2017

Use of HPC to Annotate an Entire Genome

BME 5320 BIOINFORMATICS TECHNIQUES

WADDAH MOGHRAM

FINAL PROJECT

DUE FRIDAY, DECEMBER 15, 2017

Introduction

This project uses the High-Performance Computing (HPC) at the University of Iowa. More specifically, the Neon cluster has been designated for the use of this project. The project aims to annotate the protein-coding gene sequences of a given organism against a standard database in order to identify their function. In so doing, we will eliminate any self-hits that might be already part of the database. The goal of this project is to create a program that can be repurposed to identify any sequences that might be generated in a lab by matching them with close sequences that are already known. Also, this program should have the capability to be scaled up to hundreds of thousands of sequences if so desired.

My program is composed of the following subprograms: `HPC_Main.sh`, `ProteinBreakup.pl`, `HPC_BLAST.job`, `HPC_Annotate.job`, and `ProteinFilterHits.pl`. Once, `HPC_Main.sh` is called, all the other codes will be executed automatically subsequently.

Choice of Organism, Alignment Method of Choice

In this project, *Mycoplasma genitalium (G37)* – hereby referred to as MGG – is chosen as the organism of choice to model simple free-living organisms – as opposed to viruses or obligate organisms. The number of expressed proteins of MGG, according to the latest data, are 515[1], which is a reasonable starting point for this project. This is in the hopes of scaling up this project to annotate genes in humans, which are on the order of 20,000 protein-coding genes[2].

At first, basic local alignment search tools (BLAST) will be the method of choice at first to evaluate the degree of identity and/or similarity between two sequences. While it is possible to use BLAST for nucleotide sequences, that search is not only more time consuming, but it is also less useful for multiple reasons, not the least of which is that we are interested in the protein-coding genes. Therefore, non-coding sequences in genes will add more complexity. BLASTp program is a good starting point to annotate these genes. Subsequent improvements will include using position-specific iterative BLAST (PSI-BLAST), which will give more weight to highly conserved domains.

In the starting bash script `HPC_Main.sh`, the MGM protein sequences are downloaded into the head node using the command `wget` directly from NCBI genome FTP site and then are uncompressed. Then, the numbers of sequences are evaluated by the count of "`>`" mark using a bash script. If the number is complete (i.e., 515 sequences), then a confirmation message is echoed and the program proceeds. Otherwise, a warning is issued and the program is stopped.

Dividing Protein Sequence into Jobs

Next the query protein sequences are divided into text files that contain an equal number of sequences using a Perl script `ProteinBreakup.pl` – except for the last file that contains what is left. The default number of sequences is 10. However, this code can be used to split it into any given number of sequences by passing it as the second argument when executing the code, while the first argument being the name of the files to be divided.

The choice of sequences per batch will be a judgement call that depends on the time required to perform local alignment for a single protein sequence, the availability of slots, and the time required until the next task is processed and so forth. For purposes of this project, 10 sequences per batch seem about a good

[1] https://www.ncbi.nlm.nih.gov/genome/474?genome_assembly_id=300158
[2] https://www.edinformatics.com/math_science/human_genome.htm

starting point, thereby yielding about 52 batch files, each starting with the name `ProteinInput,` which are moved to folder `Protein_Input`. The file sequence generated at first are padded with whitespace. However, they are removed with the `tr //` function. This is done to take advantage of the `$SGE_TASK_ID` environmental variable used when creating an array job in the next step. This task ID number increases sequentially as a list.

Method of Parallelization with `qsub` and `blastp`

The next step in `HPC_Main.sh` is to calculate the number of `qsub` jobs that need to be submitted to HPC. The main code that will submit `HPC_BLAST.job` to `HPC`, which will run `blastp` code on each `ProteinInput` query file. This can be done either by creating as many job files as there are input query files. However, the easiest way is by submitting the jobs as an array using the `–t` flag, followed by array `1:JobCount`. `JobCount` is determined by calculating the number of files that begin with `"ProteinInput."` Inside `HPC_BLAST.job,` the number of task in the array is represented by `$SGE_TASK_ID`.

