IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 4, April 2015.
www.ijiset.com

ISSN 2348 – 7968

Comparision of k-Means and k-Medoids Clustering Algorithms for Big Data Using MapReduce Techniques

Subhashree K[1], Prakash P S[2]

[1] Student, Kongu Engineering College,
Perundurai, Erode

[2] Assistant Professor, Kongu Engineering College,
Perundurai, Erode

Abstract

Huge amounts of structured and unstructured are being collected from various sources. These data called as Big Data which are difficult to handle by a single machine require the work to be distributed across many computers. Hadoop is a distributed framework which uses MapReduce programming model to process the data in a distributed manner. Clustering analysis is one of the important research areas in the field of data mining. Clustering is the most commonly used data processing algorithms. Clustering is a division of data into different groups. Data are grouped in such a way that data of the same group are similar and the data in other groups are dissimilar. Clustering aims in minimizing intra-class similarity and in maximizing interclass dissimilarity. k-Means is the popular clustering algorithm because of its simplicity. Nowadays, as the volume of data increases, researchers started to use MapReduce which is a parallel processing framework to get high performance. But, MapReduce is unsuitable for iterated algorithms owing to repeated times of restarting jobs, big data reading and shuffling. To overcome this problem, a novel processing model in MapReduce called optimized k-means clustering method which uses the methods of probability sampling and clustering, merging using two algorithms called weight based merge clustering and distribution based merge clustering is introduced to eliminate the iteration dependence and obtain high performance. Also the algorithms are compared with the k-medoids clustering algorithms

Keywords: k-Means, MapReduce, k-Medoids

1. Introduction

A massive improvement in the computer technology leads to the development of large new devices. Computing devices have several uses and are necessary for business, scientists, governments and engineers. The common thing among all computing devices is the potential to generate data.

The popularity of the Internet along with a sharp increase in the network bandwidth available to users has resulted in the generation of huge amounts of data.

The amount of data generated can often be too large for a single computer to process in a reasonable amount of time. The data may also be too big to store on a single machine. To reduce the time it takes to process the data, and to store the data, it is necessary to write programs that can execute on two or more computers and distribute the workload among them.

To address the above issues, Google developed the Google File System (GFS), a distributed file system architecture model for large-scale data processing and created the MapReduce programming model. The MapReduce programming model is a programming abstraction that hides the underlying complexity of distributed data processing. Therefore the difficulty of parallelizing computation, distribute data and handle faults no longer become an issue. This is because the MapReduce framework handles all these details internally and removes the responsibility of having to deal with these complexities from the programmer.

ISSN 2348 – 7968

A novel processing model in MapReduce is proposed in this paper, in which sampling is used to eliminate the iteration dependence of k-means and to obtain high performance. In particular, the contributions of the paper are summarized as follows. First, a novel method to optimize k-means clustering algorithms using MapReduce is presented, which eliminates the dependence of iteration and reduces the computation cost of algorithms. The implementation defines the mapper and reducer jobs and requires no modifications to the MapReduce framework. Second, two sample merging strategies for the processing model is proposed and extensive experiments are conducted to study the effect of various parameters using a real dataset. The results show that the proposed methods are efficient and scalable.

2. Related Work

Davidson.I et al [2], proposed an algorithm for speeding up k-means clustering with bootstrap averaging. k-Means is time consuming as it converges to a local optimum of its loss function (the distortion) and the solution converged to is particularly sensitive to the initial starting positions. As a result its typical use in practice involves applying the algorithm from many randomly chosen initial starting positions. To overcome this disadvantage bootstrap averaging is used. Bootstrap averaging builds multiple models by creating small bootstrap samples of the training set and building a single model from each, similar cluster centers are then averaged to produce a single model that contains k clusters. The speedup is because the computational complexity of k-means is linear with respect to the number of data points. The number of iterations of the algorithm until convergence is proportional to the size of the data set. This approach yields a speedup for two reasons. First, less data are clustered and second because the k-means algorithm converges more quickly for smaller data sets than larger data sets from the same source. It is important to note that we do not need to re-start our algorithm many times for each bootstrap sample.

