SCALABLE K-MEANS++ ON HADOOP

Group members:

- **Ashutosh Mahesh Pednekar - 15BCE0897**
- **Shivam Gupta - 15BCE2012**
- **Umang Aggarwal - 15BCE0136**
- **Nishant Garg - 16BCE0088**

Introduction:

K-means is one of the widely used data processing algorithms. However the initialization in the algorithm is crucial for obtaining a good solution. K-means++ caters to this, by obtaining an initial set of centers that is close to the optimum solution. But k-means++ is a sequential algorithm and is inefficient over large data-set, k-passes need to be made over the entire data to find a set of good initial centers. It is important to reduce the number of passes made to get good initialization. The algorithm implemented, k-means|| obtains optimal solution after a logarithmic number of passes. It is also possible to bring down the number of passes to a constant number.

Review of literature:

K-means++ selects the first center uniformly at random from the data. Rest of the of the centers are selected with a probability proportional to their contribution to the overall error for the previous selections. The algorithm exploits the fact that a good clustering is spread out, such that while selecting a new cluster head priority is given to those nodes that are farther away from the previously selected centers. K-means++ initialization has a time complexity of O (log k) for finding the optimum solution. K-means++ is not parallelizable as selecting the ith center requires the information of previously chosen i-1 centers. So for large data-set, for which most of the processing can be done in parallel, the application needs to wait for the clustering to be completed in sequential processing.

Related works:

Ostrovsky et al. in *The effectictiveness of Lloyd-type methods for the k-means problem*, proposed an algorithm to find an initial set of clusters for Lloyd's iteration and showed that the algorithm can achieve O(1) - approximation to the optimum solution using some data separability assumptions. Arthur and Vassilvitskii developed a similar k-means method that achieved O (log k)- approximation to the optimum but with no assumptions on the data. Ailon et al. *Streaming k-means approximation*, provided streaming algorithm based on k-means++ that makes a single pass over the data, selecting O (k log k) points, achieving a constant factor approximation in expectation.

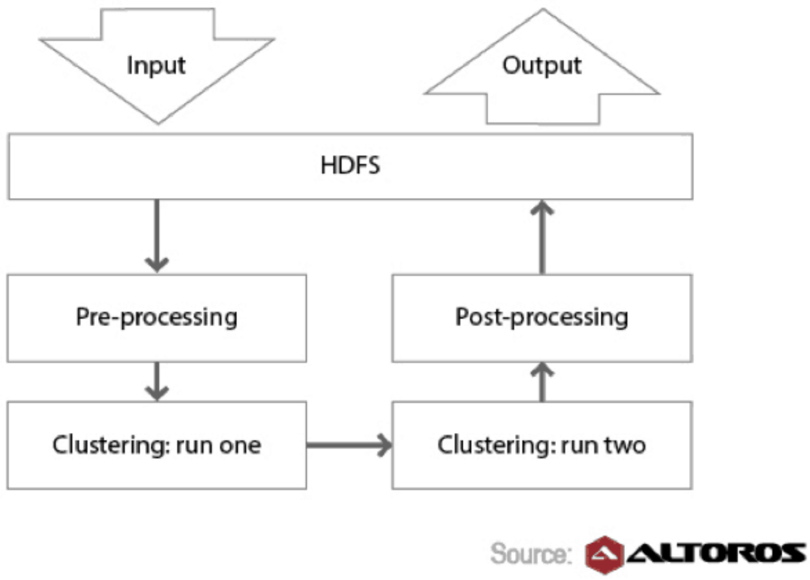
Architecture - (GPU Architecture):

GPU's are intense computational gadgets utilized as a part of designs equipment. Nowadays GPU are likewise being utilized for applications requiring high computational power which are named as General Purpose Computing on GPU (GPGPU).

executes a similar guideline. An arrangement of strings shapes a square which set up together structures a lattice. Pieces are allocated to SM for execution. SM forms one twist at any given moment where each twist is of 32 strings from a square. The capacity calls are made as bit's which release different strings to play out an undertaking in a Single Instruction Multiple Data (SIMD) design

Related work on equipment models for K-Means Clustering Many scientists have proposed distinctive equipment structures for the K-Means grouping calculation. Before 2001, just process escalated parts of the calculation, for example, Distance Measure [15], [16], were composed and actualized in equipment. This was principally because of absence of high limit reconfigurable equipment around then. These plans would run the calculation on a host processor and stream the information to the devoted equipment when performing Distance Measure calculation. A noteworthy drawback of this approach was the I/O spilling overhead, which influences the speed-execution of the outline. The creators, Leeser et al. [17], asserted to acquire around 15% speed change with a comparative plan and recommended a future half and half design, which could possibly enhance speedup by a factor of 10. The early FPGAbased outline for the K-Means was focused to applications for ghastly and hyperspectral picture

Figure 2. The platform for grouping stock trading strategies with Hadoop and k-means



Methodology:

In this work we acquire a parallel variant of the k-means++ introduction calculation and exactly exhibit its viable .The primary thought is that as opposed to testing a solitary point in each go of the k-means++ calculation, we test O(k) focuses in each round and rehash the procedure for around O(log n) rounds. Toward the finish of the calculation we are left with O(k log n) focuses that shape an answer that is inside a steady factor far from the ideal. We at that point recluster these O(k log n) focuses into k introductory places for the Lloyd's cycle. This introduction calculation, which we call k-means||, is very basic and fits simple parallel usage.

Algorithm:

To begin with, we set up some documentation that will be utilized all through the paper. Next, we exhibit the foundation on k-implies grouping and the k-means++ introduction calculation . At that point, we exhibit our parallel introduction calculation, which we call k-means|| . We introduce an instinct why k-means|| instatement can give estimation ensures the formal investigation is conceded to. At last, we examine a MapReduce acknowledgment of our calculation.

Algorithm 1 k-means++(k) initialization.
1: C   sample a point uniformly at random from X
2: while jCj < k do
3: Sample x 2 X with probability d2(x;C)_X(C)
4: C   C [fxg
5: end while

FRAMEWORKS AND SOFTWARES USED:

- Python IDLE
- Cython (C + Python)
- Hadoop
- Jupyter Notebook
- Jetbrains Pycharm