Clustering Algorithms in MapReduce: A Review

Vinod S. Bawane
Student Mobile Technology
GHRCE, Nagpur

Sandesha M. Kale
Asst. Prof. CSE Department
JDIET, Yavatmal

ABSTRACT

A MapReduce is a framework that allows processing the very big amounts of formless data in parallel across a distributed cluster of processors or individual computers. The MapReduce framework is mostly used to analyze the large amount of datasets in clustering environments. MapReduce has become a dominant parallel computing paradigm for big data. This paper describes well known strategies in MapReduce, and present comprehensive comparative algorithms in MapReduce in clustering environment.

Keywords: - Big Data, Clustering algorithm, MapReduce

1. INTRODUCTION

In recent years, MapReducing has a tremendous growth in the need for large scale data processing system. MapReducing has become a dominant parallel computing paradigm. Big data is growing term that describes any capacious amount of unstructured, semi-structured and structured data that has the potential to be mined for information.

Big data can be characterized by the excessive volume of data, the broad variety of data and the velocity at which the data required to be processed. Although big data doesn't pass on to any specific amount or quantity, the term is often used when talking about petabytes and Exabyte of data, much of which cannot be included easily. Because big data acquire in excess of time and overheads too much money to load into a conventional relational database for analysis, new advance to gathering and examining data have emerged that rely less on data schema and data attribute. As a substitute, raw data with extended metadata is aggregated in a data lake and machine learning and artificial intelligence (AI) agenda use composite algorithms to look for repeatable prototype. Big data analytics is often related with cloud computing because the analysis of large data sets in real-time needs a platform like Hadoop to gather fat data sets across a distributed cluster and MapReduce to match up, merge and process data from multiple sources.

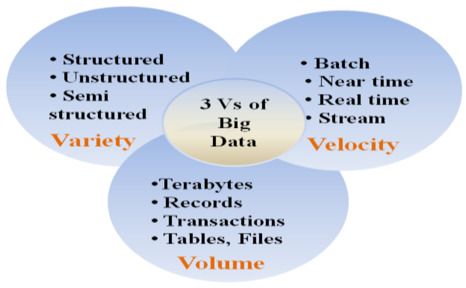

Fig.1: Big Data

In the era of petabytes of data, the processing analytical queries on massive amount of data in a scalable and reliable manner is becoming one of the most important challenges for data mining and warehousing system. To deal with the large scale data there needed a special purpose computational system for extracting valuable and knowledgeable important data. Most of the computational system extracted the data in distributed and parallel manner. Computational system also involves complex analysis based on data mining algorithms which require multiple datasets to be processed continuously.

2. OVERVIEW

In the era of large data processing, there is need for large scale data processing system. MapReduce is a programming model and used for processing and generating bulky datasets with a similar, distributed algorithm on a cluster. In MapReducing, processing can be done on data which is stored in database. MapReducing allows three main steps for processing of datasets such as build stage, map stage and reduced stage [1].

MapReduce adopts a master/slave architecture where a master node handle and observe map/reduce tasks and slave nodes process map/reduce tasks assigned by the master node, and uses a distributed file system(DFS) to manage the input and output files.