Fahim.A.M et al [3], proposed an idea that makes k-means more efficient, especially for dataset containing large number of clusters. In each iteration, the k-means algorithm computes the distances between data point and all centers, this is computationally very expensive for huge datasets. For each data point, we can keep the distance to the nearest cluster. At the next iteration, we compute the distance to the previous nearest cluster. At the next iteration, distance to the previous nearest cluster is computed. If the new distance is less than or equal to the previous distance, the point stays in its cluster and there is no need to compute its distances to the cluster centers. This saves the time required to compute distances to k-1 cluster centers. Two functions are written in the proposed method. The first function is the basic function of the k-means algorithm, that finds the nearest center for each data point.. Second function is distance_new(). Here an efficient implementation method for implementing the k-means method is proposed. The algorithm produces the same clustering results as that of the k-means algorithm, and has significantly superior performance than the k-means algorithm.

David Arthur et al [1], dealt with the speed and accuracy of k-means algorithm. The proposed algorithm rectifies the problem by augmenting k-means with a simple, randomized seeding technique, obtain an algorithm that is O(log k)-competitive with the optimal clustering.

Dean J et al [4], discussed about data processing on large clusters using MapReduce. MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Programs

ISSN 2348 – 7968

written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

Kenn Slagter et al [5], proposed an efficient partitioning technique to overcome the problem of skew. The proposed algorithm improves load balancing as well as reduces the memory requirements. Nodes which runs slow degrade the performance of Mapreduce job. To overcome this problem, a technique called micro partitioning is used that divide the tasks into smaller tasks greater than the number of reducers and are assigned to reducers in “just-in-time” fashion. Running many small tasks lessens the impact of stragglers, since work that is scheduled on slow nodes is small which can be performed by other idle workers.

Juntao wang et al [6], developed density-based detection methods based on characteristics of noise data where the discovery and processing steps of the noise data are added to the original algorithm. Preprocessing the data improves the clustering result significantly and the impact of noise data on k-means algorithm is decreased. First a pre treatment is made with the data to be clustered to remove the outliers using outlier detection method based on LOF, so that the outliers cannot participate in the calculation of the initial cluster centers, and excluded the interference of outliers in the search for the next cluster center point. We secondly apply fast global k-means clustering algorithm on the new data set which is generated previously. Fast global k-means clustering algorithm is an improved global k-means clustering algorithm by Aristidis Likas.

Min Chen et al [7], reviewed the background and state-of-the-art of big data. The general background of big data and review related technologies, such as could computing, Internet of Things, data centers and Hadoop were introduced. Then the four phases of the value chain of big data, i.e., data generation, data acquisition, data storage, and data analysis are focussed. For each phase, introduction of the general background is studied, the technical challenges are discussed, and the latest advances are reviewed. Finally several representative applications of big data, including enterprise management, Internet of Things, online social networks, medical applications, collective intelligence, and smart grid are examined. These discussions aim to provide a comprehensive overview and big-picture to readers of this exciting area.

Cluster analysis has undergone vigorous development in the recent years. There are several types of clustering hierarchical clustering, density-based clustering, grid based clustering, model-based clustering and partitional clustering. Each clustering type has its own style and optimization methods. Regarding the expansion of the data size and the limitation of a single machine, a natural solution is to consider parallelism in a distributed computational environment. MapReduce is a programming framework for processing large-scale datasets by exploiting the parallelism among a cluster of computing nodes. MapReduce gains popularity for its simplicity, flexibility, fault tolerance and scalability. All the big data processing problems cannot be made efficient by parallelism because partitional clustering algorithm requires exponentially much iteration. Also job

ISSN 2348 – 7968

exponential creation time and time of big data shuffling are hard to swallow especially when data size is huge, so just parallelism is not enough, only by eliminating the partitional clustering algorithms dependence on the iteration, high performance can be achieved.

