Homework 3: Parallelize Graph Algorithms for de Novo Genome Assembly

Group members:

Matthew French

Nawavi Naleem

Eric Johnson

1) **A description of your distributed data structures and parallel algorithms**

   As far as distributed data we used shared arrays that each thread will be able to access. The global k-mer array (shared across nodes) sped up graph construction by making each thread process its section of k-mers and place it in the global variable space.

2) **A description of the computational and "communication" motifs of the parallel algorithms.**

   Look into sharing nKmers from getNumKmersInUFX to all threads so it doesn't get done multiple times. Calculate the bytes that need to be read by using the kmer line length * the kmers that the thread will process. Only read in the kmers that the thread will be processing. For graph traversal, only traverse the kmers for each thread. 10 kmers for 10 threads, each thread traverses 1. Create a single contiguous char array for output that will equal max write length * # of kmers to write. Add each kmer to the contiguous block of characters. After that it done, open the pgen.out file and write the contiguous block to it. For writing the data, use a UPC lock to open the file and write and close so the file doesn't get corrupted.

3) **A description of the design choices/optimizations that you tried and how did they affect the performance.**

   While working on the project, we tried to find ways which would speed up the graph construction. Our results showed that the more threads we had the faster our program ran up until 16. After having 16 threads per node, the parallelism was not efficient as there were not enough cores for each thread.

4) **A description of how you avoided race conditions.**

Our group avoided race conditions by first adding barriers and making each thread do its own work on a local copy of the global variable. This way values would not be changed throughout inter calculations.

5) **Discussion of the scalability and relative costs of the parallel graph construction and traversal algorithms.**

We noticed that when running our single node program, speeds increased until we reached 16 threads. Then the speed began to decrease. We think this was due to cores not being used efficiently.