K-Means clustering

Pradyumna Meena - 2016CS10375

Default terms

- N = number of points in the dataset
- K = number of clusters
- Num_threads = number of threads to be used in openmp and pthread implementation
- Maximum number of iterations allowed = 300

Key aspects of the implemented algorithm

These are few key aspects which had a significant impact on the performance of the program compared to others

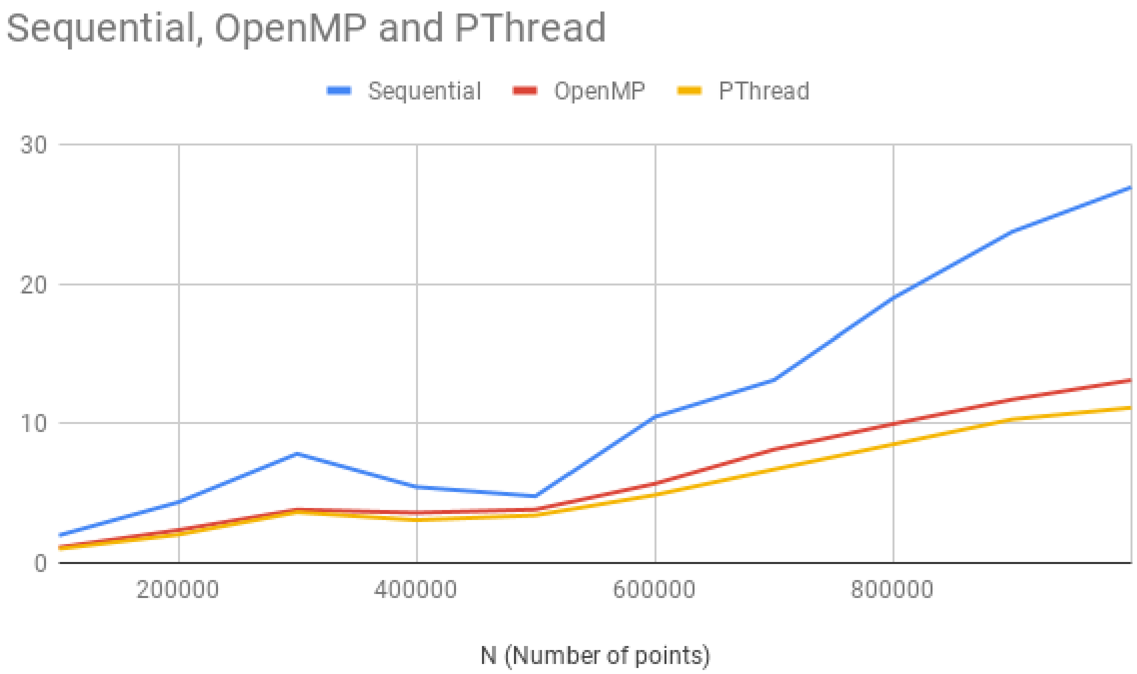
- No use of data structures like vector, arrays to avoid risk of refetching in case of false sharing. Pointers enable us to access same memory location from different threads without the risk of false sharing though data races are still there
- Avoiding additional function for small tasks like distance between two points. Functions doing such small tasks when called repetitively have significant effect on performance

Underlying risks of false sharing and data races

As said above with the use of vectors or arrays comes the risk of false sharing since these had to be shared between the threads and if not taken care properly can lead to recursive refetching of data. Data races come into picture when centroids are updated. In a parallel implementation normal strategy would be to divide the initially given points into N/K parts and each thread updates the centroid depending on the cluster it was assigned to. This may lead to two threads updating the same location and hence inconsistency arises. To handle this a 2-D array of dimensions num_threads x 3K was used. Each thread updates in its specified row of the array. After all threads have completed their part, the centroids are computed using K threads.

Performance Plots

- Parameters varied are num_threads, N, K
  - **Note:** The unnatural variations are due to different number of clusters in each of the case. Generally the one with more clusters take more time to converge.
  - **Convergence criteria:** Change in coordinates of centroid<epsilon and max_iterations



This is for 4 (#cores on my system) threads in openmp and pthread implementation