International Journal of Advance Research and Innovation
Vol. 3(1), Jan-Mar 2015, pp. 52-56
Doi: 10.51976/ijari.311509

Article Info
Received: 04 Fab 2015 | Revised Submission: 20 Feb 2015 | Accepted: 28 Feb 2015 | Available Online: 15 Mar 2015

Enhanced K-Means++ Clustering for Big Data with Mapreduce

B. Natarajan and P. Chellammal***

ABSTRACT

Clustering big data using data mining algorithms is a modern approach, used in various science and medical fields. k-means clustering algorithm is a good approach for clustering, but choosing initial centers and provides less accuracy guarantees. The enhanced k-means approach called k-means++ chooses one center uniformly at random provides better functionality, but fails to handle data of larger volume in distributed environment. The mapreduce k-means++ method handles k-means++ algorithm by enhancing it in mapper and reducer phases, also reduces the no of iterations required to obtain k centers. in which the k-means++ initialization algorithm is executed in the mapper phase and the weighted k-means++ initialization algorithm is run in the reducer phase. it reduces huge amount of communication and i/o costs. the proposed mapreduce k-means++ method obtains (α^2) approximation to the optimal solution of k-means.

Keywords: *VANET; MANET; ZRP; LAR; IDM; Vanet Mobi Sim; ns2.*

1.0 Introduction

Clustering is one of the most fundamental tasks in exploratory data analysis that groups similar points in an unsupervised fashion, applied in many areas of computer science and related fields, such as data mining, pattern recognition and machine learning [1]. K-means is widely known one of the most popular clustering algorithms owing to its simplicity and efficiency. However, k-means is iterative by nature. At each iteration of a k-means process, new centroid must be computed based on the previous one.

Distributing k-means algorithm needs much iteration to converge is costly as a global state must be reconstructed and propagated to all nodes. [2]. A hierarchical initialization approach is to treat the clustering problem as a weighted clustering problem which finds better initial cluster centers based on the hierarchical approach also requires less iteration time compared with existing approaches and has better performance in terms of convergence speed and ability to reduce the impact of noises. [3].

The K-means++ [3] algorithm which consists of the initialization step and K-means step. In the initialization step, except that the first center is chosen uniformly random, each subsequent center is orderly chosen according to its squared distance from the closet center already chosen.

More importantly, K-means++ has a provable approximation guarantee to the optimal solution. However, fc-means++ becomes inefficient as the size of data increases (like Terabyte or Petabyte) means that there are a large number of clusters, leading to a huge number of distance computations. So, fc-means++ initialization becomes inefficient. Even though scalable fc-means++ presented in [4] chooses more than one centers in each pass and is proven as a good approximation of the original K-means, it still needs too many passes in practice, which incurs huge communication and I/O costs.

Map Reduce [7] is considered to be an efficient tool in situations where the amount of data is prohibitively large.

However MapReduce-based systems are still inefficient, to generate k centers, the MapReduce implementation of k-means++ initialization needs k rounds and 2K MapReduce jobs. In addition, a large number of data need to be transferred between multiple machines.

MapReduce K-means++ initialization algorithm in very large data situation and develops it with MapReduce. The major research challenges addressed are: (1) how to efficiently implement the

**Corresponding Author: Department of Computer Science, J. J. College of Engineering and Technology, Trichy, Tamil Nadu, India (E-mail: rec.natarajan@gmail.com;)*
***Department of Computer Science, J. J. College of Engineering and Technology, Trichy, Tamil Nadu, India*

K-means++ initialization algorithm with MapReduce. The main idea behind this method is that, instead of using 2K MapReduce jobs to choose k centers, this method uses only one MapReduce job.

Both Mapper phase and Reducer phase in our method execute the K-means++ initialization algorithm. The Mapper phase runs the standard K-means++ initialization algorithm and the weighted K-means++ initialization algorithm is executed on the Reducer phase. (2) Although K-means++ is (a) approximation to the optimal fe-means, this method is (a^2) approximation to the optimal of fe-means. The major contributions of this paper are: An efficient MapReduce implementation of fe-means++ initialization which uses only one MapReduce job to choose fe centers, avoiding multiple rounds of MapReduce jobs on many machines and thus reducing the communication and I/O costs significantly. To reduce the expensive distance computation of the proposed method, this method also uses a pruning strategy can dramatically reduce the redundant distance computation, indicate that our MapReduce fe-means++ algorithm is much efficient and has a good approximation.

