Snakemake Hackathon

Intro

⚙ Prerequisites

prerequisites

HPC refresher

Recap of the most important concepts of HPC that need to be understood to run Snakemake on clusters.

Login vs compute node

Resource estimation and requests

Modules

Premade

Own modules

Many small jobs

Heavy I/O

..-

Snakemake introduction

Why workflow tools?

Example usecase from GUI -> multiple scripts run manually -> bash script to run all

> Issues with bashscripts: running only parts (commenting/uncommenting), running only for some files

Workflow tools

reproducible description of workflow, “smart”

Snakemake

Why snakemake over other tools?

Snakefile

Rules

Dependencies of steps

Wildcards

Python code in Snakefiles

Parallelism

Portability

Running Snakemake worklfow on CSC supercomputers

Snakemake workflow is one of the popular scientific workflows in the bioinformatics community. The workflow manager enables scalable and reproducible scientific pipelines by chaining a series of rules in a fully-specified software environment. Snakemake software is available as a module in Puhti supercomputing environment.

Use containers as runtime environment for portable workflows

One can use Singularity/Apptainer container as an alternative to native or Tykky container-based installations for better portability and reproducibility. If you don’t have a ready-made container image for your needs, you can build a Singularity/Apptainer image on Puhti using –**fakeroot** option.

For the purpose of this tutorial a pre-built container image which has all the software stack needed is provided to run snakemake workflow at scale.

Working with modules: Use HyperQueue executor for high-throughput workflows

If a workflow manager is using `sbatch` for each process execution (i.e., a rule), and you have many short processes, it’s advisable to switch to HyperQueue to improve throughput and decrease load on the system batch scheduler.

You can load HyperQueue and Snakemake modules on Puhti as below:

```
module load hyperqueue/0.16.0
module load snakemake/8.4.6
```

‼ In case you are planning to use Snakemake workflows on LUMI supercomputer, you can use module installations, done by CSC staff on LUMI. You can load HyperQueue and Snakemake modules as below:

```
module use /appl/local/csc/modulefiles/
module load hyperqueue/0.18.0
module load snakemake/8.4.6
```

One can use HyperQueue executor settings depending on the Snakemake version as below:

```
# snakemake version 7.xx.x
snakemake --cluster "hq submit  ..."
# snakemake version 8.x.x.x
snakemake --executor cluster-generic --cluster-generic-submit-cmd "hq submit ..."
```

Snakemake workflow execution on HPC cluster

Create and enter a suitable scratch directory on Puhti (replace with your CSC project, e.g. project_2001234):

```
mkdir -p /scratch/<project>/$USER/snakemake-ht
cd /scratch/<project>/$USER/snakemake-ht
```

Download tutorial materials (scripts and data), which have been adapted from the official Snakemake documentation, from CSC Allas object storage as below:

```
wget https://a3s.fi/snakemake_scale/snakemake_scaling.tar.gz
tar -xavf snakemake_scaling.tar.gz
```

The downloaded material includes scripts and data to run snakemake pipeline. You can use `snakemake_hq_puhti.sh` whose content is posted below:

```
#!/bin/bash
#SBATCH --job-name=snakemake
#SBATCH --account=<project>  # replace <project> with your CSC project, e.g.
project_2001234
#SBATCH --partition=small
#SBATCH --time=00:10:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=40
#SBATCH --mem-per-cpu=2G

module load hyperqueue/0.16.0
module load snakemake/8.4.6

# Specify a location for the HyperQueue server
export HQ_SERVER_DIR=${PWD}/hq-server-${SLURM_JOB_ID}
mkdir -p "${HQ_SERVER_DIR}"

# Start the server in the background (&) and wait until it has started
hq server start &
until hq job list &>/dev/null ; do sleep 1 ; done

# Start the workers in the background and wait for them to start
srun --exact --cpu-bind=none --mpi=none hq worker start --cpus=${SLURM_CPUS_PER_TASK} &

hq worker wait "${SLURM_NTASKS}"

snakemake -s Snakefile --jobs 1 --use-singularity --executor cluster-generic --cluster-
generic-submit-cmd "hq submit --cpus 5"

# for snakemake versions 7.x.xx, use command: snakemake -s Snakefile --jobs 1 --use-
singularity --cluster "hq submit --cpus 2"

# Wait for all jobs to finish, then shut down the workers and server
hq job wait all
hq worker stop all
hq server stop
```

How do you parallelise snakemake workflow jobs?

The default script provided above is not optimised to run in high-throughput way as snakemake workflow manager just submits one job at a time to hyperqueue scheduler. You can parallelise workflow tasks (i.e., rules in snakemake) by submitting more jobs from *snakemake* command as below:

```
snakemake -s Snakefile --jobs 8 --use-singularity --executor cluster-generic --cluster-
generic-submit-cmd "hq submit --cpus 5"
```

You can correct above modification in the batch script (and use your own project number in sbatch directives) before submitting the Snakemake workflow job to the HPC cluster as below:

```
sbatch snakemake_hq_puhti.sh
```

One can also use more than one node to achieve even more high-throughput as HyperQueue can make use of multi-node resource allocations.

💬 Please note that just by increasing the number jobs will not alone automatically run all those jobs. *Jobs* parameter from *snakemake* is just a maximum limit for concurrent jobs. Jobs will be eventually run when resources are available. In our case we submitted 8 parallel jobs, each taking 5 CPUs as we reserved 40 CPUs in batch script. In practice it is a good idea to dedicate few CPUs for workflow manager itself.

Follow the progress of jobs

You can already check the progress of your job by simply observing the current folder where you can see lot of new task-specific folders are being created. However, there are formal ways to check the progress of your jobs as shown below:

1. Monitor the status of submitted job

```
squeue -j <slurmjobid>
# or
squeue --me
# or
squeue -u $USER
```

2. Monitor the progress of the individual sub-tasks using HyperQueue sub-commands

```
module load hyperqueue
export HQ_SERVER_DIR=$PWD/hq-server-<slurmjobid>
hq worker list
hq job list
hq job info <hqjobid>
hq job progress <hqjobid>
hq task list <hqjobid>
hq task info <hqjobid> <hqtaskid>
```

Managing file I/O

HyperQueue creates task-specific folders (i.e., job-`<n>`) in the same directory from where you have submitted batch script. These are sometimes useful for debugging. However if your code is working fine, the creation of such large number folders may be annoying besides causing some overhead to parallel file systems like Lustre. You can prevent creating such task-specific folders by setting `stdout` and `stderr` flags to `none` as shown below:

```
snakemake -s Snakefile -j 24 --use-singularity --executor cluster-generic --cluster-
generic-submit-cmd "hq submit --stdout=none --stderr=none --cpus 5 "
```

More Information

- CSC documentation on Snakemake
- Snakemake official documentation

...

Quick Reference

Instructor's guide

Why we teach this lesson

Intended learning outcomes

Timing

Preparing exercises

e.g. what to do the day before to set up common repositories.

Other practical aspects

Interesting questions you might get

Typical pitfalls

Who is the course for?

About the course

See also

Credits