3. Proposed Method

The proposed work is divided into three modules

- Traditional k-means clustering in MapReduce
- Optimized clustering in MapReduce
- Weight-based Merge Clustering (WMC)
- Distribution-based Merge Clustering (DMC)
- k-Medoid Clustering

3.1 Traditional k-means clustering in MapReduce

MapReduce is a programming model for processing and generating large data sets with a parallel, distributed algorithm on a cluster. This has two steps namely, map and reduce. In map step, problem is divided to sub-problems ie this step process smaller problems. The results are passed to reduce task. In reduce step, the reduce task answers of sub-problems are collected and combined to form the final answer of the original problem.

3.2. Optimized clustering in MapReduce

Set *D* as the original dataset, *k* as the number of the clusters that are required, *i* and *j* are all integers, while i from 1 to 2*k* and *j* from 1 to *k*, c_i is the ith center, C_i is the collection of points belong to center c_i. MapReduce job will read disk where the large-scale datasets stored repeatedly and the large-scale data will be shuffled over the whole clusters each time. Therefore the I/O cost and network costs are very expensive. To estimate the iteration, sampling is done to get some subsets of the big data. By processing these subsets, the center sets are obtained which can be used to cluster original datasets. This method involves probability sampling and sample clustering and merging.

1. Probability Sampling

The first MapReduce job is sample selection. Sampling is performed on the original large dataset using *k* and probability $p_x = \frac{1}{\varepsilon^2 N}$, where $\varepsilon \in (0,1)$ and controls the size of the sample while N is the number of points, and then we generate some samples small enough to be processed in a single machine.

2. Sample Clustering And Merging

Small-sized sample files are obtained after sampling. In the second job, mappers cluster each sample file using *k* and get the centers for those samples. We shuffle these centers to only one reducer and merge them into k final centers, which are used to obtain final clustering result

3.3. Weight-based Merge Clustering (WMC)

Given a situation S in which two clusters are to be merged, *A* and *B*, whose centers are point c_a and point c_b, respectively, and *A* has m points while *B* has n points. If simple merging strategy is used, the new center of the merging result is the midpoint of c_a and c_b, but this is inaccurate, because, if $m > n$, the real new center of *A* and *B* will be more closer to the range of *A* and c_a, according to this theorem, WMC strategy is introduced. In the second MapReduce job, after sample clustering in each mapper, the number of points assigned to each center points is collected, which will be used to weight the centers.

3.4. Distribution-based Merge Clustering (DMC)

With rational sampling, each sample file can represent the whole dataset very well. Meanwhile, the clustering result of each map is global optimal by carefully seeding. So we come up with a merge idea to separate the $2k^2$ centers into k groups, each group has $2k$ centers and consists of one and only one center from each sample clustering results. Here we first select an intermediate clustering result randomly whose *k* centers

ISSN 2348 – 7968

are recognized as the first member of *k* different groups, and then we choose members from the rest of intermediate clustering results for each group according to the distances between centers

3.5. k-Medoids Clustering

The k-means method uses centroid to represent the cluster and it is sensitive to outliers. This means, a data object with an extremely large value may disrupt the distribution of data. k-medoids method overcomes this problem by using medoids to represent the cluster rather than centroid. A medoid is the most centrally located data object in a cluster. Here, k data objects are selected randomly as medoids to represent k cluster and remaining all data objects are placed in a cluster having medoid nearest (or most similar) to that data object. After processing all data objects, new medoid is determined which can represent cluster in a better way and the entire process is repeated. Again all data objects are bound to the clusters based on the new medoids. In each iteration, medoids change their location step by step. Or in other words, medoids move in each iteration. This process is continued until no any medoid move. As a result, *k* clusters are found representing a set of n data objects. An algorithm for this method is given below.

4. Result Analysis

The dataset considered is from real-world settings and are publicly available from the website, https://archive.ics.uci.edu. The Individual household electric power consumption Data Set (House) consists of 4,296,075,259 records in 9 dimensions. Performance offered by k-means clustering, optimized k-means clustering weighted-based Merge Clustering, Distribution based merge clustering and k-medoid clustering methods are analysed and compared. The performance is evaluated by the clustering validation parameters such as intra cluster distance, inter cluster distance and time complexity rate. Based on the comparison and the results from experiments shows that the proposed approach works better than the existing system.