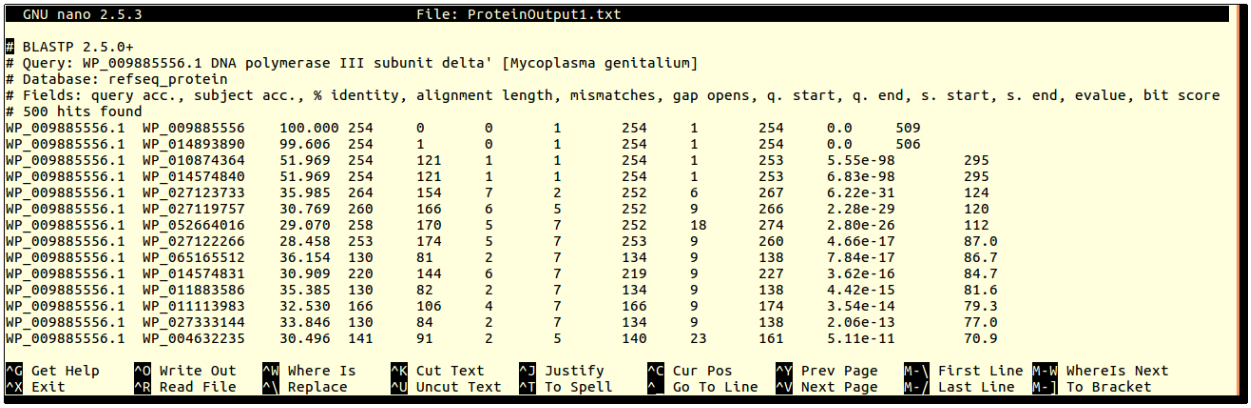
The `blastp` database used is `refseq_protein`, which comprises of "comprehensive, integrated, non-redundant, annotated sequences" of proteins. Other key parameters are the output format, `–outfmt 7`, or tabular format with the header is selected. This will make it easier to extract relevant information and annotate them. Another key parameter is `-num_threads`, which will be set to 4. Although increasing the number of threads will not have a major effect as increasing the number of slots used.

Also, in order to speed up the process, multiple queues were indicated in `qsub` flag `-q UI,DK,COE` so that any slots that are available will be used from any of these queues. And this seems to push about 14 jobs at a time usually (4 jobs on DK queue and 10 jobs on UI queue). It took about 75 minutes to perform BLAST and annotation. This seems to be consistent with about 10-20 minutes per BLAST job of 10 sequences on average, performed at 14 nodes. If these jobs were to be performed serially, it would take about 75 minutes/Job * 14 Jobs = 1050 minutes or 17.5 hours. That is with assuming 100% efficiency of 14 jobs. Annotating a human genome using this method can take at least 200 times more than that (assuming 100,000 coding proteins in humans compared to 515 proteins for MGM), or 3500 hours or 145 days. More cores and computing resources will be needed to annotate human genomes for certain!

Filtering of hits, Annotation of Results with `blastdbcmd`

After running a few trial runs, it was determined that it will take less than five minutes to annotate about the chosen 500 sequences. As a result, it is more worthwhile in this case to carry this step out on a single slot instead of parallel slots as before. However, as the size of the data increases, a better approach will be to parallelize this task as well for a sizeable chunk of blastp subject results. A reasonable size for annotation would be the results of 1000-1500 blastp sequences, which would take about 5-15 minutes each.

