Hello Awesome PI,

The worm data that you provided was significantly smaller and took less time to download. I was able to download the worm data from Ensembl by making a small adjustment to the getGiabData.sh script. I changed the file that you already had in there (Human genome) to the new link that you provided (Worm genome). I even added a way to time how long the file takes to download. The worm data took about 10 seconds to download, which is a very short time. Since there were files on the HPCC, I added a method for the script to copy those data sets into the directory you are working in. Take note, that you should run the getGiabData.sh on the directory that you will you want to work in.

I then moved onto trying the BWA index script on the file downloaded from Ensembl. I noticed that you added your own path to get to the Ensembl file but I modified it so the script will change to the directory that the file is in. That means that this will work for anyone if the file can be found. In order to load the BWA module, a GNU and OpenMPI module had to be loaded as well. I added these commands in the script to get the modules to load correctly. Since I changed the time and memory for the job, I also added a command to show you how much time and memory was actually used. I even added a time command to time how long the index runs. For the worm data, the index took about one minute and thirty seconds and took 147396K of memory. I attempted to run the index script on the Human genome file, but that job ran for too long for me to add it to the timing study.

The first things I changed about the alignment script were the changing of the directory and the module loads, like with the index script. The getGiabData.sh should move all of your data sets into one directory so make sure that they stay together or else this script will not work. Another add to the script was the timing of the alignment and that took about on average 23 minutes for the Worm data. Now to me, this is quite a long time, so I modified the file to run on multiple CPUs. This was so that the alignment would run faster. I ran it multiple time with a different number of CPUs each time. The main change to the script was requesting more nodes and two for loops outside of the srun command. The first for loop will change the number of nodes used, I used 2, 4, 8, and 16. The second loop nested in the first one will run the command 3 times with each number of CPUs to get an average of the time for alignment. It looks like the code run best with 4 CPUS, anything more than that took a decent amount of time.

I figured I could make the work easier writing a “getexample” of this type of problem. I created a directory called answer and it contains all of the scripts you proved with my edits. I even wrote a README file to get the steps of this example into a single workflow. By getting this directory and its contents, you can run all of these files and run them one by one or running the README file. The README file can be ran by doing ./README in the answer directory. Also, this example runs in the most optimal way which is on 4 CPUs.

This should be all that you have asked for, hope I helped you with your problem. I even added my timing study for you for when I tested the parallel code.

Sincerely,

Awesome Programmer (TJ Nguyen)