Sean Goldfarb
209320977

Operating instructions:

To run the program, you need to run the .exe file through the wmpiexec program with two arguments: the input file, which contains the points, numbers of clusters and more information needed, and the output file where the result will be written. The command should look like this: <exe location> <input file> <output file>.

Project explanation:

The project goal is to implement the kmeans algorithm for finding clusters for points space in an efficient parallel form with mpi+omp+cuda for better calculation time.

MPI:

With the mpi we perform a static time division, not in an interval division but rather in a more efficient way to create better load balancing. The division is that if there are n processes than p(i) will calculate the times i*dt + n*k*dt where k is 0,1,2… In this way when a process finds good clusters and it will notify about them to all other processes, so they could stop earlier and not run through all options until it finds his time.

OMP:

With omp we create two threads: one first will activate the cuda calculation (gpu thread),and the second will later be divided into more threads to calculate the algorithm (cpu thread). These two threads divide their work in a dynamic time division, each thread takes his own time to work on. The cpu divide's his work by running different iteration in for loops on different cores in cpu.

CUDA:

The cuda is doing everything that the cpu does but with much more threads which makes the cuda code much more effective in big sets of points.

NOTES:

Although the cuda can find the quality of set of clusters in a more efficient way than the cpu, not all computers can use this for big sets of points because of Windows Display Driver Monitor Time Detection and Recovery Delay is shorter than the calculation time.

The quality method is also the hardest job that needed to be done while running the program and it creates a "Bottleneck" which slows the program significantly. To overcome this problem, it is possible to use a numeric algorithm (That will produce a similar result but not the exact result, with much better computation time). Another way is to use a more Sophisticated algorithm (ex: Convex hull), but it is a hard algorithm to implement in 3d space.