K-Means Parallelization

Blake Davis, Chris LaBerge, and Tyson O'Leary

What is our project?
Parallelized Code

- K-Means Clustering is an algorithm that aims to cluster data points into groups
- Machine Learning algorithms are often slow as they often use large amounts of data.
- The code we used is from the Department of Electrical and Computer Engineering at Northwestern University by Wei-king Liao.

Sequential Program	**Given OMP Program**
137 lines of code	251 lines of code
Execution in 16.20 seconds at 13.22 GFLOPS*	Execution in 2.78 seconds at 76.97 GFLOPS*

*With 29,400 data points, 9 dimensions, and 5000 cluster centers

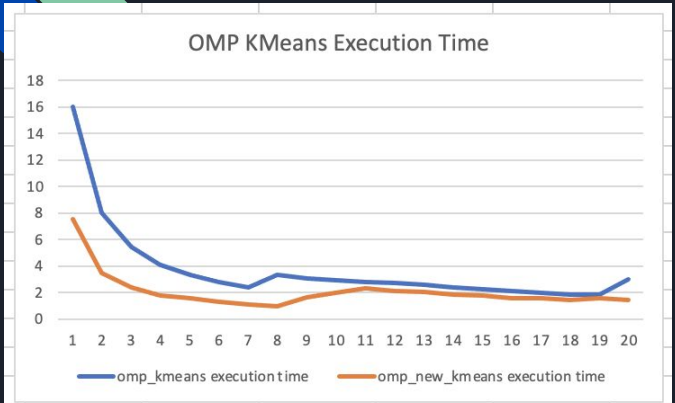
Details of Work: What We Did

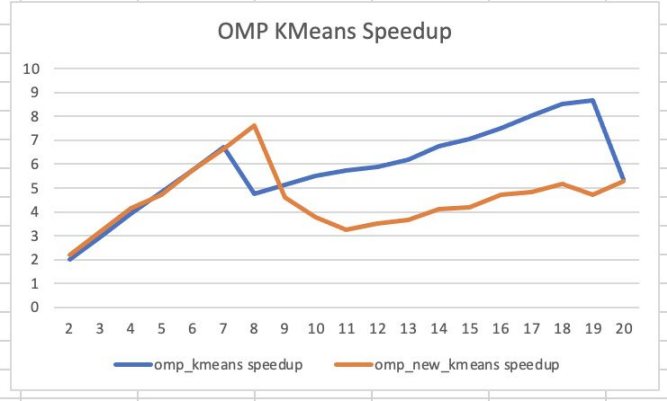
- CUDA proved to be difficult. Decided to pivot to improving the given OMP code.

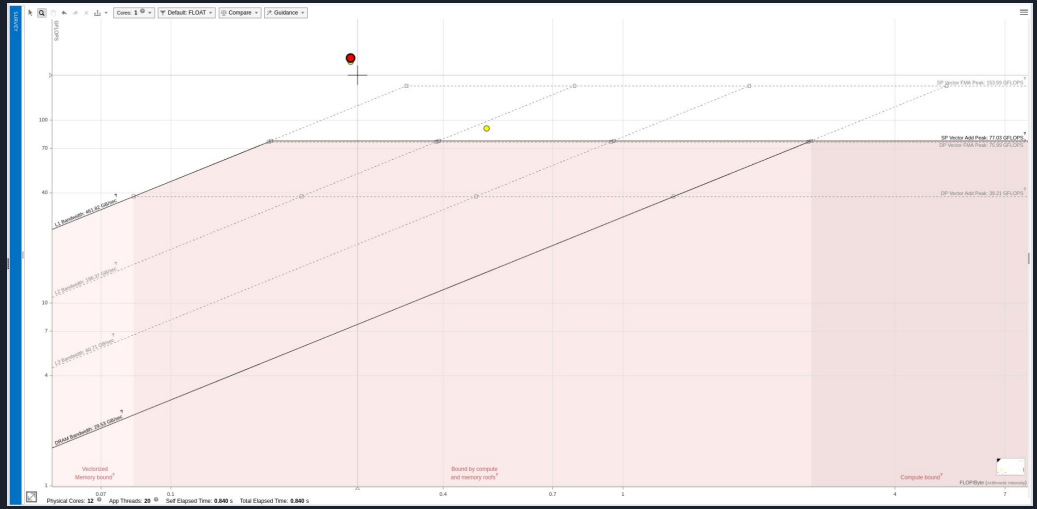
To Improve the OMP version:

- Swapped the loops that looped through each dimension and each cluster with the goal to vectorize loop operations
- In order for our program to have vectorized loops, we had to transpose the matrix that stores the locations of our cluster centers.
- If we did not transpose the matrix, the memory accesses would not be adjacent. This would prevent the vectorization of our loops.

Details of Work: Results







Conclusions and Lessons

- Vectorization
  - Vectorizing a loop requires that all operations done within the innermost loop are vectorizable and that all memory accesses are adjacent. Also requires the ICC compiler
- Cache Locality
  - A result of adjacent memory accesses also means better cache locality
  - Further improvement could be using blocking
- Debugging
  - Parallel debugging skills improved
  - Cannot use valgrind, so had to devise our own method. Using print statements to isolate behavior
- Hail Mary
  - Creating a variable to store duplicate memory accesses did not work
  - Sanjay for the win 🎉

```
// int current_object_coord = objects[i][k];
for (int j=0; j<numClusters; j++){
    distArray[j] = (objects[i][k]-clusters[k][j]) * (objects[i][k]-clusters[k][j]);
```