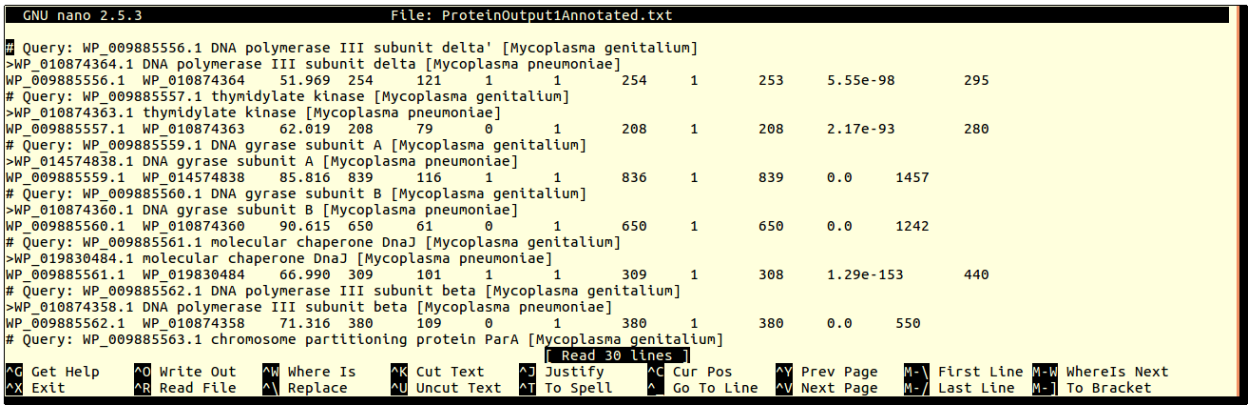
The Annotation code, `HPC_Annotate.job,` is submitted via `qsub` from inside `HPC_Main.sh`. However, it is placed on hold until all `HPC_BLAST.job` tasks are complete by using the flag `-hold_jid`. This parameter is extracted from the first qsub lines by a one-line bash script and passed on as variable `hold_JIB`. The output of each BLAST job is saved in text files starting with `ProteinOutput`, which are moved to folder `Protein_Output`.



`HPC_Annotate.job` will determine the number of `JobCount` files using a bash script again since that variable is lost between jobs. After that it will run `ProteinFilterHits.pl`, which is a custom-written Perl program to any given output file. `ProteinFilterHits.pl` is called continuously to loop through all the `ProteinOutput**.txt` files. In the Perl program, the files are read line-by-line, Then, lines are either classified as sequence header or hit results. The hit results are checked for E-value to see if it is less than the desired threshold value – chosen to be 1, which is pure chance although it can be set to a stricter criterion. From within Perl, `blastdbcmd` is invoked using `qx//`, as opposed to `exec()` or `system()`, since `system()` also outputs an "exit return status", which confuses things, while `exec()` does not return the `STDOUT`.

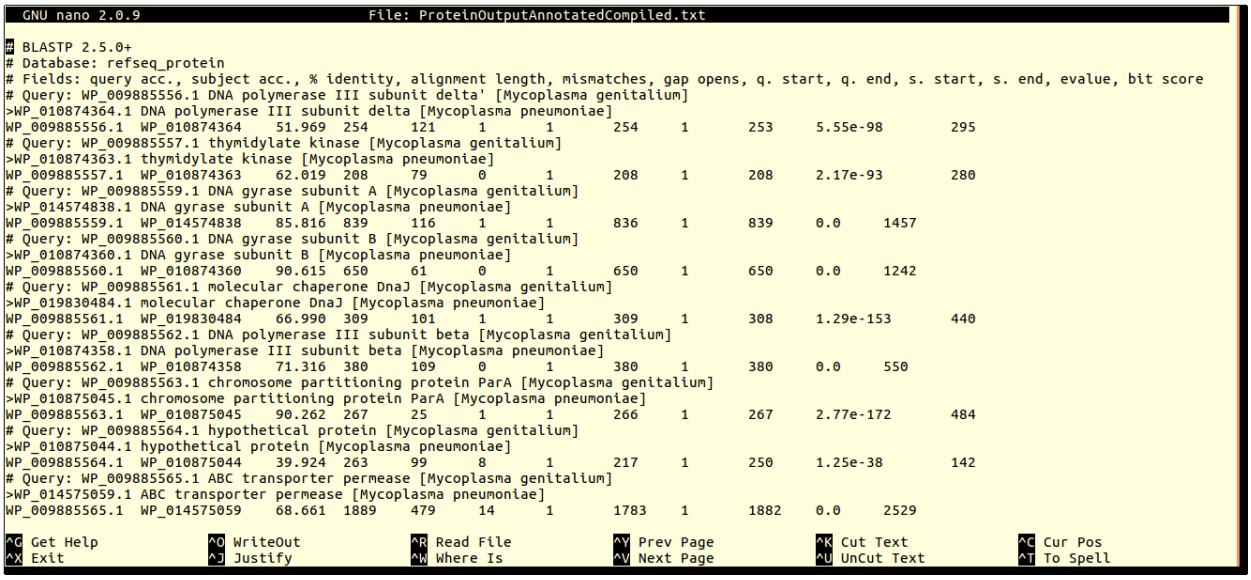
The annotation is checked to see if it is for the same organism or not (Note: Uncomment print "same organism" line to see how this works). Also, the length of the query sequence is called by using `blastdbcmd` followed by some Perl code to extract that information. This information is needed in order to calculate the coverage percentage. The subject percentage coverage cutoff chosen is 50% of the query sequence, but it can be modified in the code. If the organism is the same, then the next hit is searched. Once a hit that is not for the same organism is encountered, the annotation command is not executed. However, the program continues reading lines. (NOTE: This is not exactly very efficient code, but it is the simplest). When the next query sequence is encountered, a new search is started.

