K Means Clustering in Parallel

Instructor- Dr. Russ Miller

Presenter - Nilesh Dogra

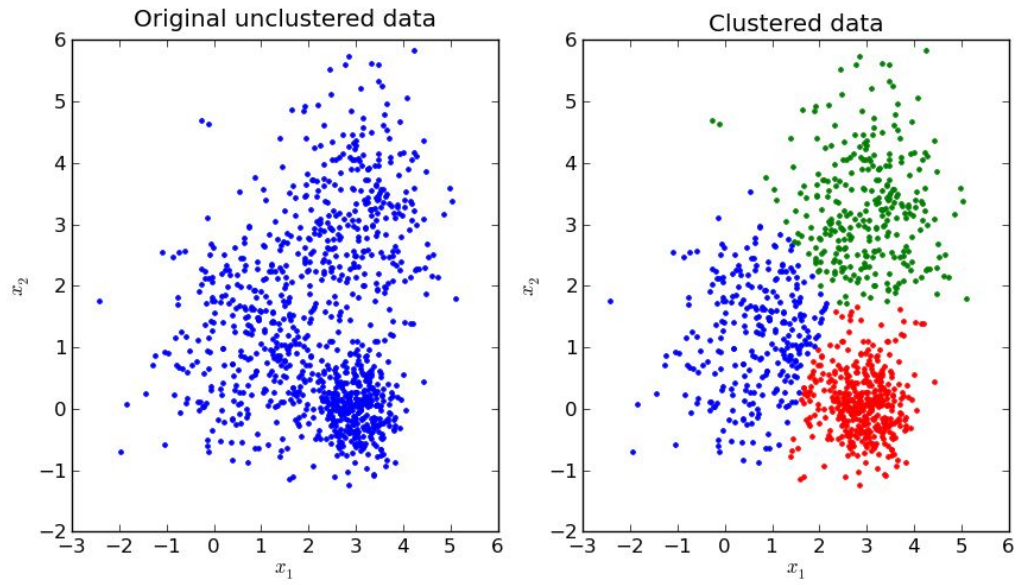
CSE 633 - Parallel Algorithms

Date - 05/10/2022

University at Buffalo The State University of New York

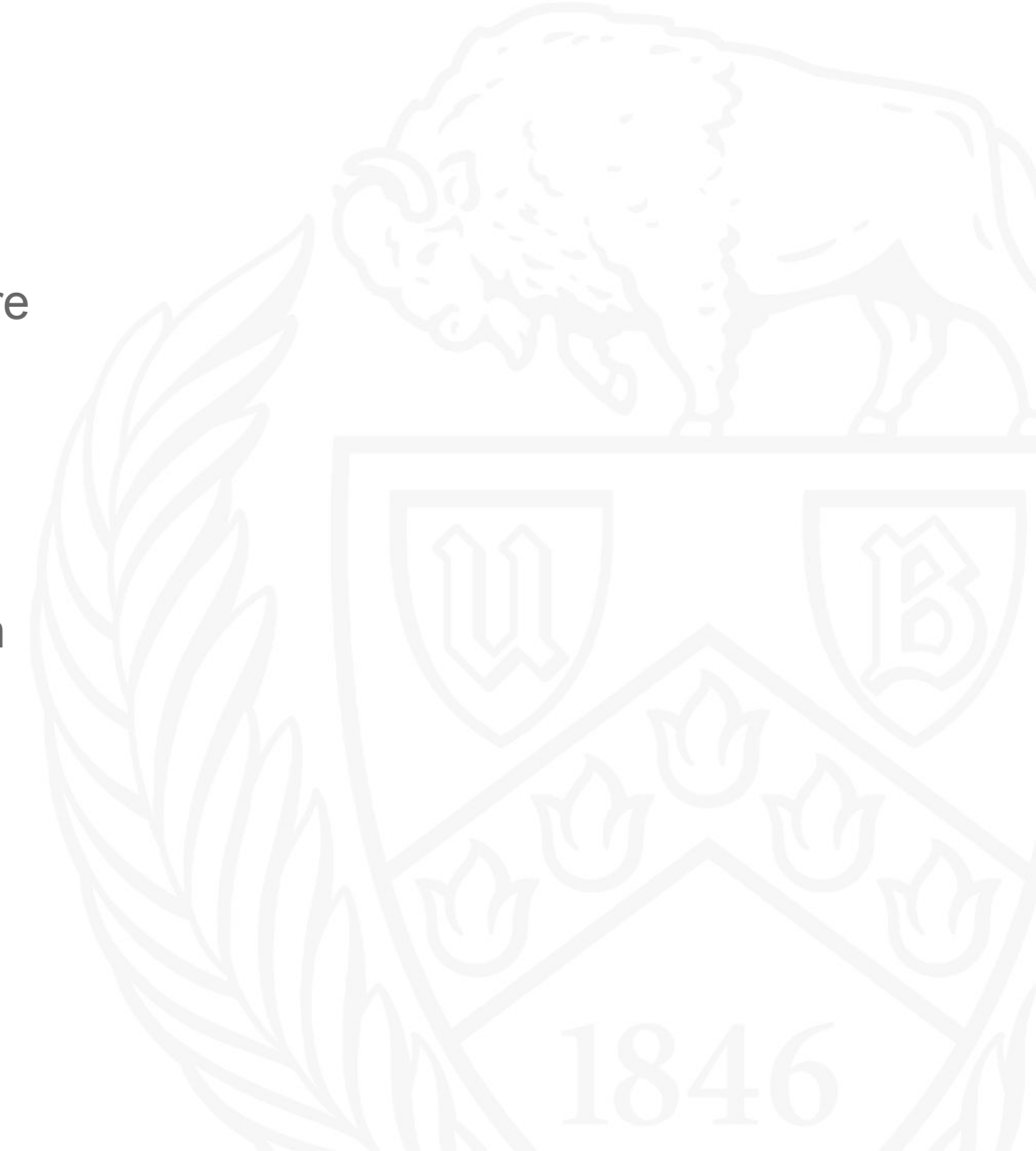
What is K means Clustering

1. Unsupervised machine learning algorithm.
2. K is the number of clusters.



Working of Sequential K means Clustering

1. Select random K cluster points from all the numbers which are called centroids.
2. Assign remaining numbers to points nearest to it's K cluster point.
3. Re-calculate K cluster points by calculating mean of the each cluster.
4. Repeat steps 2 - 3
5. Stop when there is no change in clusters.



Example of K means clustering

Numbers = {2,5,8,6,10,7,9,1,3,4}

K = 2

Random Cluster points = 4, 8

Cluster 1 = {2,5,6,1,3,4}

Cluster 2 = {8,10,7,9}

New Mean or Cluster points

K1 = 3, K2 = 8