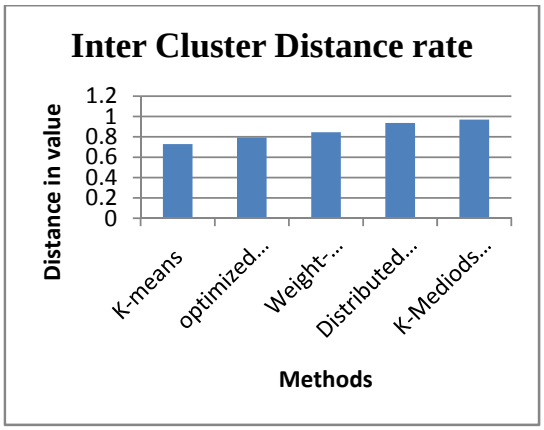

Fig.1.Inter Cluster Distance

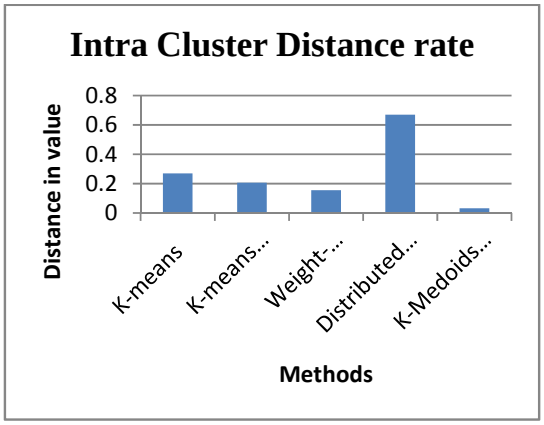

Fig.2. Intra Cluster Distance

5. Conclusion

Iteration of k-means algorithm was the important factor which affected the performance of clustering and hence a novel efficient parallel clustering model was proposed. Experimental results on large real-world dataset demonstrate that the proposed optimized algorithm is efficient and performs better compared to the existing algorithms. Clustering validation shows that the quality of our clustering methods is improved compared to k-means. From experimental results, it is identified that proposed system is well effective than existing system in terms of effectiveness of the system

References

ISSN 2348 – 7968

[1] Arthur, David, and Sergei Vassilvitskii (2007), 'K-Means++: The Advantages of careful seeding', Proc. 18th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA), Vol. 80, No. 3, pp.1027-1035

[2]Davidson I, Satyanarayana A (2003), 'Speeding up k-means clustering by bootstrap averaging', IEEE data mining workshop on clustering large data sets, Vol. 25, No. 2, pp. 179-183

[3]Fahim, A. M., A. M. Salem, F. A. Torkey, and M. A. Ramadan (2006), 'An efficient enhanced k-means clustering algorithm', Journal of Zhejiang University Science, Vol. 7 No. 10 pp.1626-1633.

[4]Dean, Jeffrey, and Sanjay Ghemawat (2008), 'MapReduce: simplified data processing on large clusters', Communications of the ACM, Vol. 51, No. 1 pp. 107-113.

[5]Slagter, Kenn, Ching-Hsien Hsu, Yeh-Ching Chung, and Daqiang Zhang (2013), 'An improved partitioning mechanism for optimizing massive data analysis using MapReduce', The Journal of Supercomputing, Vol.66 No.1 pp. 539-555.

[6]Wang, Juntao, and Xiaolong Su (2011), 'An improved K-Means clustering algorithm', In proc IEEE 3rd International Conf on Communication Software and Networks (ICCSN), Vol. 30, No.7, pp. 44-46.

[7]Chen, Min, Shiwen Mao, and Yunhao Liu (2014), 'Big Data: A Survey', Mobile Networks and Applications, Vol.19 No. 2 pp.171-209