Workflow Management Software on HPCC

CMSE 890-602

Why use High Performance Computing Clusters?

- Access to lots of computational resources
    - CPUs
    - GPUs
    - Memory
    - Storage
- High speed network connections
- Run workflow steps in parallel
- Accelerate individual workflow steps
- Much cheaper than commercial cloud resources
- National access via…ACCESS
    - https://access-ci.org

Environment

- Almost always a Linux distribution
- Often includes pre-installed software as modules
    - Commonly using “environment modules” or “lmod” tools
- Typical structure consists of
    - Development nodes for testing
    - Compute nodes for job execution
- Nodes may have a range of hardware available
- Limited storage
- No root access
    - Can’t build containers
    - Can’t perform certain tasks

Job scheduling

- HPCCs require management of user computing
- Job schedulers control:
  - Resource usage (CPU, GPU, memory etc)
  - Time usage
  - Priority access
- Users submit jobs to the scheduler via command line or scripts
- Common schedulers: SLURM, Torque

Requesting Resources

- Use development nodes where possible to estimate memory usage and time
- Run test jobs with smaller data sets or simpler configurations to gauge performance
- Analyse results with tools:
  - `top, htop` to investigate memory and CPU usage
  - `nvidia-smi` for GPU usage
  - `seff` to get post-job information
  - At ICER we also have `reportseff` to get post-job info about many jobs at once
- If in doubt, request more than you think you need and reduce in later jobs

High throughput computing

- Subtly different from regular HPCC use
- Running many thousands of small jobs at once, instead of a few jobs with many resources each
- Uses distributed resources across many HPCCs
- Jobs are often “pre-emptible” i.e. local users can kick you off
- Open Science Grid
    - https://osg-htc.org/

Workflow manager interface

- Automatic creation of jobs
- Runs best on a development node (uses few resources)
- Allocate per-process/rule resources
- Both managers we have discussed (SnakeMake and Nextflow) use Executors for scheduling
- SnakeMake has somewhat better SLURM integration
    - Including HPCC profiles, default resource allocation
    - https://snakemake.github.io/snakemake-plugin-catalog/plugins/executor/slurm.html
- Nextflow has better commercial cloud integration
    - AWS support, Kubernetes support
    - https://www.nextflow.io/docs/latest/executor.html#slurm

Snakemake syntax

- Install slurm executor plugin into snakemake environment
- `--executor slurm`
- Set default resource request `--default-resources runtime=10 mem_mb=512 cpus_per_task=8 --jobs 10`
- Customize resource request in each rule with `resources` directive
- Load environment modules with the `envmodules` directive
  - Activate them using `--software-deployment-method env-modules` replacing `conda`
- Create a profile to store the default resource request and executor type

Nextflow syntax

- Set `process.executor = 'slurm'` in the Nextflow configuration file
- Use process directives to set the resource request,
  - `cpus`
  - `memory`
  - `time`
  - `clusterOptions` to add arbitrary command options

In-class assignment

Go to https://msu-cmse-courses.github.io/CMSE_890-602_snakemake/

Scroll to section 3.16

We will go through this together!

Homework

Work on your semester project!