K-means Report (Lab - I)

K.Anirudh
2016CS10362

Design decisions:

A. Sequential K means:

Implementation strategy : For sequential part i have implemented K means by first assigning initial k means to some k randomly selected data points , then i have clustered the data points according to the nearest mean available to them and then updated the old centroids by taking the average of data points present in a cluster and this process of clustering goes on until no datapoint is changing in clusters.

B. Pthread K means:

Implementation strategy: For parallelising using pthreads i thought of worksharing by sharing clustering function among t provided threads since there we will iterate through whole dataset and calculate distances from each point to all the k means i have parallelised it using pthread two race conditions araised during implementing pthreads because one for the check_count (checking if the point is changing to new cluster) and other for summation of coordinates, In check_count threads count their individual swaps but we need total shifts so i added to a global variable using pthread_mutex lock and preserved mutual exclusion for correctness.
I have optimised the update centroids , by combining the summation of coordinates part in clustering function and so threads individually count summation for some part of data set and then added to global var by acquiring lock
Tried but failed : i have checked how pthreads going to work by splitting threads among for almost every function and observed that for many functions even instructions in order of N (datasets) p thread’s work sharing made loose speedup maybe inter thread

communication and thread generation and destroying is taking more time than executing sequentially.
In plots it can be seen that even for these optimisations pthread did not work well for my case because time taken for pthreads is greater than sequential in my code for almost all threads

C. Openmp Kmeans:

Implementation strategy: For openmp implementation i have used same optimisations as in case of pthreads one for checking number of updates happened and other for calculating mean and average of data points by splitting among threads used omp parallel for and i have tried optimising with omp parallel sections and manual splitting of threads by omp_get_thread_num but then i am getting much less performance so i left with just two major optimisations and managed data races in them by using omp_set_lock .

Below i have plotted time ,speedup and efficiency vs datasizes plots for 6 clusters and 2 to 15 threads in case of pthreads and omp.

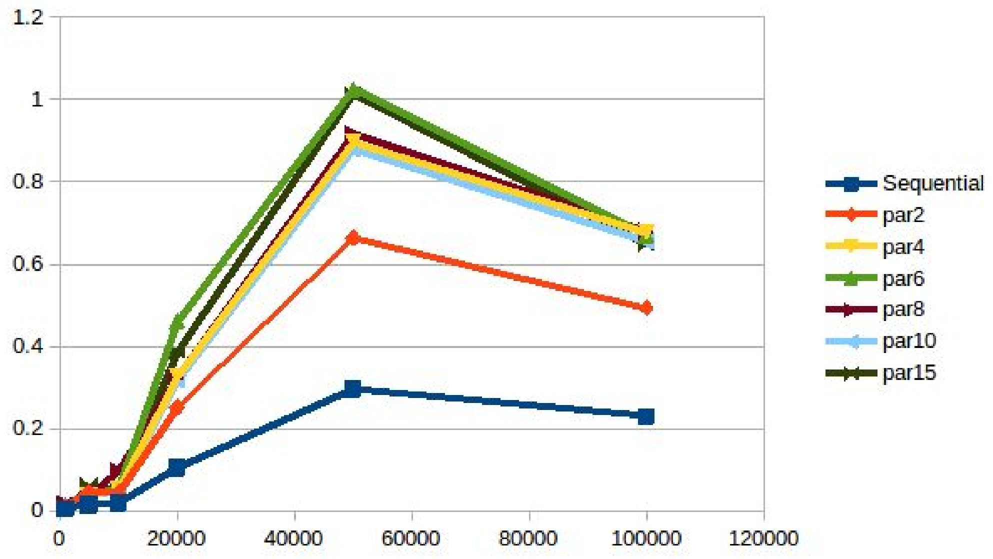

Fig 1. Time Vs DataSet (for Pthreads K means and Sequential K means).