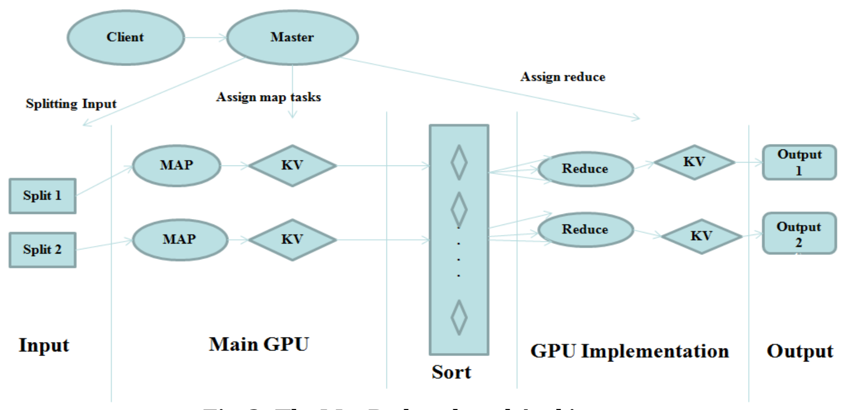

Fig. 2: The MapReduce-based Architecture

In this architecture, build stage [1] decomposes rules into sub rules by the master, which are distributed to different workers with certain strategy. Then, workers compile the received sub-rules into a net separately, which is dedicated for rule matching facts, with Rete algorithm. Map stage is also called as matching stage. In the map stage [2], each map task first parses its equivalent input split up into a set of input key-value pairs. After that it applies the map function on each one input key value pair (kv-pair) and make a set of intermediate key value pairs which are sorted and partitioned into r partitions, where r is the number of configured reduce tasks. An optional combine job can be applied on the intermediate map output to reduce its size and hence the communication expense to transfer the map output to the reducers1. In the reduce phase, each reduce task first gets its equivalent map output partitions from the map tasks and combine them. Then for each key, the reducer applies the reduce function on the values connected with that key and outputs a set of final key-value pairs [3].

Table 1 shows the example of map reduced jobs and working of its stages. Table shows the seven jobs from j1 to j7

Tabel1: Example of MapReduce jobs [2]

Id	**Job**	**<Key, Value>**
J1	**select** *a*, sum(*d*) **from** *T* **where** *a* ≥ 10 **group by** *a*	*<a, d>*
J2	**select** *a*, *b*, sum(*d*) **from** *T* **where** *b* ≤ 20 **group by** *a*, *b*	*<(a, b), d>*
J3	**select** *a*, *b*, *c*, sum(*d*) **from** *T* **where** *c* ≤ 20 **group by** *a*, *b*, *c*	*<(a, b, c), d>*
J4	**select** *a*, sum(*d*) **from** *T* **where** *b* ≤ 20 **group by** *a*	*<a, d>*
J5	**select** *b*, sum(*d*) **from** *T* **where** *a* ≥ 20 **group by** *b*	*<b, d>*
J6	**select** * **from** *T*, *R* **where** *T.a* = *R.e*	*T :<a, T.*> R:<e,R.*>*
J7	**select** * **from** *T*, *R* **where** *T.a* = *R.e* and *T.b* = *R.f*	*T :<(a, b), T.*> R:<(e, f),R.*>*

For the job *Ji*, here uses *Ki* to denote its map output key, *Ai* denote the set of attributes in *Ki*, |*Ai*| denote the number of attributes in *Ai*, *Mi* denote its map output, and *Ri* denote its reduce output. Such as, for *J2* in Table 1, *K2=(a, b)*, *A2={a, b}* and |*A2*|=2. Here make use of $Ki \leq Kj$ to denote that *Ki* is a prefix of *Kj*, and $Ki < Kj$ to represent that *Ki* is a proper prefix of *Kj* (i.e $Ki \neq Kj$). Such as, $K4 < K2$ and $K5 \nless K2$. Consider a map output *Mi* with schema *(Ai, Vi)* where *Ai* and *Vi* refers to the map output key and value elements, correspondingly. Given a set of attributes $A \subseteq Ai$, here use M^{A}_{i} to indicate the map output derived from *Mi* where its map output key attributes are projected on *A*; i.e., $M^{A}_{i} = \prod A,Vi(Mi)$. For example, $M^{\{a\}}_{2} = M4$. Think about two jobs *Ji* and *Jj* where $Aj \subseteq Ai$. here use $Mi,j \subseteq Mi$ to represent the subset of *Mi* such that $M^{Aj}_{i,j} = M^{Aj}_{i} \cap Mj$ denotes the subset of *Mj* that can be resulting from *Mi*. Furthermore, use *Mi* *Mj* to denote the (key, value-list) illustration of the map output $Mi \cap Mj$. For example, if $Mi \cap Mj = \{(k1, v1), (k1, v2), (k2, v3)\}$, then *Mi* *Mj* $= \{(k1, < v1, v2>), (k2, < v3>)\}$ [2].