Cluster 1 = {2,5,1,3,4}

Cluster 2 = {8,6,10,7,9}

New Mean or Cluster Points

K1 = 3, K2 = 8

Cluster 1 = {2,5,1,3,4}

Cluster 2 = {8,6,10,7,9}

No change in clusters

Stop Algorithm

Parallel K means clustering

1.Divide N data across each node equally.

2.Randomly select K cluster points(Centroids) on each node.

3.Allocate N to closest centroid K.

4.Calculate new centroids and broadcast it to across each node.

5.Allocate N to closest cluster points across all centroids.

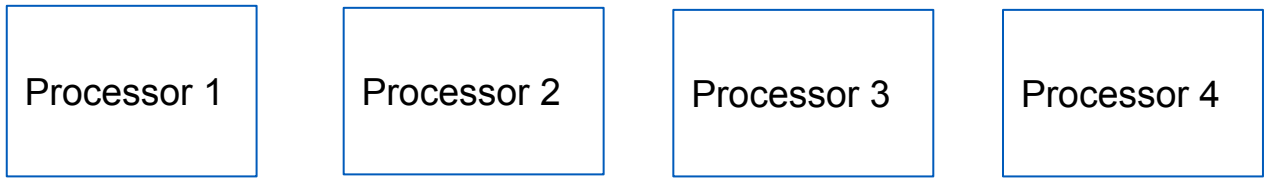
6.Repeat till there are no transfers of points between clusters.

Working of Parallel K Means Algorithm

Understanding the flow of model with example:

Initials - Total Clusters = 256, Total Data - 128*128 = 16384 ,

Processors = 4, Range - 1024



[22, 865, 541, 912, 34, 45 ….] 16384 data with each number between 1 and 1024

Initially each Processor will divide data equally among themselves.

