COL380 Assignment 1 Report

-Vinayak Rastogi
2016CS10345

STRATEGY:-

1) I have created a **struct** named **Point** with attributes:-
    1) **X** -> X-coordinate of the point : type double
    2) **Y** -> Y-coordinate of the point : type double
    3) **Z** -> Z-coordinate of the point : type double
    4) **Cluster** -> Cluster # to which the point belongs: type int

2) Following are the helper functions that I've implemented along with their Descriptions:
    1. **mean_recompute** ->
       Given N,K, the array of struct of data points and array of centroid Points recomputes the Centroid Locations by taking average of locations of the Data points in a particular Cluster
    2. **addtwo** ->
       Given 2 points of type Point(struct), returns a new point as the sum of the Given two points, assuming that both the points belong to the same cluster
    3. **euclid** ->
       Given 2 points of type Point(struct), returns the Euclid's distance between them
    4. **assignclusters** ->
       Given N,K, the array of struct of data points and array of centroid Points recomputes, the nearest cluster for all points and reassigns their values of Point.cluster using euclid
    5. **putback** ->
       puts back values of the centroids in the global vector to be returned by driver func
    6. **checkClosestCluster ->**
       helper for assign clusters function

3) The algorithm is assumed to converge when the cluster values of all the points remains the same before and after an iteration of assign-clusters()

4) For parallelisation, I have parallelized the loop where for each data point, the distances are computed from all centroids and then the index of the distance from the centroid number(from 0 to k-1) is allotted to the Point.cluster value for each point, here this loop is executed by multiple threads where each thread updates the cluster values for each Data point

5) As far as Load Balancing is concerned, in the implementation of Pthreads(where P is the number of threads). each thread gets a total of N divided by P data points for centroid updation(where N is the total number of data points) and hence the Load for each thread is balanced

N (# of Data Points)	K (# of Clusters)	Seq.	Pthreads	Open MP	Num_Threads
5000	3	0.01	0.009	0.008	2
			0.007	0.007	4
			0.006	0.004	8
50,000	4	0.057	0.035	0.035	2
			0.032	0.031	4
			0.029	0.033	8
50,000	10	0.72	0.452	0.462	2
			0.432	0.588	4
			0.337	0.432	8
5,000	10	0.15	0.096	0.087	2
			0.097	0.094	4
			0.091	0.086	8