2.0 Related Work

This section briefly describes the use of Map-Reduce as well as distributed and parallel frameworks dedicated to grid and cloud computations such as Hadoop.

2.1 Map-reduce

The Map-Reduce framework is originally by Google, and it is a programming model for processing extremely large datasets. It exploits data independence to do automatic distributed parallelism. The developer has tasked with implementing the Map and the Reduce functions. The input data is distributed in blocks to the participating machines using the distributed Google file system GFS [6]. When a job is launched, the system automatically spawns as many Map functions as there are data blocks to process. Each Mapper reads the data iteratively as a key/value pair record, processes it and, if necessary, outputs key/value pair bound for a Reduce function. All records with the same key go to the same Reduce task. The framework thus includes a copy-merge-sort data shuffle step, where data from several mappers gets directed to specific reducers depending on theirkey. Once enough data is locally available to reducers, they process the records and produce the final output. The Map-Reduce run-time environment transparently handles the partitioning of the input data, schedules the execution of tasks across the machines and manages the communications between processing nodes when sending/receiving the records to process. The run-time environment also deals with node failures and restarts aborted tasks on nodes, possibly on replicated data in case of unavailability.

The frame work uses as little network bandwidth as possible by processing data where it resides or at the nearest available node, paying attention to the network topology and minimizing reading over machine-rack boundaries.

2. Hadoop and HDFS

The Map-Reduce programming model has been implemented by the open-source community through the Hadoop project. Maintained by the Apache Foundation and supported by Yahoo!, Hadoop has rapidly gained popularity in the area of distributed data-intensive computing. The core of Hadoop consists of the Map-Reduce implementation and the Hadoop Distributed File System (HDFS).

HDFS was built with the purpose of providing storage for huge files with streaming data access patterns, while running on clusters of commodity hardware. HDFS implements concepts commonly used by distributed file systems: data is organized into files and directories, a file is split into fixed size blocks that are distributed across the cluster nodes.

The blocks are called chunks and are usually of 64 MB in size (this parameter specifying the chunk size is configurable). The architecture of HDFS consists of several data nodes storing the data chunks and a centralized name node responsible for keeping the file metadata and the chunk location. HDFS handles failures through chunk-level replication (default 3 replicas). When distributing the replicas to the data nodes, HDFS employs a rack-aware policy: the first replica is stored on a data node in the same rack, and the second replica is shipped to a data node belonging to a different rack (randomly chosen).

2.3 Scalable k-means++

In [4], Bahman Bahmani et al., have compared k-means, k-means++ proper initialization methods. A proper initialization of k-means is crucial

for obtaining a good final solution. A major downside of the k-means++ is its inherent sequential nature, which limits its applicability to massive data: it takes passes over the data to find a good initial set of centers. Scalable k-means++ show how to drastically reduce the number of passes needed to obtain, in parallel, a good initialization.

This is unlike prevailing efforts on parallelizing k-means that have mostly focused on the post-initialization phases of k-means. Scalable k-means++ obtains a nearly optimal solution after a logarithmic number of passes, and then show that in practice a constant number of passes suffices.

The main idea is that instead of sampling a single point in each pass of the k-means++ algorithm, sample O(k) points in each round and repeat the process for approximately 0(log n) rounds. At the end of the algorithm they have left with 0(k log n) points that form a solution that is within a constant factor away from the optimum.

Then recluster these 0(k log n) points into k initial centers for the Lloyd's [8] iteration.

However, the analysis of the algorithm turns out to be highly non-trivial, requiring new insights, and is quite different from the analysis of k-means++. This algorithm needs too many MapReduce jobs.

2.4 Hierarchical initialization approach for k-means clustering

In [3], Lu et al., have implemented a hierarchical initialization approach to the K-Means clustering problem. This method treats the clustering problem as a weighted clustering problem so as to find better initial cluster centers based on the hierarchical approach also it requires less Iteration time compared with existing approaches and has better performance in terms of convergence speed and ability to reduce the impact of noises.