Each Processor = 4096 data

1 of 4 processors:

Each processor centroid count = Total centroid/Processors = 256/4 = 64

Initial Centroids = 64,
Each Centroid initialized with mean

We will be initializing centroid of all processors by step.

1st Centroid point = 1.0

2nd Centroid point = Earlier Centroid + (1024/256) = 5.0 and so on till last processor's last centroid mean.

256th Centroid point = Earlier Centroid + (1024/256)

Why not random?

Now once all centroids are initialized, each processors in parallel will work on their data.

Proc 1 - Assign 4096 data points to its centroid based on its closest distance	Proc 2 - Assign 4096 data points to its centroid based on its closest distance	Proc 3 - Assign 4096 data points to its centroid based on its closest distance	Proc 4 - Assign 4096 data points to its centroid based on its closest distance

Now each value in each centroid of all Processors will calculate their closest distance centroid, could be in another processor or another centroid of same processor.

Once it's done we will send and receive values based on the above step.

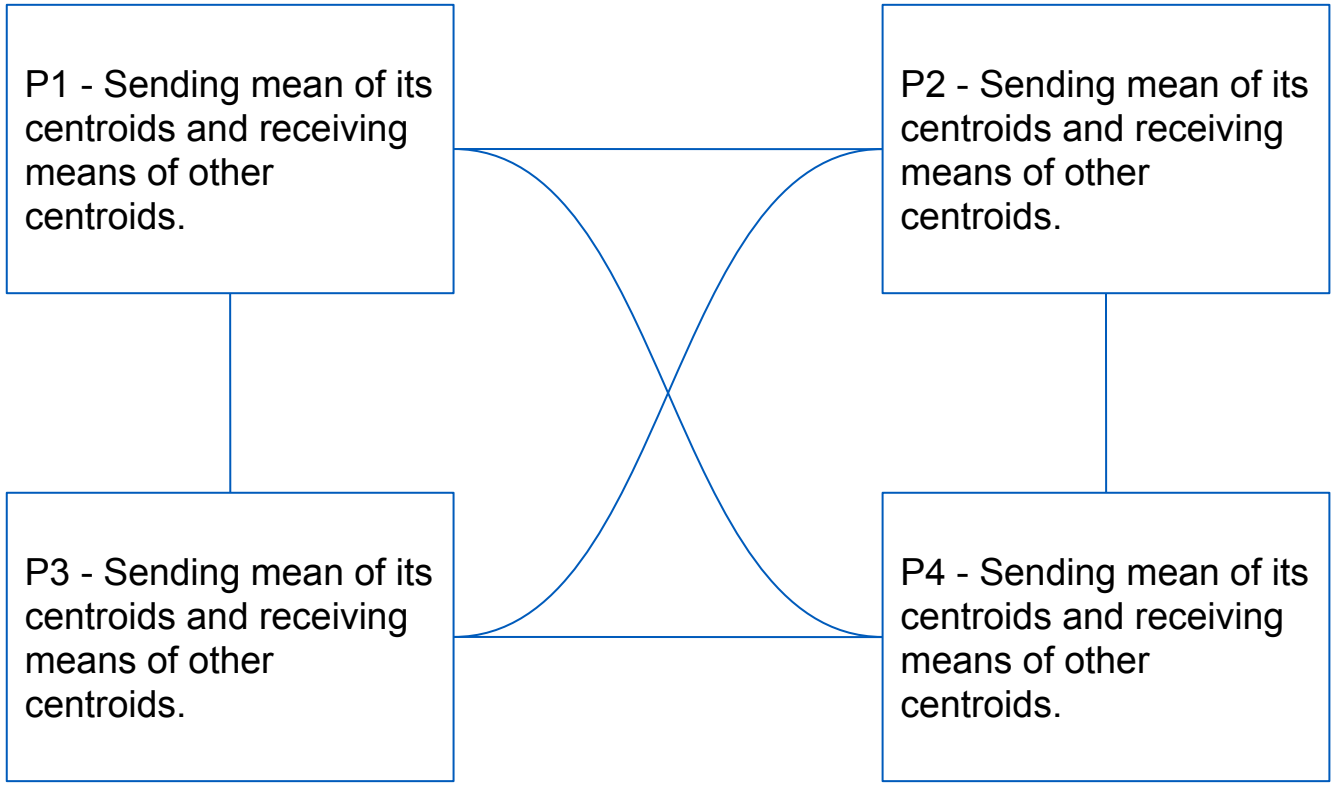
Re calculate Centroids by calculating means of clusters.