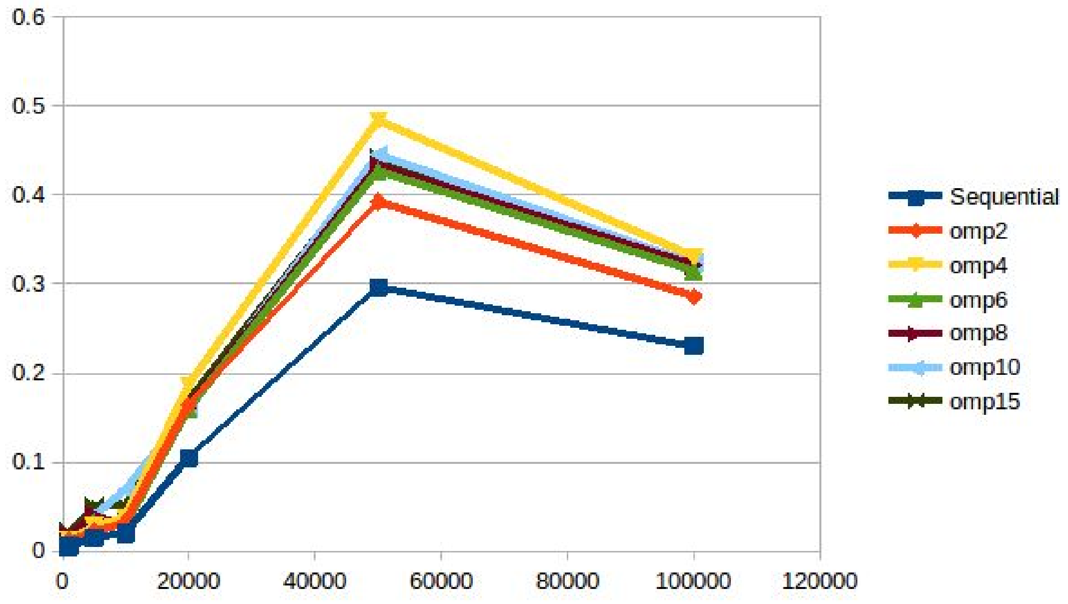

Fig 2. Time vs DataSet (for Openmp K means and Sequential K means)

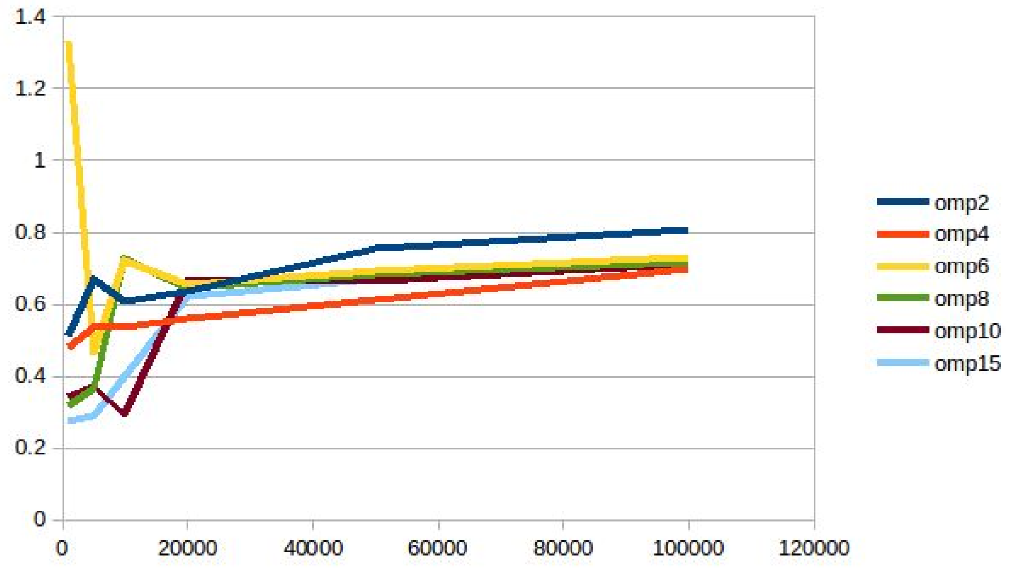

Fig 3. Speedup vs DataSet (for Openmp K means for 2 to 15 threads).

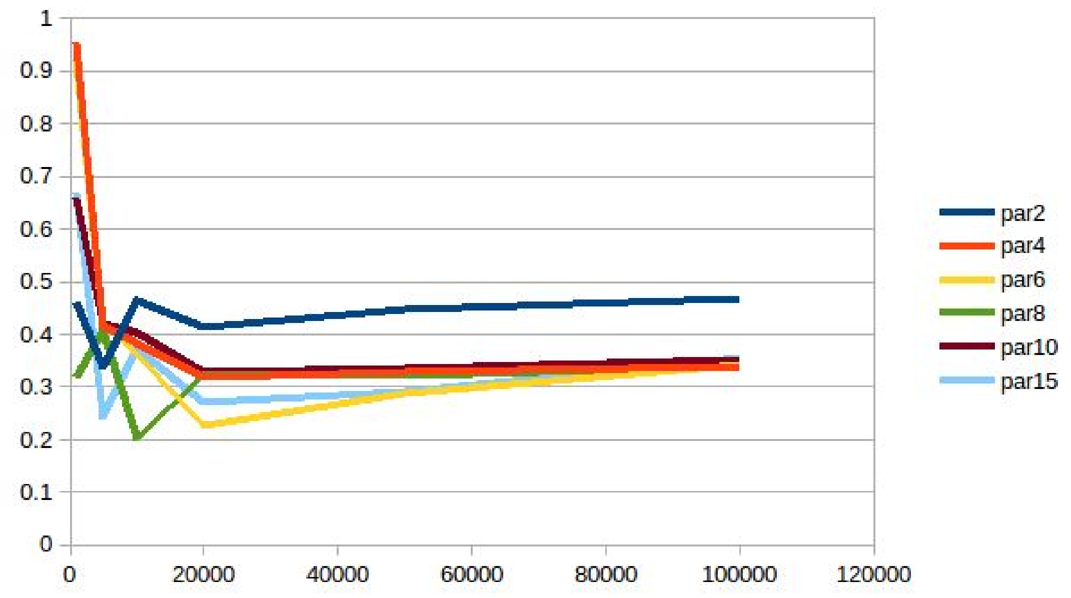

Fig 4. Speedup vs DataSet (for Pthread K means for 2 to 15 threads).