The core of this method is as follows: averaging is used as a sampling method so as to reduce data hierarchically, then clustering is performed on the reduced data, meanwhile, the clustering problem is treated as a weighted clustering one. Based on this idea, better initial cluster centers can be found, and the efficiency can be improved by hierarchical clustering due to clustering on the reduced data.

The only requirement for this method is that the Euclidian distance is used as the distance metric which is reasonable for most of applications. This method can provide better initial cluster centers and generally faster than performing clustering using the standard K-Means algorithm with random initialization.

The reason for this is that better initial cluster centers are found which speed up the final clustering procedure, and reduce the number of iterations. This algorithm is also suitable for clustering of high dimensional data and can provide cluster centers with different accuracy requirements. Also, this Also, this algorithm has the ability to reduce the impact of noises.

2.5 K-means++: the advantages of careful seeding

In [3], David Arthur and Sergei Vassilvitskii solved the difficulties present in the k-means like proper initialization and no approximation guarantees. The k-means++ algorithm which works as follows, the k-means algorithm is a simple and fast algorithm for this problem. 1. Arbitrarily choose an initial k centers

C= {cl,c2, ,ck}.

2. For each i $\in$ {1, . . . , k}, set the cluster Q to be the set of points in X that are closer to Cjthan they are to Cj for all j * i.
3. For each i $\in$ = {1 . . . k}, set q to be the center of mass of all points in

$$C_i' : c_i = \frac{1}{|C_i|} \sum_{x \in C_i} X$$

4. Repeat Steps 2 and 3 until C no longer changes.

It is standard practice to choose the initial centers uniformly at random from X. For Step 2, ties may be broken arbitrarily, as long as the method is consistent. *k-mems++* algorithm which consists of the initialization step and *k*-means step.

In the initialization step, except that the first center is chosen randomly, each subsequent center is orderly chosen according to its squared distance from the closet center already chosen.

More importantly, *k-means++* has a provable approximation guarantee to the optimal solution.

However, *K*-means++ becomes inefficient as the size of data increases.

Large data (like Terabyte or Petabyte) means that there are a large number of clusters, leading to a huge number of distance computations. So, *k-mems++* initialization becomes inefficient and even impossible to process large data.

2.6 Parallel *k*-means clustering based on mapreduce

In [7], Weizhong Zhao et al., have discussed the enlarging volumes of information emerging by the progress of technology, makes clustering of very large scale of data a challenging task.

In order to deal with the problem they have proposed a parallel *K*-means clustering algorithm based on MapReduce, which is a simple yet powerful parallel programming technique, *K*-means algorithm in MapReduce framework is implemented by Hadoop to make the clustering method applicable to large scale data.

By applying proper <key, value>pairs, their proposed algorithm can be parallel executed effectively. Parallel K-Means algorithm needs one kind of MapReduce job.

The map function performs the procedure of assigning each sample to the closest center while the reduce function performs the procedure of updating the new centers. In order to decrease the cost of network communication, a combiner function is developed to deal with partial combination of the intermediate values with the same key within the same map task.

2.7 DisCo: distributed co-clustering with map-reduce.

In [5], Spiros Papadimitriou et al., propose the Distributed Co-clustering (DisCo) framework, which introduces practical approaches for distributed data preprocessing, and co-clustering.

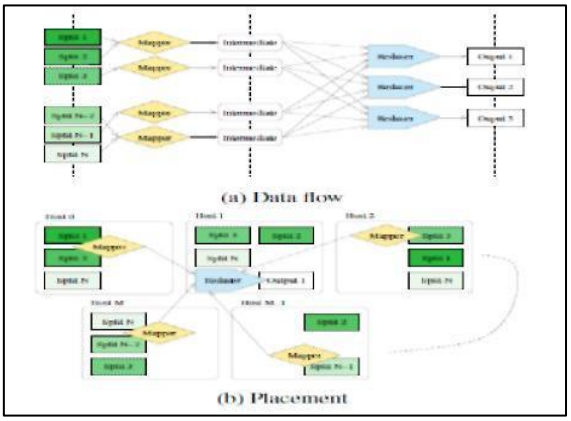
They have developed DisCousing Hadoop, an open source Map-Reduce implementation. DisCocan scale well and efficiently process and analyze extremely large datasets (up to several hundreds of gigabytes) on commodity hardware.