3. CLUSTERING ALGORITHMS

In a clustering environment there are so many clustering algorithms are used for MapReduce system. Accordingly, the processes of different clustering algorithms can be broadly classified as partition based, hierarchical based, density based, grid based, and model based. Following figure shows an overview of clustering algorithm.

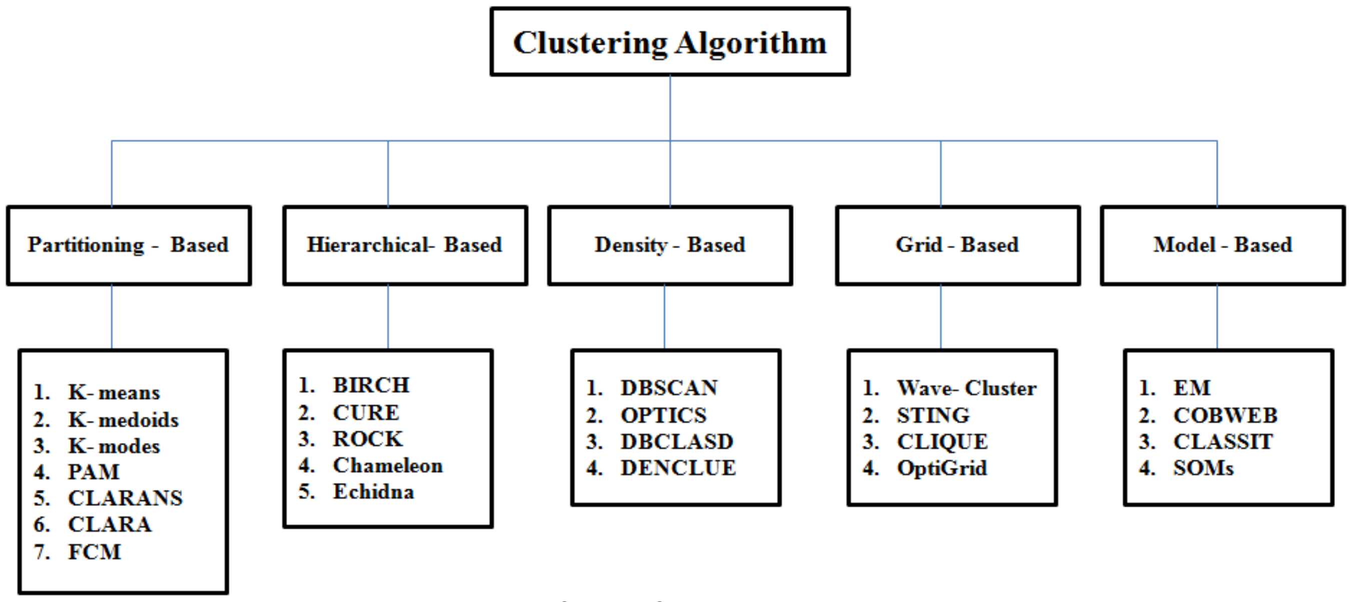

Fig. 3: Classification of clustering algorithm

This section shows the comparison between few algorithms for proving the energy efficient algorithm for the MapReduce system for large datasets in a clustering environment.

3.1 K-Means Algorithm:

K-Means Clustering [4][5] is a method used to classify semi structured or unstructured data set.

One of the most important and commonly used methods for grouping the items of a data set using K-Means Clustering is calculating the distance of the point from the chosen mean. This distance is usually the Euclidean Distance [5] though there are other such distance calculating techniques in existence. This is the most common metric for comparison of points.