Broadcast and Receive the means for all centroids.

Repeat these steps till we get same Centroid Means for 2 Iterations.

Broadcasting of Centroids

Send and Receive Centroids



Results

So there were multiple experiments done with different data size,range,clusters.

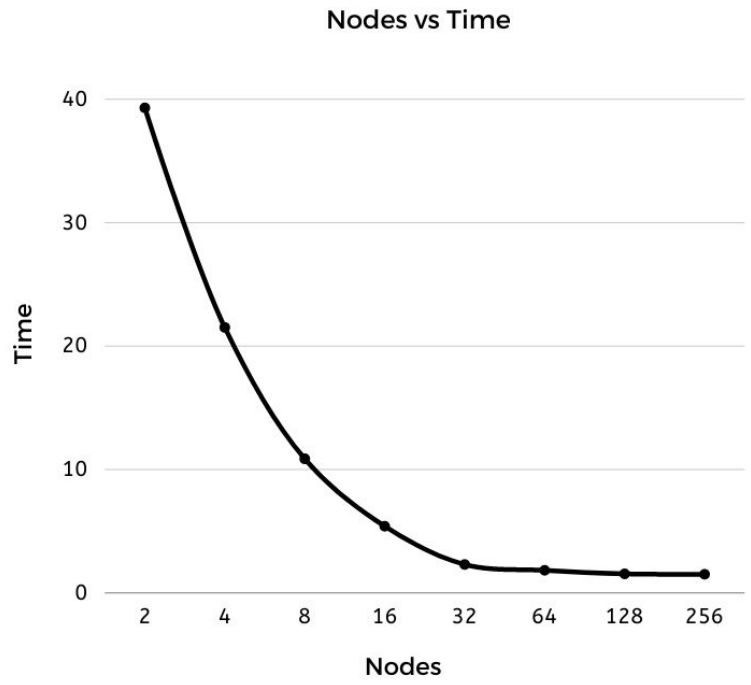
Let’s analyse the results and what we have inferred from it.

1. For 512*512 data, 256 clusters

   Range - 1024

Note - 256 achieved by running 2 tasks for 128 processors

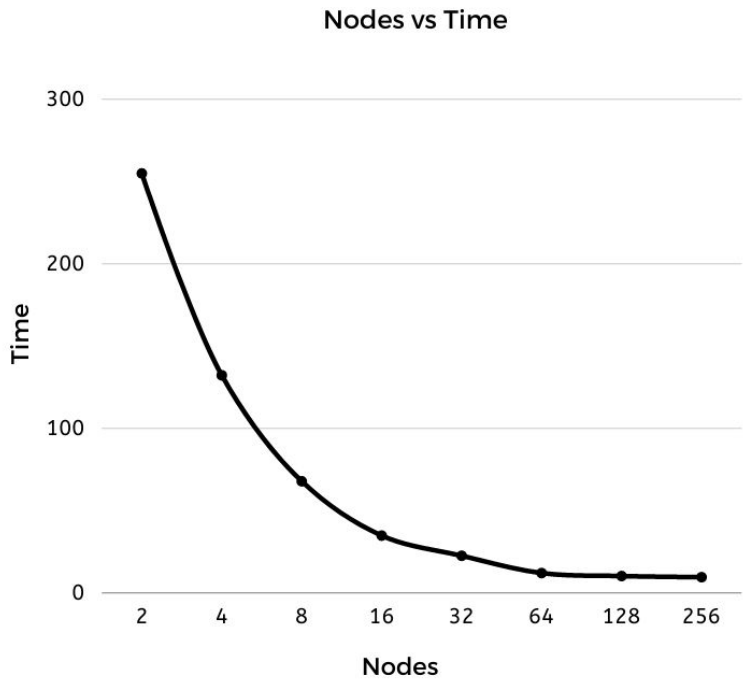
Processors	Time
2	39.313
4	21.522
8	10.872
16	5.418
32	2.313
64	1.836
128	1.554
256	1.521