Comprehensive Distributed Co-clustering (DisCo) solution from the raw data to the end clusters.

DisCo using Hadoop an open source package which includes a freely available implementation of MapReduce and has been widely embraced by both commercial and academic worlds.

DisCo is a scalable framework underwhich various co-clustering algorithms can be implemented. Since both data pre-processing (i.e., graph extraction) and co-clustering components need efficient sequential scans over the entire data set, only need to use thecore Hadoop components.

Fig 1: Overview of the Map-Reduce execution Framework



2.8 Least squares quantization in pulse code modulation

In [8], Lloyd defines, In*K*-means initially chooses k centers randomly. For each input point, the nearest center is identified. Points that choose the same center belong to a cluster. Now new centers are calculated for the clusters. Each input point identifies its nearest center, and so on. This process is repeated until no changes occur. The process of identifying the nearest center for each input point and recomputing centers is referred to as iteration. The number of iterations taken by Lloyd's algorithm is unknown. This algorithm may converge to a local minimum with an arbitrarily bad distortion with respect to the optimal solution

2.9 Fast k-means algorithms with constant approximation

In [10], Mingjun Song et al., have been proposed three constant approximation algorithms for*K*-means clustering.

The first algorithm runs in time $O((\frac{k}{\varepsilon})^k nd)$, where k is the number of clusters, n is the size of input points, d is dimension of attributes. The second algorithm runs in time $O(k^3n^2 \log n)$. This is the first algorithm for *K*-means clustering that runs in time polynomial in n, k and d. The run time of the third algorithm $O(k^5 \log^3 kd)$ is independent of n. Though an algorithm whose run time is independent of n is known for the *K*-median problem, this is the first such algorithm for the *K*-means problem.

3.0 Conclusion

In this paper the important problem of clustering and studies *K*-means++ algorithm in a very

large data situations. It leads to develop *K*-means++ initialization with MapReduce efficiently and propose the MapReduce *k*-means++ algorithm. The MapReduce initialization algorithm uses only one MapReduce job to choose *k* centers.

The standard *k-means++* initialization and weighted *k*-means++ initialization are applied to the Mapper phase and Reducer phase, respectively. For the reduction of MapReduce jobs, our algorithm saves a lot of communication and I/O cost. Furthermore, the proposed algorithm provides an(a2) approximation to the optimal solution of *K*-means

References

[1] E. Chandra, V. P. Anuradha, A survery on clustering algorithms for data in spatial database management systems," Computer Applications, 24(9), 2011, 19-26

[2] J. F. Lu, J. B. Tang, Z. M. Tang, J. Y. Yang, Hierarchicalinitialization approach for k-means clustering, Pattern Recogn. Lett., 6, 2008, 787-795

[3] D. Arthur, S. Vassilvitskii, K-means++: The advantages ofcareful seeding, in Proceedings of the Eighteenth Annual ACMSIAM Symposium on Discrete Algorithms, 2007, 1027-1035

[4] [4] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++, PVLDB, 5(7), 2012, 622-633

[5] S. Papadimitriou, J. Sun, Disco: Distributed co-clusteringwith map-reduce: A case study towards petabyte-scale end-to-endmining, in Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, 2008, 512-521

[6] A. Ene, S. Im, B. Moseley, Fast clustering using mapreduce, in Proceedings of the 17th ACM SIGKDD

[7] International Conference on Knowledge Discovery and Data Mining, 2011, 681-689

[8] R. L. Ferreira Cordeiro, C. Traina, Junior, A. J. Machado Traina, J. L'opez, U. Kang, C. Faloutsos, Clustering very large Multidimensional datasets with mapreduce, in Proceedings of the 17thACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, 690-698

[9] S. Lloyd, Least squares quantization in pcm, Information Theory, IEEE Transactions on, 28(2), 1982, 129-137

[10] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu, A local search approximation algorithm for k-means clustering, Comput. Geom. Theory Appl, 28(2-3), 2004, 89-112

[11] M. Song, S. Rajasekaran, Fast k-means algorithms with constant approximation, in Algorithms and Computation, 2005, 1029-1038