Parallel Algorithms
K-means Clustering

Instructor: Dr. Russ Miller
Name: Mrunal Inge
UB Person Number: 50337040
OVERVIEW:

- Introduction to K-means
- Sequential Algorithm
- Parallel Approach
- Sample Readings
- Graphs
- Conclusion
Introduction:

- Clustering is the process of dividing the entire data into groups (also known as clusters) based on the patterns in the data.

- The k-means clustering method is an unsupervised machine learning technique used to identify clusters of data objects in a dataset.

- The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centers, one for each cluster.

- Conventional k-means requires only a few steps. The first step is to randomly select k centroids, where k is equal to the number of clusters you choose. Centroids are data points representing the center of a cluster.
Sequential Algorithm

✧ Specify the number ‘k’ of the clusters to be assigned.

✧ Randomly initialize ‘k’ centroids.

✧ Assign each point to its nearest centroid by calculating euclidian distance from the point to the centroid.

✧ Compute the new centroid by calculating the mean of all points in the cluster.

✧ Repeat steps 3 and 4 till there is no change in the centroid positions.
Initial Data points
Steps
After K-Means clustering
Example:

- Suppose we take a few data points \([1,3,5,12,13,14]\). Initially let’s assume cluster centers to be 1 and 5.

- Clusters \(c_1: [1,3]\) \(c_2: [5,12,13,14]\). We recalculate centroids by calculating mean of the cluster: \(c_1 = \text{mean}(1,3) = 2\) \(c_2 = \text{mean}(5,12,13,14) = 11\).

- Now we calculate Euclidean distance of each cluster point w.r.t new centroids and update the clusters: \(c_1 = [1,3,5]\) \(c_2 = [12,13,14]\).

- Again we calculate mean \(c_1 = \text{mean}(1,3,5) = 3\) \(c_2 = \text{mean}(12,13,14) = 13\).

- Again calculate euclidean distance and calculate points and rearrange clusters \(c_1 = [1,3,5]\) \(c_2 = [12,13,14]\).

- No change in cluster. This will be our stopping point.
Parallel Approach:

- Data is divided among each processor equally.
- The processor with rank 0 initializes k random centroids and broadcasts it to all other processors.
- Each processor then locally calculates distance of the point from each centroid and is divided into K clusters.
- Then the processor locally calculates the sum of each cluster and returns the sum and length of each cluster to the processor with rank 0.
- Then, Processor with rank 0, receives the sum and length of the clusters and calculate the new clusters centroids, and broadcast it to all the processors.
- Above procedure of clustering continues for n iterations.
Readings:

Below readings show the new cluster centers calculated for different number of data points with increasing number of processors.

```bash
[mrunalna@vortex2:~]$ ./kmeans-parallel.sh
Anaconda Python 2.7 version 2019.10 has been loaded.
Intel-MPI is in your path. This is adequate for compiling and running most codes. Source the
/util/academic/intel/17.0/compilers_and_libraries_2017/linux/mpi/intel64/bin/mpivars.sh file for more features.
[[8.02073600 1.04728374]
[0.97921048 0.94142644]
[5.02002949 5.05160844]]
0.0372860431671
[mrunalna@vortex2:~]$ ./kmeans-parallel.sh
Anaconda Python 2.7 version 2019.10 has been loaded.
Intel-MPI is in your path. This is adequate for compiling and running most codes. Source the
/util/academic/intel/17.0/compilers_and_libraries_2017/linux/mpi/intel64/bin/mpivars.sh file for more features.
[[1.00343614 0.93228404]
[4.96220196 4.95370008]
[7.90394033 0.9955709 ]]
0.0279741287231
[mrunalna@vortex2:~]$ ./kmeans-parallel.sh
Anaconda Python 2.7 version 2019.10 has been loaded.
Intel-MPI is in your path. This is adequate for compiling and running most codes. Source the
/util/academic/intel/17.0/compilers_and_libraries_2017/linux/mpi/intel64/bin/mpivars.sh file for more features.
[[1.01064311 0.92602857]
[4.88073738 5.01299327]
[7.97499368 1.03243626]]
0.0527241230011
[mrunalna@vortex2:~]$ 
```
## Sample Readings of Data Points and Number of Processors

<table>
<thead>
<tr>
<th>Number of Data Points</th>
<th>2 processors</th>
<th>4 processors</th>
<th>8 processors</th>
<th>16 processors</th>
<th>32 processors</th>
<th>64 processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>0.03410450617</td>
<td>0.02399983</td>
<td>0.03170152509</td>
<td>0.128037254</td>
<td>0.331446768</td>
<td>2.418035666</td>
</tr>
<tr>
<td>15000</td>
<td>0.14970914</td>
<td>0.1061520179</td>
<td>0.0765351804</td>
<td>0.07758331299</td>
<td>0.06714200974</td>
<td>4.067548831</td>
</tr>
<tr>
<td>150000</td>
<td>1.318409486</td>
<td>0.9297121763</td>
<td>0.7582879464</td>
<td>0.6338289976</td>
<td>0.7354993025</td>
<td>3.061531308</td>
</tr>
<tr>
<td>15000000</td>
<td>152.7570831</td>
<td>103.5258799</td>
<td>81.64824339</td>
<td>70.10960893</td>
<td>69.4403375</td>
<td>132.9132652</td>
</tr>
</tbody>
</table>
Amdahl's (Time v/s Processors) 1500000 points
Conclusion:

• We observe significant speedup unto 32 processors.

• Cost of communication affects the speedup significantly when the number of processors exceeds 64.

• The significant change in the speedup is observed for a large number of data and more number of clusters.
References:

Thank You!