2. For 512*512 data, 256 clusters

   Range - 10240

Processors	Time
2	255.006
4	132.319
8	67.878
16	34.89
32	22.636
64	12.124
128	10.315
256	9.682

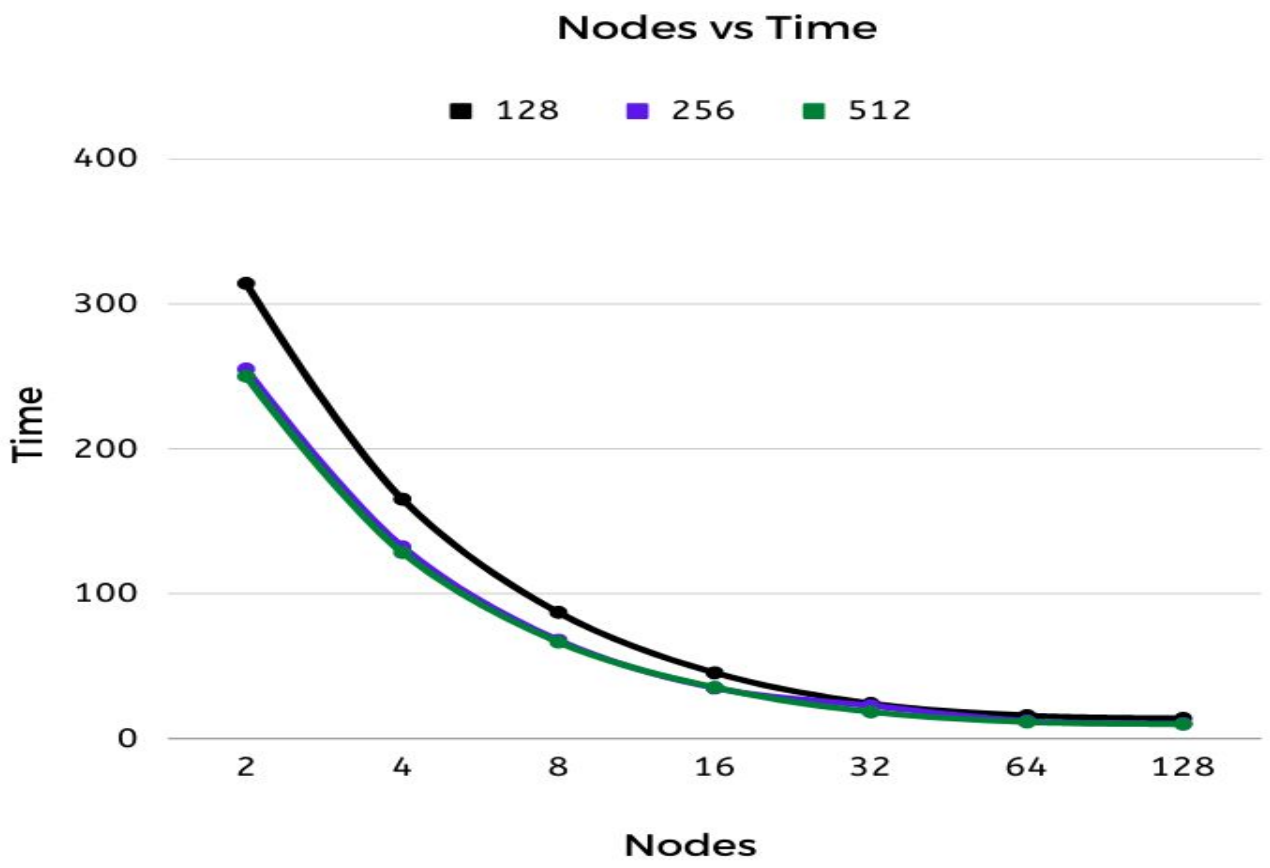


3. Now we will calculate the times for same data 512*512, Clusters - 128, 256, 512

