Brief Explanation

- **Method:**
  - I have parellised K-Means algorithm when K is known already and feeded this K using elbow method which was done independently.
  - Reason: I did this because goal of assignment was to speedup the problem of clustering using Parallel Techniques.
- **Approach and Pseudo Code:**
  - Kmeans algorithm contain 3 main steps. ``Initializing cluster mean``, ``Reassigning Points to Cluster Means`` and ``Recomputing Cluster Means``.
  - Let there be N data points , P process to be divided into K clusters. ``Reassigning Cluster Means`` step is independent for any two points, so I divided points so that each process has approx. N/P points and find nearest Cluster Mean in K steps. So, Computing of CLuster Means now takes (N*K/P). Which will theoretically give Linear speedup.
  - ``Recomputing Cluster Means`` is not independent on the division of data points, But it can be done in two steps. First step calculates ``Sum of all points and cluster size for each cluster``. Second step calculates avg by using these two. Calculating sum and cluster size is easy {Each node calculates them for their own cluster and Allreduce takes their sum}. Second step takes [N/P] time
  - Since ``Init cluster Mean`` will not be an overhead compared to other main steps, it is computed as follows. First root{0} process sends random no. to all process via MPI_Bcast and then each of the process initialize cluster mean if they have access to corresponding point. Finally, each of them communicate this populated cluster means via MPI_Allreduce.
  - Rest of the Algorithm follows same Pseudo Code as Sequential K-Means. First, Initialize Clusters then keep repeating{``Reassign Points to Cluster`` then ``Recompute Clusters``} until max_iterations is reached or Convergence is obtained. Currently, max_iteration is 200 and convergence is mean shift is less than 5e-5 for all clusters.

Data Decomposition

- I divide File into P non-overlapping contiguous parts of size FILE_SIZE/P rounded so that each file contain full points{no points are cut-off}. Thus, each file has avg. N/P points. Since the file chunks are non-overlapping, I used MPI Parallel I/O for faster Input-Output.

Heuristic

- For determining K, I used elbow method. I ran K-Means using python for different Ks upto 100 for data1 and 50 for data2.
- I was able to see elbow formed at around 50 for data1 and 10-20 for data2. As particles timestamp are near, number of cluster will mostly be same. So, I kept K fixed for all timestamps and ran my Kmeans to find cluster mean.

Observations

1. Total time/Process time changes very less after certain P:

   a. We see that Total time does not change much after (14/16 for Total Time and 12-16 for Processing Time) process. This can be seen as direct effect of Amdahl’s Law which says that every program has inherently sequential part which cannot be decreased further by just increasing parallelism. We can say that after this point increasing P wont make much difference.
   b. In our code, The fact that each assign step must be done after recompute and vice-versa can be an example of Inherent Sequentialism of Amdahl’s Law.

   See fig1.png

2. Variability of Pre-processing time and Increasing general trend:
   - Since increasing nodes increases I/O operations, after some point the effect of read/write access time becomes significant and that is why we see general median increase as P increases.
   - Variability(seen as outliers denoted by circle) is expected as no. of I/O operations increases. With P.

   See fig2.png
3. HPC cluster matches theoretical predictions more closely than CSE cluster:
   a. This is evident by the fact that HPC is more fast and uses better scheduling algorithm than CSE cluster and also by the fact that it has better network connectivity. Also, HPC has less cluttered process and it is exclusive and CSE is shared.
4. Speedup starts decreasing after some point for CSE cluster but not for HPC:
   a. The “special point”after which Speedup starts decreasing is where increasing parallelism does not help us. It also depends on the computing power of machine. Since HPC has better computing power, its “special point” is further than CSE. In 1 to 60 range, CSE’s cluster “special point” is reached but increasing P may still help in case of HPC.

   See fig3.png and fig1.png