Bioweek: Using Modern HPC Environment Efficiently

03.02.2020

Running Jobs in CSC Servers

Exercise 1: Running a simple batch job in Puhti

We will run a small alignment using BWA:

https://docs.csc.fi/#apps/bwa/

Log into puhti**.csc.fi**

host: **puhti.csc.fi**

login as: **trainingXXX (replace XXX with your account number)**

If you did not create your own folder in /scratch in the morning session, do it now:

```
cd /scratch/project_2002389
mkdir trainingXXX
```
(replace XXX with your account number)

```
cd trainingXXX
```

Copy a set of test files to your current folder:

```
tar xf /scratch/project_2002389/ex1.tar.gz
cd ex1
```

Many of the bioinformatics tools in Puhti are loaded with command: `module load biokit.` There are, however, several tools that have their own set up commands, or which don't need a set up command. You can check the software specific information from the bioscience program list:

http://research.csc.fi/bioscience-programs

https://docs.csc.fi/#apps/

Among the files you copied, is a template batch job script called `bwa_run.sh` Open it with a text editor. You can use for example *nano or emacs* editors:

```
nano bwa_run.sh
emacs bwa_run.sh
```

Edit batch job script so that:

- It reserves **4** cores in **one** node**, 8 GB** memory and **30 min** time
- runs in the **small** partition
- captures **stdout** and **stderr** separately
- make sure billing project is specified

  You can use command
  `csc-workspaces` (CSC specific)
  or
  `id` (should work on all systems)

  to check your available projects. For the course we use **project_2002389**

Submit the job with command:

```
sbatch bwa_run.sh
```

After submission you can follow the status of your job with commands:

```
squeue -u trainingXXX
squeue -l -u trainingXXX
```

Exercise 2: Using Allas in a batch job

In the case of batch jobs, as execution of a batch job can take several days and some cases it may take more than eight hours before the job even starts. In these cases you should open Allas connection with command:

```
allas-conf –k
```

This will save your password in an environment variable $OS_PASSWORD

Add the following command to the batch job script to renew the connection:

```
source /appl/opt/allas-cli-utils/allas_conf -f -k $OS_PROJECT_NAME
```

Using what we learned in the morning, modify bwa_run.sh so, that after completing the alignment, it will save the resulting SAM file to Allas.

Exercise 3: Looking at used resources

After the jobs in Exercises 1 -2 have finished, you can check what resources it used with

```
seff <jobid>
```

(If your jobs have not finished, you can use jobid 451423 instead.)

Also look at jobid 451193 (an array job)

Were the resource requests reasonable?

Another tool to look at resource usage is sacct

To see details of a single job you can use

```
sacct –l –j <jobid>
```

sacct lists a rather long list of values, many of which are of interest mainly to sysadmins. To get a more manageable output you can use the –o option to define which fields you want to see.

```
sacct -o jobid,jobname,maxrss,maxvmsize,state,elapsed -j <jobid>
```

Here we have elected to show jobid (JobID), jobname (JobName), maximum used memory (MaxRSS), maximum used virtual memory (MaxVMSize), state of the job (State) and elapsed time (Elapsed). These can be helpful when we decide on the resource allocation parameters for our next similar job.

Compare results from sacct and seff. Which datafield from sacct you need to replicate the info from seff?

Why might sacct be preferable for array jobs?

For details on the different available fields and other options see

```
man sacct
```

More information about running batch jobs in Puhti:

https://docs.csc.fi/#computing/running/getting-started/

Exercise 4: Working with Singularity containers

Get the template batch job scripts and data for this exercise:

```
tar xf /scratch/project_2002389/ex4.tar.gz
cd ex4
```

In most cases you would probably prefer to do the container building in an interactive session, but in this case we will do it as a batch job as getting an interactive session might take some time.

First create some folders:

```
mkdir singularity
mkdir singularity/tmp
mkdir singularity/cache
```

Edit **singularity-build.sh** to to replace all instances of “trainingXXX” with correct folder names.

Compare the `singularity build` command to the `docker pull` command on this page:

https://bioconda.github.io/recipes/fastx_toolkit/README.html

Submit the job:

```
sbatch singularity-build.sh
```

You should end up with image file fastx.sif

Edit **singularity-run.sh** as above. Submit the job:

```
sbatch singularity-run.sh
```

We also have a native installation of FASTX-toolkit available in Puhti. As an extra task you could try running the same analysis using that. Replace the singularity command with:

```
module load fastx-toolkit
fastx_quality_stats -i test_1.fastq -o test_1.fq.stats.2
```

Compare the commands. Use seff to compare used resources.

If your jobs are still queueing, you can use job-ids **1075941** (run with singularity) and **1075981** (run with native fastx-toolkit).