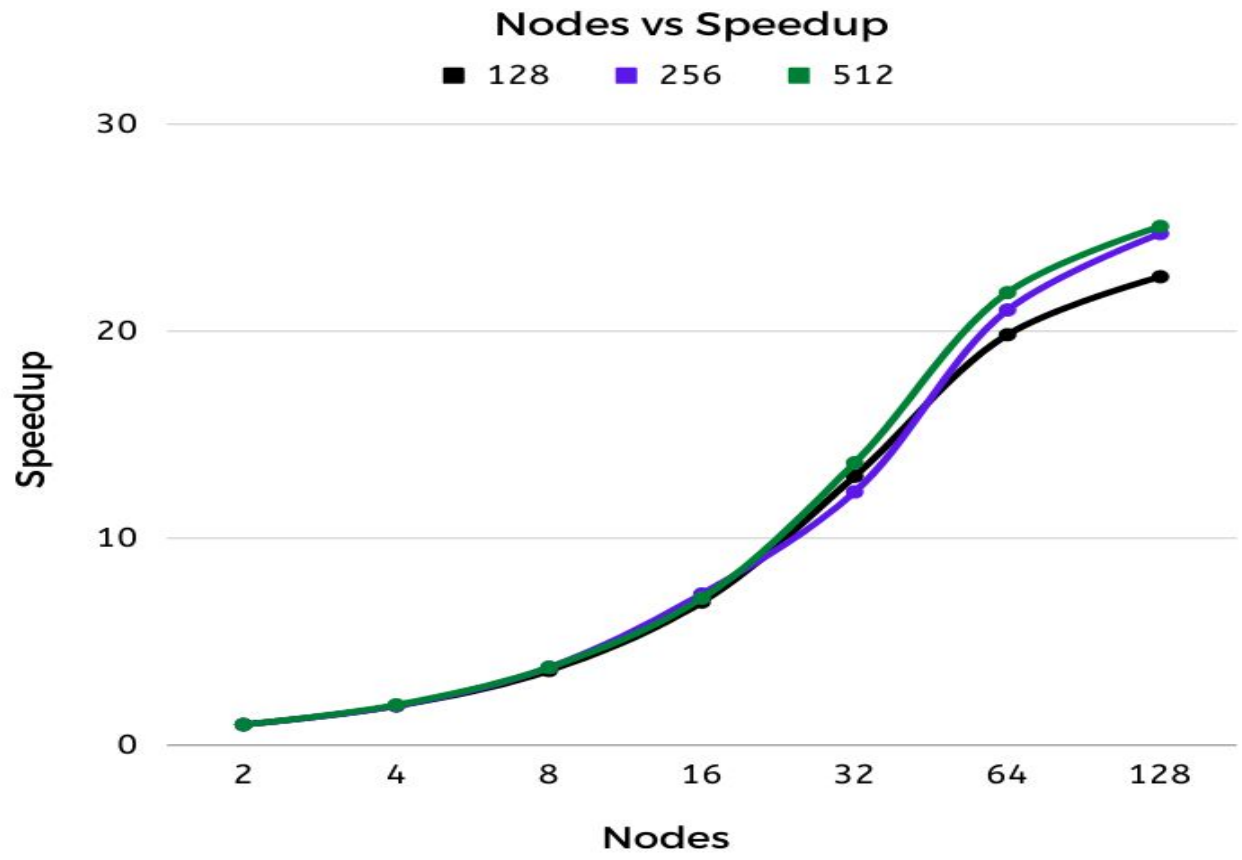
   Range - 10240

Processors	128 Cluster	256 Cluster	512 Cluster
2	314.167	255.006	250.067
4	165.232	132.319	128.38
8	87.03	67.878	66.56
16	45.372	34.89	35.187
32	24.182	22.636	18.326
64	15.838	12.124	11.433
128	13.878	10.315	9.975

