Distributed Data Analytics

Lab 3

Submitted By- Abdullah Al Foysal

Id:277458

Exercise 1: Distributed K-means Clustering

For this exercise I have used my previous Lab 2 assignment tf.idf results to perform the distributed k-means clustering algorithm for the 20-newsgroup dataset. I have loaded the data files from the local file system. Each tf-idf results are of counter object type. I have not used any scikit to convert it to sparse matrix and instead continued to perform the experiment using my Counter objects. Now moving on to the exercise. I have initialized my cluster size by some k value and choose k random cluster centres as global mean using "np.random.choice()". My master node is responsible for scattering the data among all the workers and also it broadcasts this global mean across all workers. Now the data will be shared equally across all the workers and each worker would have a copy of this global mean. Now next step was to calculate Euclidean distance of each data point wrt to this global cluster centres using the formula shown in (1) and assign the data point to the cluster which has the least distance. Then I have calculated my new local cluster means by taking the average of each cluster group and have made this local mean as global mean for further iterations. The problem I have faced using this is that there was no way that my local cluster and global cluster to remain same. So, in order to make sure that my algorithm converges. I have calculated the distortion in each iteration by using the formula shown in (2) and my all individual workers would send their respective distortion to master node. I have done this using "comm.reduce". My master node will calculate the complete distortion in each iteration and compares the value with previous iterations. My total distortions values throughout the exercise were decreasing at each iteration and I have put convergence condition to stop if my total distortion value increases from current value

Euclidean distance:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \qquad \text{..... (1)}$$

Distortion calculation:

$$J(c,\mu) = \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2 \qquad \text{..... (2)}$$

Where:
μc = cluster mean
$x^{(i)}$ = each data point belonging to that cluster μc

O/p for: mpiexec –n 4 python ex_1.py (for cluster size k=2)

```
iteration=0 and total distortion=25614585.308833838
iteration=1 and total distortion=24894017.630731076
iteration=2 and total distortion=24634214.18926554
iteration=3 and total distortion=23791404.44260764
iteration=4 and total distortion=22636206.87818278
iteration=5 and total distortion=21767579.03588944
iteration=6 and total distortion=20695278.186403446
iteration=7 and total distortion=17234729.51064492
iteration=8 and total distortion=15854018.15256009
converged at iteration=9
total_parallel_execution time is=: 2181.2939608013257
converged at iteration=9
```

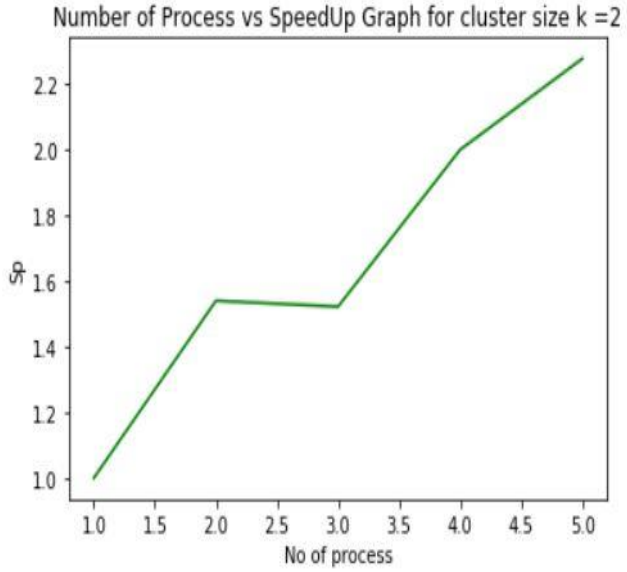
b) Speed up graph and parallel efficiency:

In this exercise, I have calculated the speedup graph and parallel efficiency graph for the 20-newsgroup data using my implemented distributed k-mean algorithm in previous exercise. Speed up graph can be calculated as follows:
Speedup: S = Ts/Tp
Where Ts = Best serial execution time
P = # of processes
Tp = Execution time on P processes
Sp = Speed up on P processes
And parallel efficiency is calculated as: Ep = Sp/P

I have used tf-idf results only for 6 categories (alt.atheism, comp.os.ms-windows.misc, comp.graphics, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x) instead of 20 categories. So I have calculated my results for just 6000 data instead of total 20000 data since calculating Ts on large dataset using one single process takes a lot of time. As my laptop can't handle calculation of such big dataset. Results are noted based on these 6000 data for cluster sizes of k = 2 with different workers p = 1,2,3, 4, 5. And results and graphs are noted as shown below:
For K = 2

Data Size	Cluster Size (k)	Serial Process Execution (Ts) In sec	No: parallel process (P)	Total parallel execution time (Tp) In sec	Speed up (Sp): Ts/Tp In sec	Parallel Efficiency(Ep): Sp/p In sec
6000	2	4967.93	1	4967.93	1	1
6000	2	4967.93	2	3227.60	1.5392	0.76
6000	2	4967.93	3	3265.47	1.5213	0.5071
6000	2	4967.93	4	2485.42	1.9988	0.4997
6000	2	4967.93	5	2183.16	2.27	0.454

```
speed_up:  [1.0, 1.5392025034081052, 1.521352209635979, 1.9988291717295266, 2.275568442074791]
```




```
parallel_effeciency:  [1.0, 0.7696012517040526, 0.5071174032119931, 0.49970729293238164, 0.45511368841495814]
```