Introduction:

We were invited to participate in the Summer Science Institute, a 10 week full time summer undergraduate research program at the University of Mary Washington. For our research, we studied the performance of various parallel programming libraries. This was set to be an interesting summer because Zach had only a little parallel programming experience, while Jerome did not have any at all.

As a standardized test, each piece of hardware solved a travelling salesman using an un-optimized brute force algorithm. The traveling salesman problem consists of a certain number of cities, each a known distance away from all the other cities. The solution to the problem is the shortest path which goes through all the cities and returns to whatever city at which the path started. We tried problems varying from 10 to 13 cities on a LittleFe portable computing cluster.

The brute force algorithm has a big-O complexity of x!,

Learned:

The brute force solution to the travelling salesman problem is embarrassingly parallel. It is necessary to simply assign a subset of all possible tours to each available thread, and then compare the best solutions found by each thread. To perform this test on all the systems available to us, we had to learn several programming extensions/libraries, including openMP, CUDA, and MPI.

In our experience, openMP was by far the easiest extension to work with. The #pragma directive was quite intuitive to understand. We found that knowledge of how to use a for-loop was almost all that was needed to utilize a pragma directive. Using openMP required almost no knowledge of the hardware to run, and only basic concepts such as race conditions and critical sections to write a scalable program.

CUDA was the most difficult to use. Utilizing the GPU required the most knowledge of hardware memory, as well as the comprehension of how to utilize thread blocks. The method by which you transfer memory from the CPU to the GPU is difficult for programmers new to parallel programming. Of especial confusion was the need to reserve and name memory on the host system, even though the host system should not be interacting with that memory. Also, the GPU can only run functions written specifically for the GPU, which sometimes forces the programmer to rewrite functions for use on the GPU.

MPI was middle of the road in terms of difficulty. The Bootable-Cluster CD (BCCD) software on the LittleFe greatly simplified the environment setup needed to distribute and run an MPI program across multiple computer nodes. Jerome had some trouble determining intuitively which variables within the program were shared and which variables existed as private copies for each node. We found it easy to perform node-specific operations using conditional logic.