3.2 DBSCAN Algorithm

DBSCAN [6] [7] is Density-Based Spatial Clustering of Applications with Noise. Itis an effective density-based clustering method. It was first proposed in 1996. Compared with other clustering methods, DBSCAN possesses several attractive properties. First, it can divide data into clusters with arbitrary shapes. For example, it can find clusters totally surrounded by another cluster. Second, DBSCAN does not require the number of the clusters a priori. Third, it is insensitive to the order of the points in the dataset. DBSCAN has two most important advantages such as [6]:

1. The sizes of the datasets are growing rapidly so that they cannot be held on a single machine any more.
2. the advantages of DBSCAN come at a cost, i.e., at a much higher computation complexity compared with other clustering methods such as K-means

Designing an efficient DBSCAN algorithm in MapReduce has following three main challenges [6] [7]:

1. Due to the simplicity of MapReduce
2. MapReduce can process text based data queries efficiently, it becomes quite clumsy when dealing with spatio-temporal data.
3. Maximum parallelism can only be achieved when the data is well balanced.

The aim of clustering algorithm is to divide mass raw data into separate groups (clusters) which are meaningful, useful, and faster accessible. Density based algorithm continue to grow the given cluster as long as the density in the neighborhood exceeds certain threshold. This algorithm is suitable for handling noise in the dataset. The following points are enumerated as the features of this algorithm [6] [7].

1. Handles clusters of arbitrary shape
2. Handle noise
3. Needs only one scan of the input dataset
4. Needs density parameters to be initialized

DBSCAN clustering is density based clustering with O (n) time complexity and O (n + s^2) space complexity. This clustering is simply opposed to the K-means clustering that does not require to specify number of clusters in data priori, it require just two parameter to perform clustering on big data or unstructured data, its arbitrary shape cluster is additional advantages to the DBSCAN. It is not deterministic and quality is depends on measure distance.

The purpose of clustering algorithm is to convert large amounts of raw data into separate clusters in order to better and quicker access. DBSCAN and K-means are two major algorithms for processing clustering problem. DBSCAN [6] [7] is a density-based clustering algorithm, which can generate a number of clusters, and also use for the distribution of spatial data. K-Means [4] [5] algorithm is based on the first of its kind, which can locate the fairly accurate class for the given value. Compared to K-means algorithm, DBSCAN does not essential to know the number of classes to be formed in advance. It can not only find free form class, but also to identify the noise points.

Class is defined as a collection contains the maximum number of data objects which density connectivity in DBSCAN algorithm.

3.2.1 Algorithm: [6] [7]

Input: dataset D, radius Eps, density threshold MinPts

1. Select point p
2. Retrieve all points density reachable p with respect to Eps and MinPts
   Where,
   Eps- Maximum radius of neighborhood
   MinPts- Minimum no. of points in an Eps neighborhood of the points
3. If p is core point, cluster is formed
4. If p is border point, no points are density reachable from p and DBSCAN visit the next point of database
5. Continue the process until all the points have been processed

Following figure shows the flowchart for DBSCAN based on MapReduce

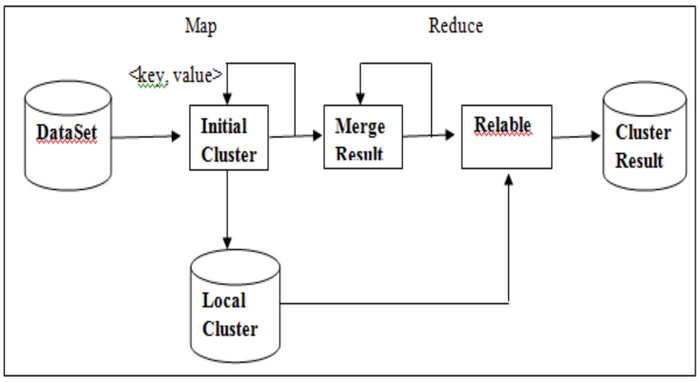

Fig. 2: DBSCAN flow based on MapReduce

(1) Input data sets are automatically divided into smaller blocks, and distributed to the processor.

