PARALLEL K-MEANS CLUSTERING

Tanvi Tanwar
UBIT – tanvitan
UB Person No. - 50290576

University at Buffalo The State University of New York

AGENDA

- Introduction
- Algorithm
- Experiments and Observations
- Conclusion
- References

INTRODUCTION

K-Means Clustering

K-Mean Clustering

- Simplest and popular unsupervised machine learning algorithm.
- Objective is to group similar data points and discover underlying pattern.
- K in “K-Means” is fixed for number of clusters to be formed.
- K also refers to number of centroids required for data points.
- To group the data points, Euclidean Distance is used as a measurement.

K-Mean Clustering

- The algorithm works as follows:

1. First initialize k points, called means, randomly.
2. Categorize each item to its closest mean by calculating Euclidean Distance.
3. Update the mean's coordinates, which are the averages of the items categorized in that mean so far.
4. Repeat the process for a given number of iterations or till the calculated means become fixed.

ALGORITHM

Parallel K-Means Clustering

Parallel Algorithm

- Data points to be clustered are partitioned equally amongst all processors.
- Each processor reads its set of data points.
- The processor with rank 0 initializes k random centroids and broadcasts it to all other processors.
- Each and every processor then, calculates Euclidean distance of its data points with k centroids and creates k clusters of data points.
- Each processor except a processor with rank 0, shares the total sum and total length of all K clusters.
- Then, Processor with rank 0 gathers and calculates new k centroids. It broadcasts the calculated k centroids for clustering.
- This process of clustering continues for n iterations.

EXPERIMENTS

Parallel K-Means Clustering

Readings for multiple processors

- Experiments with different number of data points are performed with 2, 4, 8, 16, 32, 64, 128 and 256 processors.

```
/util/common/python/py27/anaconda-2019.03/bin/python
Launch job
Hello I am processor0
Hello I am processor1
*******old centroid**********
[[8, 2], [16, 16], [0, 12], [7, 16]]
***********final centroid*****************
[[13, 3], [14, 13], [3, 4], [4, 14]]
0.0408790111542
All Done!
~
~
```

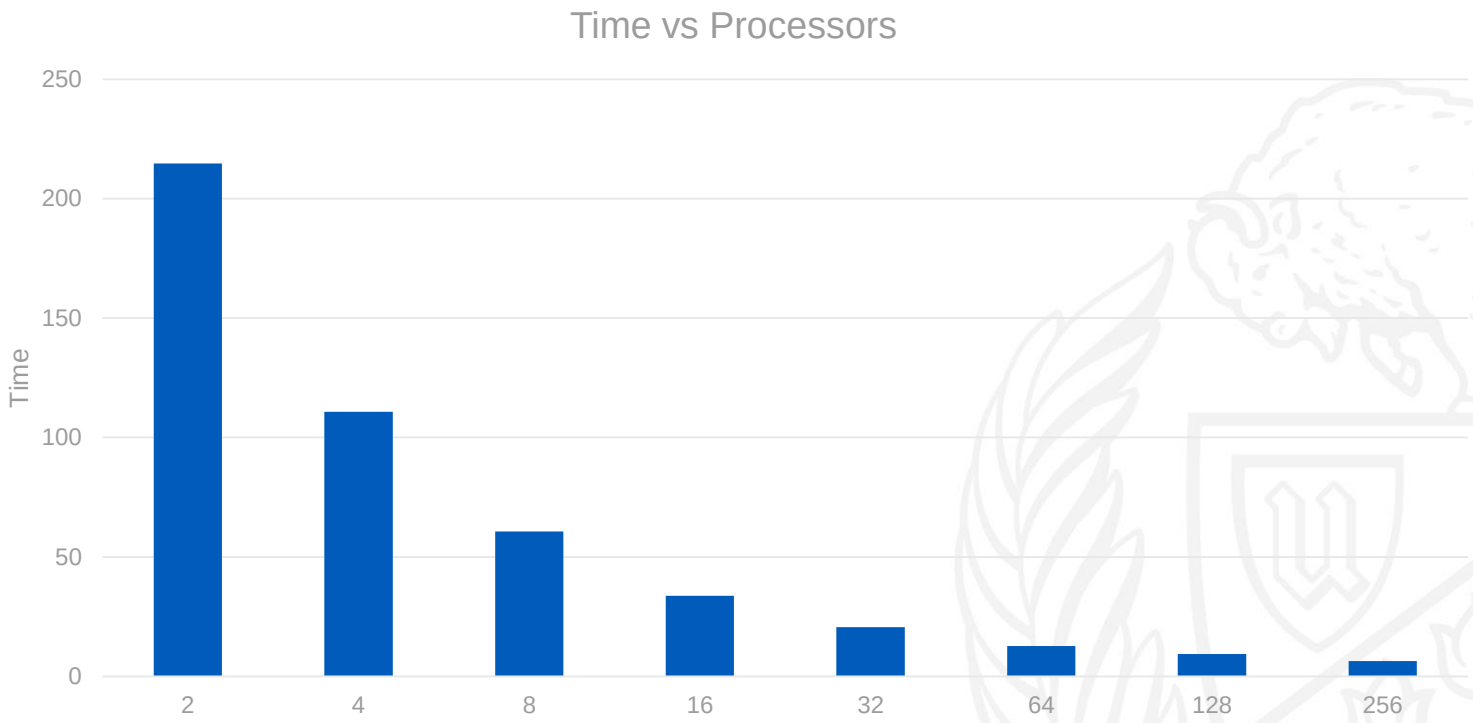
```
/util/common/python/py27/anaconda-2019.03/bin/python
Launch job
Hello I am processor0
Hello I am processor1
Hello I am processor2
Hello I am processor3
*******old centroid**********
[[9, 5], [3, 8], [5, 6], [8, 3]]
***********final centroid*****************
[[12, 7], [3, 12], [5, 6], [8, 0]]
0.0126440525055
All Done!
~
~
```

