


A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is light green. They are positioned diagonally, with the blue one partially covering the green one.

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

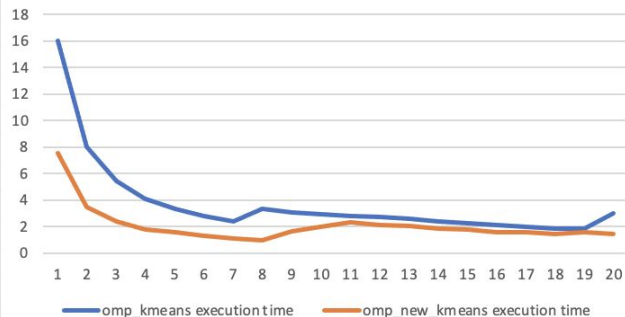
- CUDA proved to be difficult. Decided to pivot to improving the given OMP code.

To Improve the OMP version:

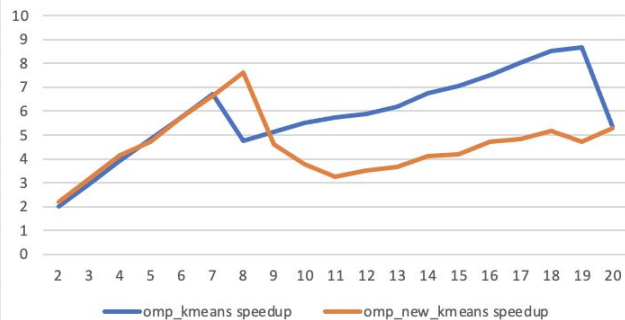
- Swapped the loops that looped through each dimension and each cluster with the goal to vectorize loop operations
- In order for our program to have vectorized loops, we had to transpose the matrix that stores the locations of our cluster centers.
- If we did not transpose the matrix, the memory accesses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

Details of Work: Results

OMP KMeans Execution Time



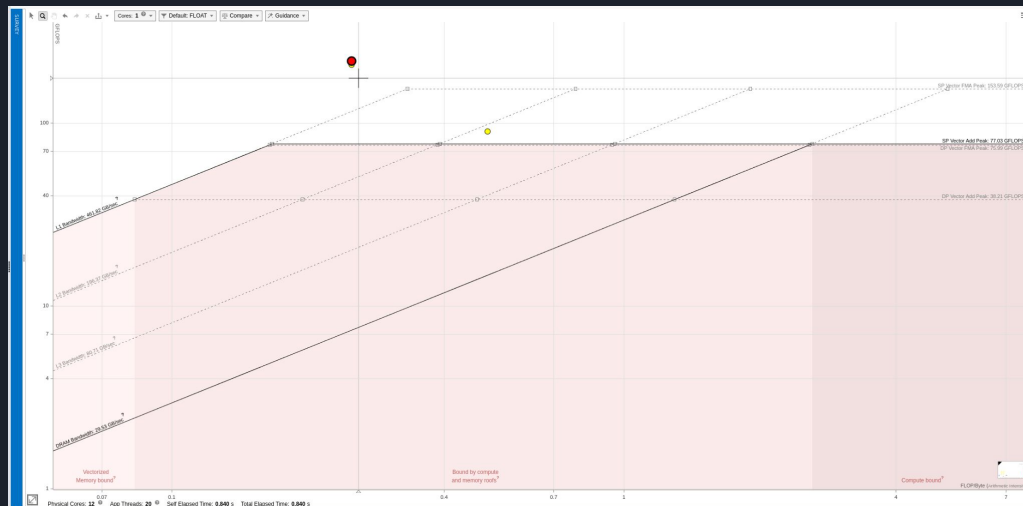
OMP KMeans Speedup



GFLOPS 176.59
GFLOP Count 221.691
FP Arithmetic Intensity 0.259
GINTOPS 39.85

Total
22.40s 100%
12.48s 55.7%
9.92s 44.3%

Consider rebuilding your application with Intel Compiler 16.0 and higher





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]);  
}
```