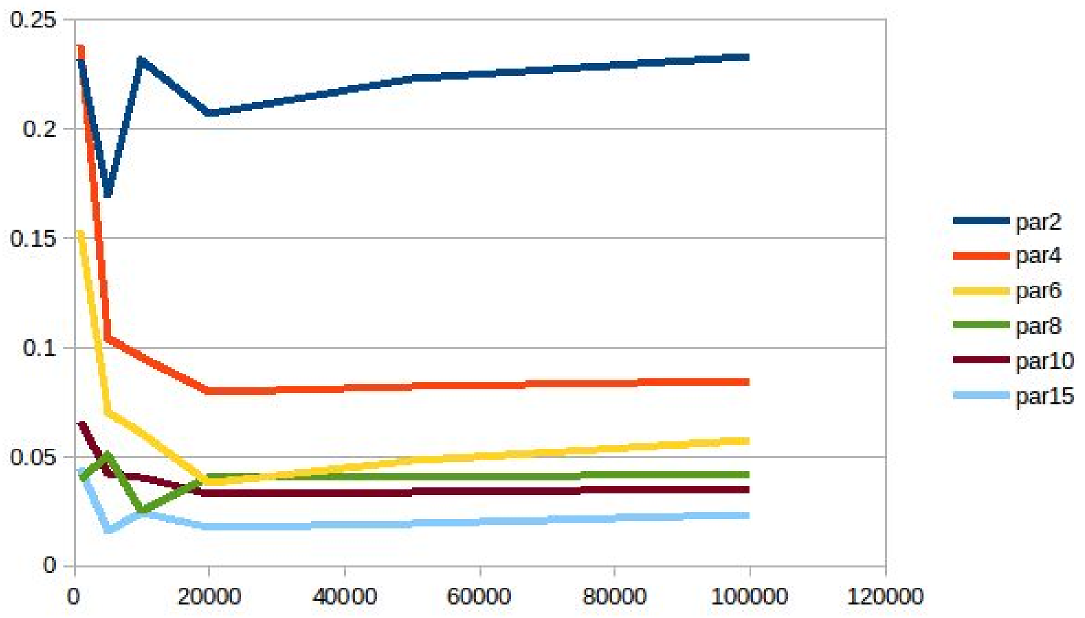

Fig 5. Efficiency vs DataSet (for Pthread K means for 2 to 15 threads).

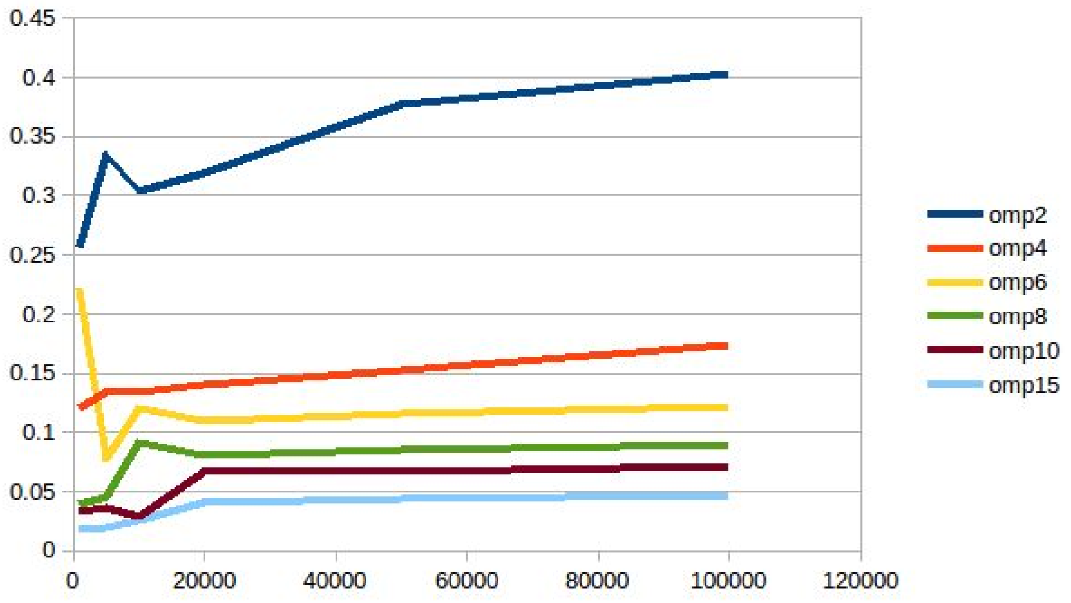

Fig 6. Efficiency vs DataSet (for Openmp K means for 2 to 15 threads).