Readings Sample for data points of different size

Data Points	2 Processors	4 Processors	8 Processors	16 Processors
1600	0.0408	0.0326	0.0284	0.0295
16000	0.4893	0.2648	0.1336	0.0093
160000	4.2248	1.9636	1.0283	0.5398
1600000	40.20385	20.3621	10.6454	5.3440
16000000	224.7681	110.8064	60.6263	33.7233

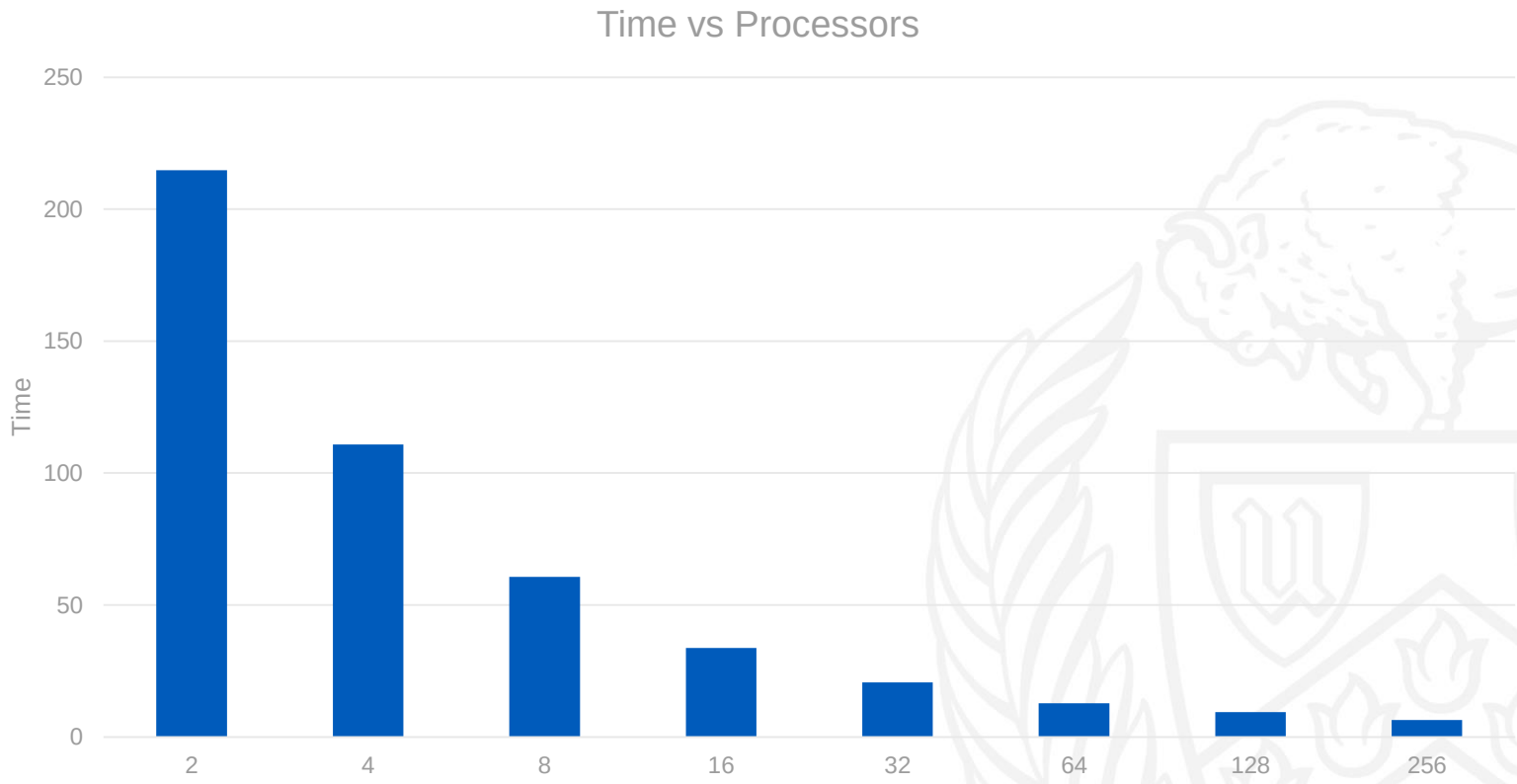
Reading for 1 Million data points

Processors	Readings
2	40.2038
4	20.3621
8	10.6454
16	5.3440
32	3.6983
64	2.4576
128	1.2648
256	0.9735


Time vs Processors
Time
250
200
150
100
50
0
2
4
8
16
32
64
128
256

Readings for 10 million data points

Processors	Readings
2	214.7681
4	110.8064
8	60.6263
16	33.7233
32	20.6362
64	12.7535
128	9.3451
256	6.4530


Time vs Processors
Time
250
200
150
100
50
0
2
4
8
16
32
64
128
256

CONCLUSIONS

Parallel K-Means Clustering

Results

- The execution time decreases as we increase the number of processors
- But the difference in execution time of multiple processors is observed only when the data size is huge.
- Increase in number of processors might lead to decrease in efficiency of parallel algorithm because of communication overhead in the algorithm.
- So, for better performance the optimal number of processors should be chosen.

REFERENCES

Parallel K-Means Clustering

References

- Mpi4Py official documentation

THANK YOU!!