The output of `ProteinFilterHits.pl` will be append the word `Annotated` to its respective `ProteinOutput` file, which are moved to folder `Protein_Output`. Each annotated file will list the queries (starting with # sign), followed by the subject, followed by top annotation subject obtained by `blastdbcmd` (starting with > sign), and then the results of the top hit extracted from `blastp` results for each query protein in tabular format.

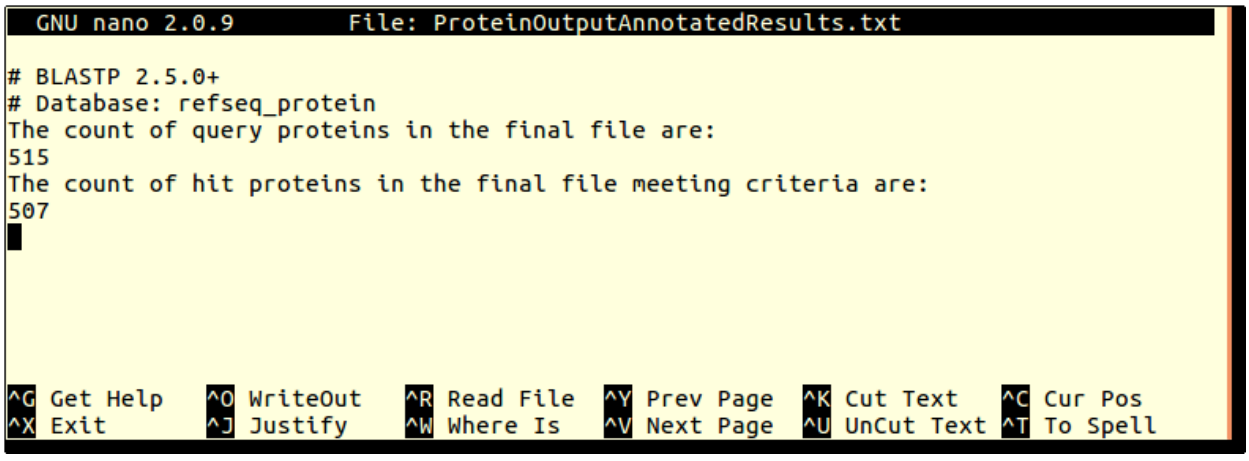


Compiling of Annotated Results and Addition of Headers

Once all the files are annotated, the final output file `ProteinOutputAnnotatedCompiled.txt` under `Protein_Output` folder is generated, by appending all `ProteinOutput**Annotated.txt` end-to-end. Next, a header is created at the beginning by copying lines 1, 3 and 4 of the standard `BLASTp` results in output format 7. These header lines are the version of BLAST program, the database used, and a header for the results columns.



One final check of the compiled file to ensure that all the query proteins are present. These results are output to `ProteinOutputAnnotatedResults.txt`, which is also under folder `Protein_Output`. Moreover, the number of hit queries that meet our criteria are count. Out of 515 query proteins, only 507 protein sequence hits were found. The others that were not found were either from the same organism in the database – which we do not want, or had E-values that were above the desired cut-off value. I also included a header indicated the BLASTP edition, and the database used.



Possible Room for Improvement

1. Provide a check for maximum number of cores. Select that number if it has the least desire number of slots instead of having to wait for the entire cluster.
2. Provide a check against any jobs that might have been killed. In my case, this was not necessary for the clusters I am working in. However, if calculations are carried out on `all.q` node. Such a check is done by check the size of the file ending in `.e<JobID>`. If this size of such a file is greater than zero, then read the corresponding file ending in `.o<JOBID>`, which contains the name of the `ProteinInput**.txt` file that was used for the query.
3. Parallelize Annotation job as mentioned previously.