3. Data 512*512, Clusters - 128, 256, 512. Range - 10240



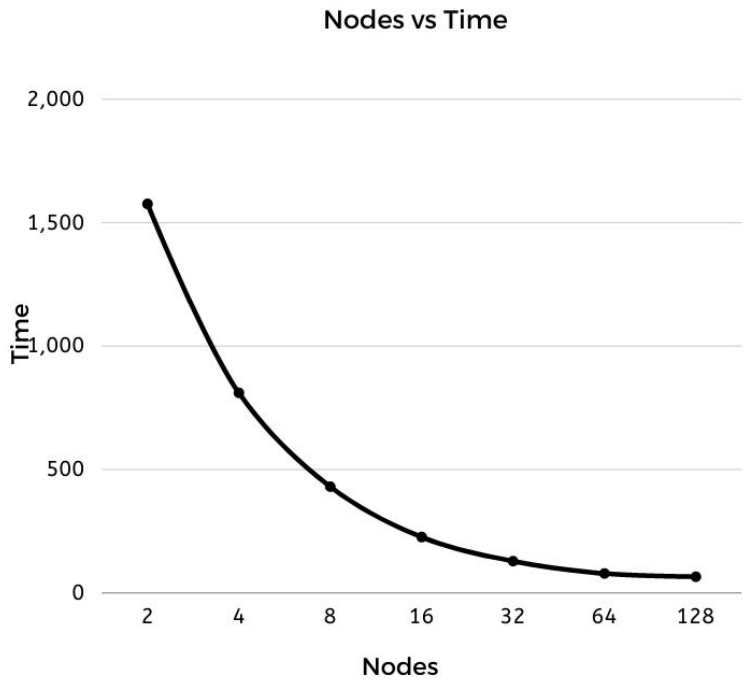
3. Data 512*512, Clusters - 128, 256, 512. Range - 10240



4. For 1024*1024 = Million data points, 512 clusters

   Range - 10240

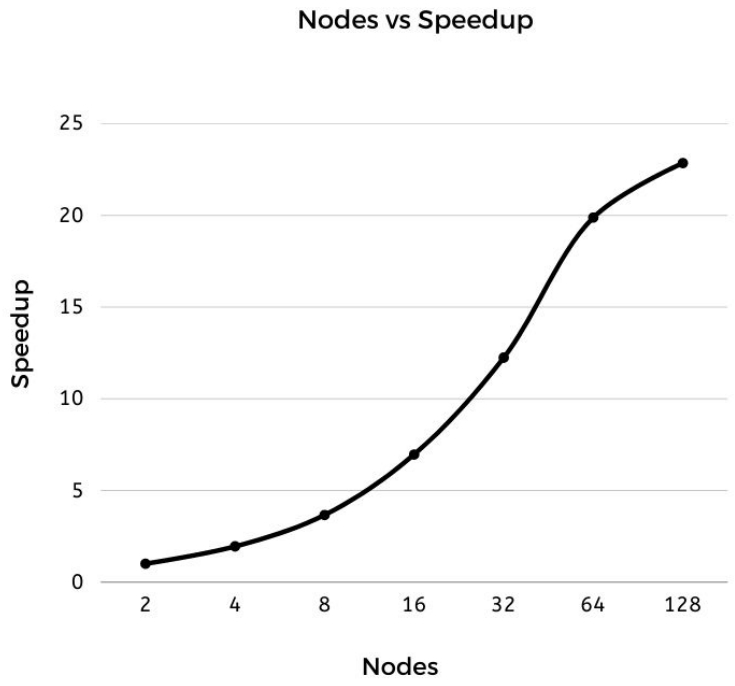
Processors	Time
2	1576.176
4	810.624
8	430.87
16	226.548
32	128.811
64	79.296
128	68.989



4. For 1024*1024 = Million data points, 512 clusters

   Range - 10240

Processors	Time
2	1576.176
4	810.624
8	430.87
16	226.548
32	128.811
64	79.296
128	68.989



Inferences

1. During above experiments, till 64 processors the speedup increases, but after 64, speedup starts to plateau.
2. As we increase the range of numbers, this effects the time as well, even if we have same number of data points with same clusters.
3. 64 processors was the optimal count for processors when we had the data in range 10240.
4. 32 processors was optimal for 1024 range of data, depending on range of data we have to decide the count of processors.
5. When we increased the data size to million, 64 was still the optimal processor count.

Challenges

1. Getting access of high count of nodes.
2. To find optimal parameters
3. Testing the algorithm
4. Performing various iterations using different parameters (Different clusters, Range of data, Size of data)

References

1. Algorithms Sequential and Parallel: A Unified Approach

   (Dr. Russ Miller, Dr. Laurence Boxer).

2. https://ubccr.freshdesk.com/support/solutions/folders/13000001591

3. Dr. Matthew Jones slides and lectures.

4. Google for MPI Related queries.

Thank You