(2) The Map functions parse these inputted small blocks into the format of <key,value> pairs, and then perform the DBSCAN algorithm to Initial Cluster and generate an initial class. The data object identifier oid as "key" and "value". It will generate key-value pairs like <oid, Cid> after executing the Map functions.

(3) Combiner functions classify the same value of key as a group and distribute to idle processor to perform Reduce process, when the dataset is bulky, the objects in each of the divided sub-dataset are closer, during this time the intermediate value Map functions generated will have a high repetition rate, it is the reason why add Combiner functions to lower the repetition rate.

(4) Processors execute Reduce process, doing hierarchical merging for the initial clusters generated by the Map stages and output the final result.

3.3 Naïve Bayes Algorithm:

Naive Bayes [5] is a clustering algorithm with identical complexity for both time and space, complexity is O $(m * n^2 + n * c)$. It handles real and discrete data, it fast to classify and also fast for single scan, features are not sensitive to relevant datasets. It handles stream data very well and also handles real time data as well as discrete data. The disadvantage of this clustering is that it works very slowly on small data. Assuming independence feature is also disadvantages for the clustering algorithm.

3.4 Support Vector Machine Algorithm:

Support Vector Machine [5] with complexity O $(m^3 * n)$ and O $(m^2 * n)$ for time and space is not well suited for big data. Its uses kernel trick and makes user thinking about over fitting due regularization factors. But it has time speed and size limitation and choice of kernel is biggest limitation of the SVM. It is an approximation to bound on the test error rate. Discrete data present is an another problem for clustering algorithm.

Following table 2 shows the comparison of complexity of the clustering algorithm:

Table 2: Comparison between time and space complexity

Sr. No.	Algorithm	Time Complexity	Space Complexity
1	K- means	O(m*n*c)	$O(n+s^2)$
2	Naïve Bayes	$O(m*n^2+n*c)$	$O(m*n^2+n*c)$
3	SVM	$O(m^3*n)$	$O(m^2*n)$
4	DBSCAN	O(n)	$O(n+s^2)$

4. CONCLUSION

In this paper, here studied about Map-Join-Reduce system that extends and improves the MapReduce runtime system to efficiently process complex data analytical tasks on large clusters. The novelty of Map-Join-Reduce is that it introduces a filtering-join-aggregation programming model. This programming model allows users to specify data analytical tasks that require joining multiple data sets for aggregate computation with a relatively simple interface offering three functions: map(), join(), and reduce() and compare some clustering algorithm like K-means and DBSCAN.

5. REFERENCE

[1] Bin Cao, Jianwei Yin, Qi Zhang, Yanming Ye, "*A MapReduce-based architecture for rule matching in production system*", 2nd IEEE International Conference on Cloud Computing Technology and Science ,2010.

[2] Guoping Wang and CheeYong Chan, *"MultiQuery Optimization in MapReduce Framework"*, 40th International Conference on Very Large Data Bases, September 2014.

[3] Thomas Wirtz and Rong Ge, *"Improving MapReduce Energy Efficiency for Computation Intensive Workloads"*, Green Computing Conference and Workshops (IGCC), 2011 International.

[4] Prajesh P Anchalia, Anjan K Koundinya, Srinath N K, *"MapReduce Design of K-Means Clustering Algorithm"*, 2013 IEEE.

[5] Cheng T. Chu, Sang K. Kim, Yi A. Lin, Yuanyuan Yu, Gary R. Bradski, Andrew Y. Ng, and Kunle Olukotun, *"Map-Reduce for Machine Learning on Multicore"*, NIPS, page 281--288. MIT Press, 2006.

[6] Yaobin He, Haoyu Tan, Wuman Luo, Huajian Mao, Di Ma, Shengzhong Feng, Jianping Fan, *"MR-DBSCAN: An Efficient Parallel Density-based Clustering Algorithm using MapReduce"*, 2011 IEEE 17th International Conference on Parallel and Distributed Systems.

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, *"A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise"*, Published in Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96).