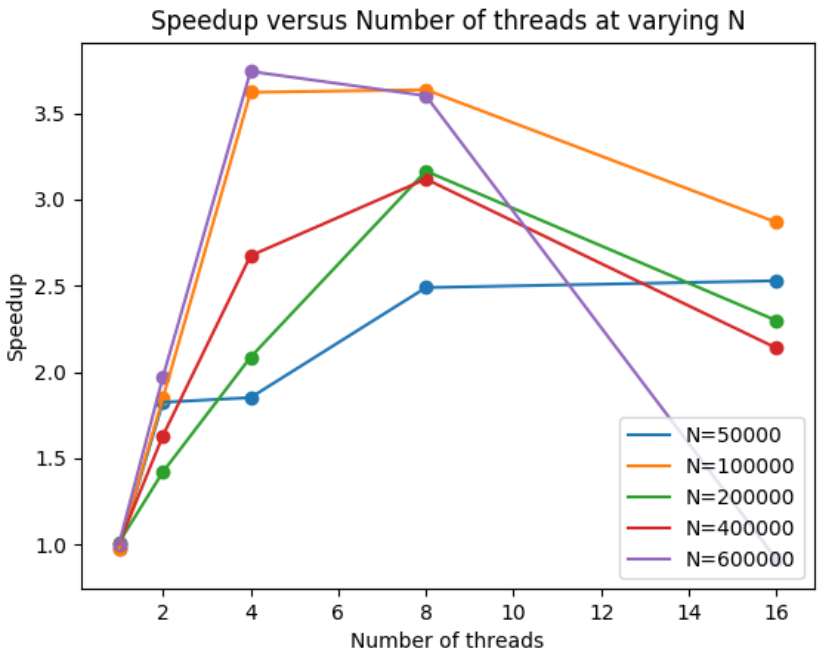
COL380 Lab-1

Anubhav Palway 2016CS10368

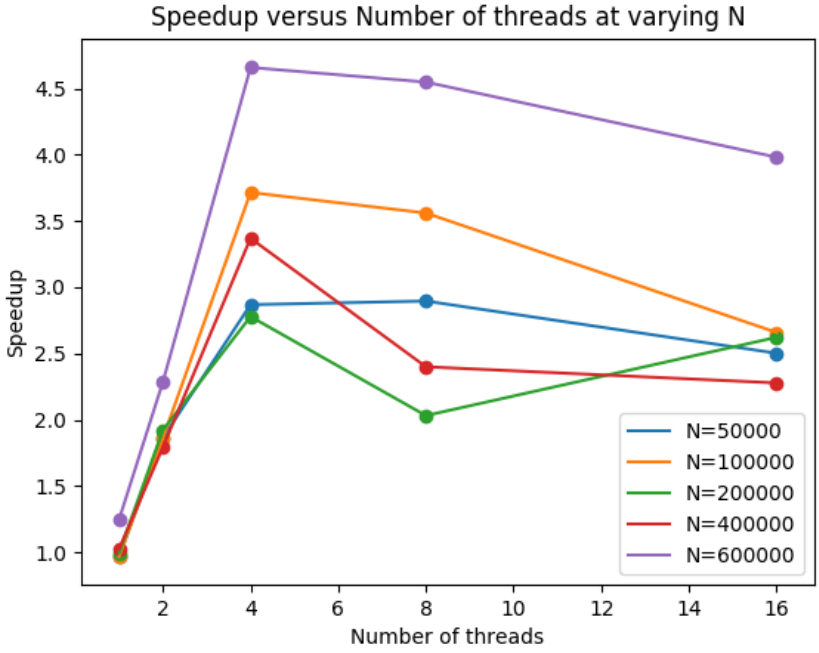
February 2019

1 SpeedUp

1.1 Pthread

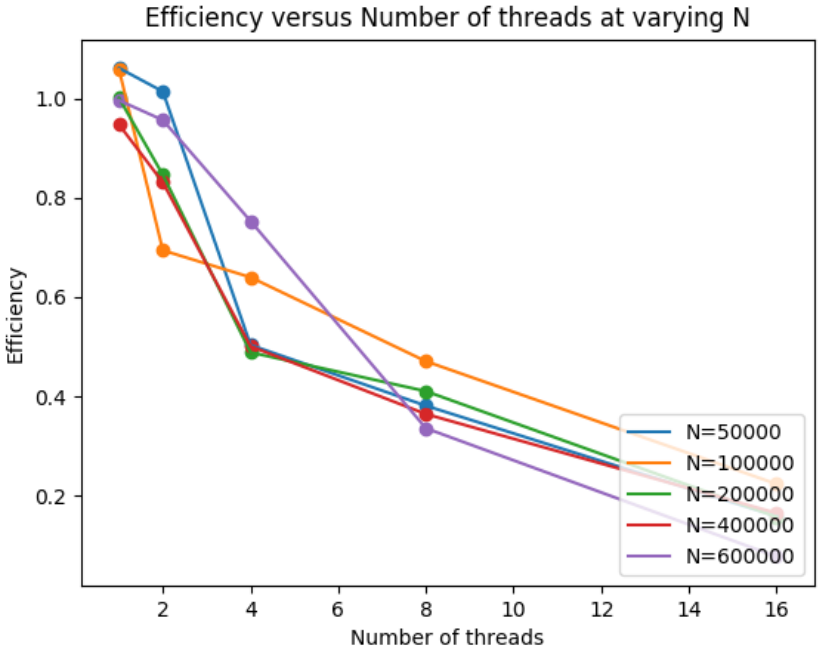


1.2 Open MP

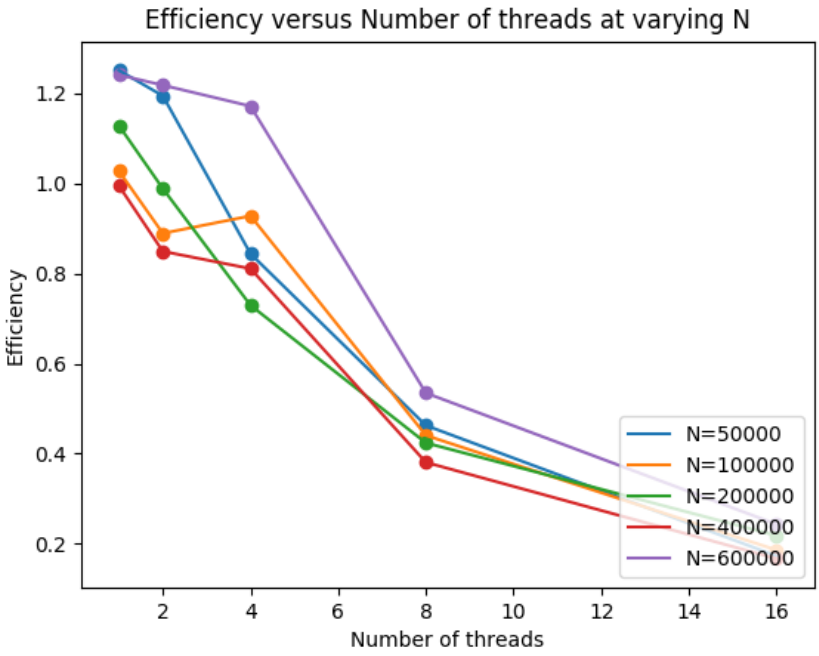


2 Efficiency

2.1 Pthread



2.2 Open MP



3 Design Decision

- For a given k, choose k centroids. I am choosing first k points as the centroids so that the centroids obtained by sequential and parallel algorithms is same.
- Divide dataset to into p parts, where p is the number of threads.
- For each thread p, assign points belonging to its dataset of size n/p to the nearest centroid. And for each cluster, keep adding the point locally in a sum variable. Acquire lock, and for each cluster, add the sum of points to global points.

3.1 Approach 1

- Recompute mean in the main thread. Check for the convergence. If converged then,stop.
- Repeat the above steps.
- This approach creates and destroys threads in each iteration which causes extra overhead. Solution to this is discussed in next section.

3.2 Approach 2

- Wait for all threads to synchronize. Use barrier for this step.
- Thread 0 will recompute the mean and check convergence and set a global flag to 1. Other threads will wait here because of the barrier.
- If flag = 1 then all the threads will exit.
- Repeat above steps.
- This approach does not destroy the threads and uses the same threads for each iterations. Advantage of this over the above approach is that the extra overhead to create and destroy threads is not there.

4 Discussion

- We observe almost same speedup and efficiency graph with pthread and openmp.
- Speedup increases till p = 4 for some N and till p = 8 for others and then decreases. This is because of the communication overhead and other overheads which occurs for higher number of threads.
- Speedup is not affected much after p=8 because my device is octa-core. Rather this is decreased because of communication overheads and scheduling.
- Speedup was different for different runs as the actual time given by processor to this process may be different due to CPU scheduling.
- At same p, efficiency for greater problem size is more. This is because of higher fraction of parallel part for higher N.
- In the above algorithm, we perform data parallelism by dividing the data points into different parts for p threads.

5 How to run

Compilation scripts are provided along with C++ code for kmeans.

./seq K input output1 output2
./pthread K NUM-THREADS input output1 output2
./omp K NUM-THREADS input output1 output2