March 14th Milestone Report
Sunny Nahar

Major Changes:

No major changes, project moving as expected.

What You Have Accomplished Since Your Last Meeting:

I started work on parallelizing the infrastructure. The first step is loading and building the hashtree (hashtable of trees) representation of the genome from data stored on disk. To facilitate parallelism (and slightly easier coding), I split the data into numerous pieces, which threads can load and operate on independently. There is roughly 80GB of data split into 256 files. When loaded into memory, this is about 350GB.

Each file is loaded and contains the data for a few trees. These trees are built in parallel. Disk access is sequential, so each thread must wait for its turn to read the file. I immediately load the file into a buffer in memory to keep the disk lock for as short of a time as possible. This method has some disk thrashing (but not too much), as files are not accessed in sequential order. I will eventually change this to a single disk thread and several worker threads.

A thread reads a single file in a critical section. This is to prevent context switching during the read and call some other thread performing a read to another file. Constantly accessing different files will cause serious disk thrashing and degrade performance.

One problem before was the large overhead due to malloc / new calls. I changed the architecture to use my own memory allocator. Since all calls are allocs, and one free at the end, the allocator logic is really simple. In addition, each thread gets a memory allocator. Hence there substantially less contention for memory calls then a naïve multithreaded implementation calling malloc.

Not all trees in the hashtree are the same size. The variance is several orders of magnitude. To offset the workload imbalance, I use a dynamic scheduler through OpenMP for the threads. However, this is not enough as this is online scheduling, but we have all the sizes of the work (offline) so there can be improvements.

I also worked on parallelizing the construction of the frequency predictor. Each thread takes a portion of the tree, and writes to a global array. Since the tree is static, it can be accessed concurrently by the threads. All computation updates a global array, so this is reflected through atomic writes. Initial tests show almost a 60x improvement over 80 threads, which is about 75% of the maximum.

Meeting Your Milestone:

I have met my milestone for these three weeks.

Surprises:

No particular surprises.

Looking Ahead:

The next steps are to continue work on parallelizing the infrastructure. I will try to perform analysis on sequential and parallel portions and determine the speedup on the parallel portions. I will start optimizing the heuristic.

Revisions to Your Future Milestones:
No current revisions are needed. The schedule is continuing as planned in the revised milestones.

Resources Needed:
I ran out of disk space! But this is easily remedied, as more hard drives are being added / mounted.