Pronghorn Homework Module

All personal directories are located: /data/gpfs/assoc/biol\_bids-2/

Setting up

Submission script is located:

/data/gpfs/assoc/biol\_bids-2/atram.slurm.sh

Copy this file to your personal directory.

In your personal directory you will need to create two directories:

mkdir temp

mkdir atram\_out

Below is an example of the submission script and the bold sections within square brackets should be changed accordingly.

*atram.slurm.sh*

#!/usr/bin/bash -l

#SBATCH --job-name=**[give your job a descriptive name]**

#SBATCH --account=cpu-s5-biol\_bids-2

#SBATCH --partition=cpu-core-0

#SBATCH --nodes=1

#SBATCH --ntasks=1

#SBATCH --cpus-per-task=1

#SBATCH --time=14-00:00

#SBATCH --mem-per-cpu=3500M

#SBATCH --output=hostname\_%j.out

#SBATCH --error=hostname\_%j.err

#SBATCH --mail-type=ALL

#SBATCH --mail-user=**[your email]**

singularity exec /apps/atram/aTRAM.sif python3 /aTRAM/atram.py -b /data/gpfs/assoc/biol\_bids-2/Course\_content/atram\_files/AlspHabad/AlspHabad -t /data/gpfs/assoc/biol\_bids-2/**[netID]**/temp -Q /data/gpfs/assoc/biol\_bids-2/Course\_content/atram\_files/query\_files/**[Number of genes]**\_proteins.fasta -a trinity -o /data/gpfs/assoc/biol\_bids-2/**[netID]**/atram\_out/

Target files

All target files are found here: /data/gpfs/assoc/biol\_bids-2/Course\_content/atram\_files/query\_files/

Each file begins with the number of genes it contains. In class you will assemble 2 genes. For your homework you will assemble 20 genes.

Processors

You will need to select the number of processors you want to use per job. You can change the number of tasks and CPUs per task in the SBATCH options in the submission script. Pronghorn has 32 CPUs per node. Pronghorn will allocate the resources you request to aTRAM as it runs. You can start with a single task and CPU per task, measure the gene assembly rate, and change the computational resources from there.

Submitting

Submit the job using:

sbatch atram.slurm.sh

Queue

To check on the status of your job, use squeue in the command line. This will pull up a list of all of the jobs currently running or waiting in the queue. You will see your job by the job name and your user name. If your job is waiting in the queue, the time will say 0:00. Once it starts running the timer will start. You can check on your jobs only with squeue -u <USERNAME>

**For your homework you will assemble 20 genes.**

**Answer the following questions:**

**1. What is the purpose of High Performance Computing (HPC) and when might you use an HPC cluster?**

High performance computing (HPC) is the application of cutting-edge computing methods and tools to resolve challenging computational issues that need a substantial amount of computer power. To solve issues that are beyond the scope of conventional computing systems, HPC aims to offer exceptionally high levels of processing power, memory capacity, and storage capacities.

HPC clusters are typically used in scientific and engineering research, where complex simulations, data analysis, and modeling are required. These applications often involve large datasets and require complex algorithms that can take a long time to complete on a single computer. Genome sequencing and analysis, climate modeling and prediction, and astrophysics simulations are some examples of applications that might require HPC clusters.

**2. What is a slurm script? Print the path of your slurm script here (including file name).**

A Slurm script is a batch script used for submitting jobs to a Slurm workload manager in a High Performance Computing (HPC) cluster. The script specifies the resources required for the job, such as the number of CPUs, amount of memory, and duration of the job, as well as the commands to be executed.

/data/gpfs/assoc/biol\_bids-2/jeshurun/atram.slurm.sh

#!/usr/bin/bash -l

#SBATCH --job-name=jeshurun

#SBATCH --account=cpu-s5-biol\_bids-2

#SBATCH --partition=cpu-core-0

#SBATCH --nodes=1

#SBATCH --ntasks=1

#SBATCH --cpus-per-task=1

#SBATCH --time=14-00:00

#SBATCH --mem-per-cpu=3500M

#SBATCH --output=hostname\_%j.out

#SBATCH --error=hostname\_%j.err

#SBATCH --mail-type=ALL

#SBATCH --mail-user=jbiney@unr.edu

singularity exec /apps/atram/aTRAM.sif python3 /aTRAM/atram.py -b /data/gpfs/assoc/biol\_bids-2/Course\_content/atram\_files/AlspHabad/AlspHabad -t /data/gpfs/assoc/biol\_bids-2/jeshurun/temp -Q /data/gpfs/assoc/biol\_bids-2/Course\_content/atram\_files/query\_files/20\_proteins.fasta -a trinity -o /data/gpfs/assoc/biol\_bids-2/jeshurun/atram\_out/

**3. What is the path to your gene assemblies on pronghorn (i.e., output files).**

/data/gpfs/assoc/biol\_bids-2/jeshurun/atram\_out

**4. How many CPUs per task did you ask for?**

1 CPU for class work

20 CPUS for homework

**5. How many genes assembled?**

19 genes assembled

**6. Did all genes assemble? Why or why not?**

Not all genes assembled

1) Some genes were found to be missing one gene section.

2) While others exhibited high sequence divergence compared to the reference.

3) Some had a small exon at one end of the gene.

As the original query sequence only included exons and aTRAM assemblies include introns, genes with a small exon at one or both ends may not exhibit a high BLAST match of these small exons to the original gene sequence. These findings indicate that although the gene may not be identified as complete at the end of the iterations, the entire gene may still be assembled. Furthermore, our experience suggests that adding more iterations as the assembled contig grows can lead to the complete assembly of the target locus. This is especially true for very large genes, where more iterations may be required to achieve a full gene assembly.

**7. How long did it take for them to assemble? What processes might have slowed this down?**

Slurm Job\_id=4515212 Name=jeshurun Ended, Run time 01:59:34, COMPLETED, ExitCode 0

*Hardware architecture*: The performance of HPC clusters depends largely on the underlying hardware architecture.

*Software stack*: The software stack used to run applications on the HPC cluster can also impact performance.

*Workload characteristics*: The performance of an HPC cluster can vary depending on the type of application being run, the input data size, the communication patterns within the application, and the level of parallelism required can all impact the performance of the cluster. Some applications may be more I/O-bound or memory-bound than others, while some may require a high degree of inter-node communication.

*System tuning*: HPC clusters require careful tuning to ensure optimal performance. This includes optimizing the network configuration, tuning the storage subsystem, and adjusting system-level parameters such as memory allocation, process scheduling, and interrupt handling.

*User behavior*: The behavior of users can impact the performance of an HPC cluster.