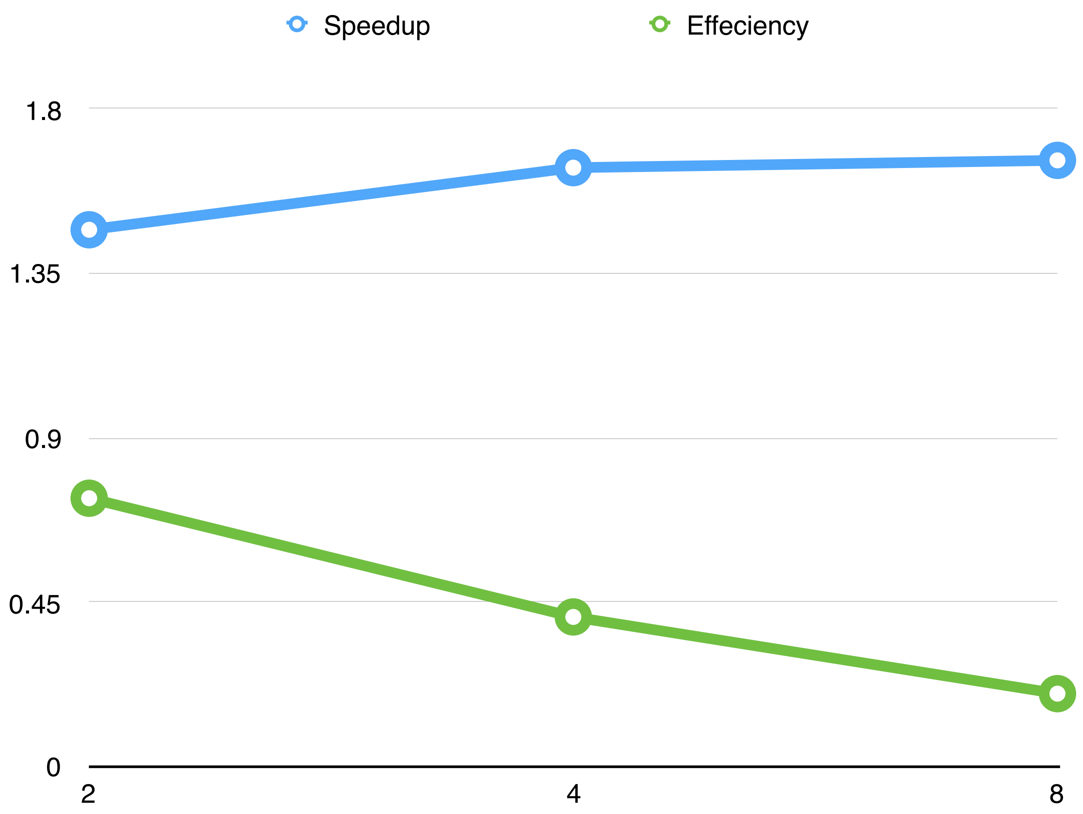
For Problem Size of 50,000 -

Speedup -> 1) open mp : 1.59(p = 2) , 1.61(p = 4), 1.67(p = 8)
2) pthreads : 1.47(p =2) , 1.64(p = 4), 1.66(p = 8)

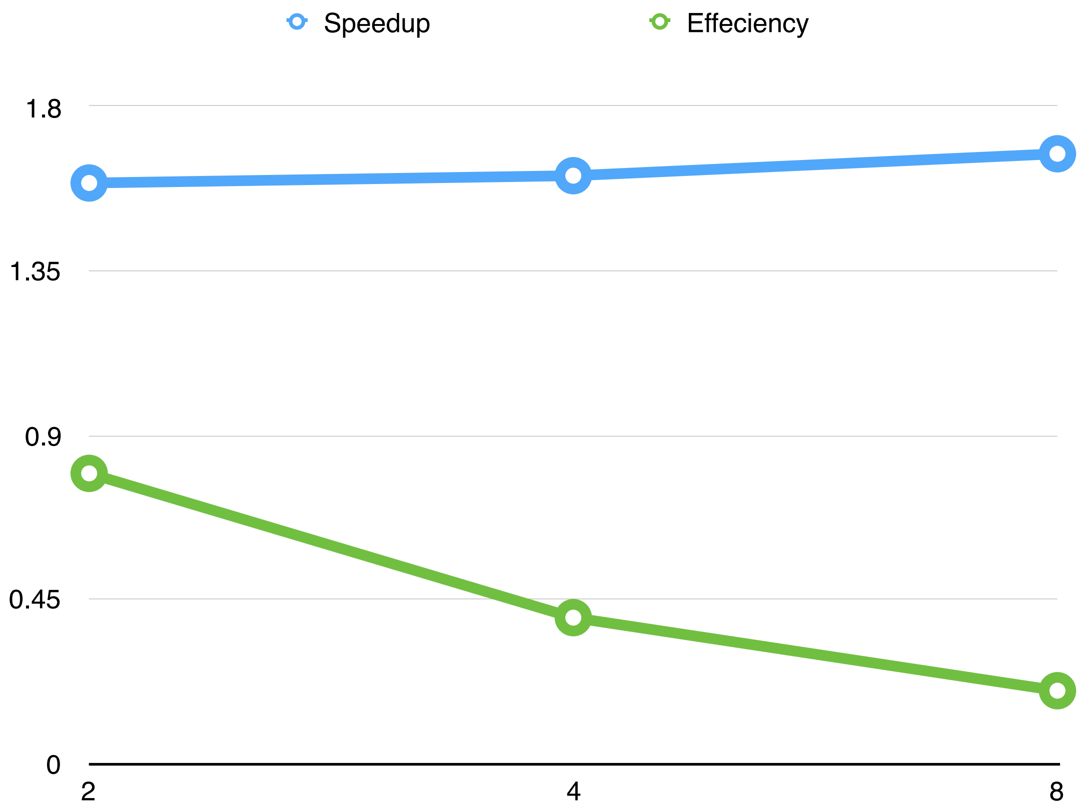
Effeciency -> 1) open mp : 0.795(p = 2) , 0.40(p = 4), 0.20(p = 8)
2) pthreads : 0.735(p =2) , 0.41(p = 4), 0.20(p = 8)

*Speed-up (S) = T (seq) / T (parallel)
*Efficiency (E) = S/ # of threads(P)

FOR P THREADS:



FOR OPEN MP


Speedup
Effeciency