduling

- when you connect, you're on the login node, don't run stuff there
- `srun --pty bash` - to run a computational node interactively
- `srun -p gpu --gpus 1 --pty bash` - a gpu node interactively
- `#SBATCH -p gpu` in the `my_job.sh` file to have GPU nodes
- `tail -n 100 out_43598734.txt` to check on a job

Outline

Sherlock - Stanford University Cluster

All info at: www.sherlock.stanford.edu

- `ssh $USER@login.sherlock.stanford.edu`
- https://login.sherlock.stanford.edu - web-based interface

Comments

- uses Slurm
- access via a linux shell without `sudo` privileges
- has 100s general computational nodes and ∼100 GPU nodes
- account is free, but requires Sherlock admin's and your PI's approval
- usage is free for research, requires publication acknowledgement
- can request all the nodes, but will spend time in queue
- simple job wait: 1 node - 10 s, 10 nodes - 5 min, 20 nodes - 1 hour

Sherlock Storage

Storage

- three types of storage - *long*, *short* and *very short* term
- storage is very generous (a lot of GB or TB)
- *short* and *very short* storage locations are vastly faster
- *short* storage locations have file expiration (90 days)
- *very short* storage location is cleared on job end

Available storage locations - these are bash variables

- *long* term storage
  - `$HOME` - personal long term storage
  - `$GROUP_HOME` - shared research group storage (your lab)
- *short* term storage
  - `$SCRATCH` - personal short term storage
  - `$GROUP_SCRATCH` - shared research group storage (your lab)
- *very short* term storage - very good performance
  - `$L_SCRATCH` - local to node
  - `$L_SCRATCH_JOB` - local to node & job

Sherlock Tips

Persistent login

Under `~/.ssh/config` put

```
Host login.sherlock.stanford.edu
    ControlMaster auto
    ControlPath ~/.ssh/%l%r@%h:%p
```

Multiple terminal windows at computational node

- query what nodes are yours `squeue --user $USER`
- issue `ssh sh02-01n43`

Installing things

- Sherlock does not give `sudo` access
- many installations do not *really* require `sudo`
- install and use stuff from `~/.local/lib` `~/.local/bin`

Copying files back and forth

- use `rsync`! excellent tutorial from Digital Ocean here