International Journal of Database Theory and Application
Vol.7, No.3 (2014), pp.41-48
http://dx.doi.org/10.14257/ijdta.2014.7.3.05

Research of parallel DBSCAN clustering algorithm based on MapReduce

Xiufen Fu, Shanshan Hu and Yaguang Wang

School of Computer, Guangdong University of Technology, 510006, P.R.China
xffu@gdut,edu.cn, 895962584@qq.com

Abstract

For the lack of "density-based spatial clustering with noise" (DBSCAN) algorithm in dealing with large data sets, MapReduce programming model is proposed to achieve the clustering of DBSCAN. Map functions to complete the data analysis, and get clustering rules in different data objects; Then Reduce functions merge these clustering rules to get a final result. Experimental results show: the DBSCAN of MapReduce running on the cloud computing platform Hadoop has good speedup and scalability.

Keywords: *DBSCAN; clustering analysis; MapReduce; Hadoop*

1. Introduction

Clustering is a process which the objects in the collection is divided into a number of different types of clusters, a cluster is a set of similar data object [1]. Those objects in the same cluster are similar to each other, but be different from other clusters. Clustering algorithms are an important method for data mining, and be widely used in many fields, the research and application of clustering mining analytical method is related to multiple fields, such as statistics, machine learning, pattern recognition, spatial databases, *etc.* [2]. So far, many clustering algorithms have been proposed, one of which is DBSCAN (density-based spatial clustering with noise). The DBSCAN is not sensitive to the order of the objects in the dataset, and data can be divided into clusters of arbitrary shape while using the DBSCAN algorithm to clustering. However, in the practical application of today, DBSCAN algorithm faces many challenges, for example, it has higher computational complexity compared to other clustering algorithms (such as K-means). Also with the rapid growth in the size of the dataset, only in stand-alone mode running DBSCAN is inefficient, traditional DBSCAN algorithm faced with scalability issues. In response to these problems, executing DBSCAN in parallel on shared-nothing clustering is proposed to improve the efficiency of clustering. MapReduce [3] as an ideal platform for parallel programming, due to its simplicity, scalability and fault-tolerance, are increasingly being applied to large data set analysis. This paper based on the cloud computing platform of Hadoop, study the parallel implementations of DBSCAN algorithm with MapReduce, to resolve the scalability problem in dealing with large data sets that DBSCAN algorithm faced.

2. MapReduce Programming Model

MapReduce is a software framework proposed by Google, which is a basis computational model of current cloud computing platform. Its main function is to handle massive amounts of data. Because of its simplicity, MapReduce can effectively

ISSN: 2005-4270 IJDTA

deal with machine failures and easily expand the number of system nodes. MapReduce provides a distributed approach to process massive data distributed on a large-scale computer clusters. The input data is stored in the distributed file system (HDFS), MapReduce adopts a divide and conquer method to evenly divided the inputted large data sets into small data sets, and then processed on different node, which has achieved parallelism. In the MapReduce programming model, data is seen as a series of key-value pairs like <key,value>, as shown in Figure 1, the workflow of MapReduce consists of three phases: Map, Shuffle, and Reduce. Users simply write map and reduce functions. In the Map phase, a map task corresponds to a node in the cluster, as the other word, multiple map tasks are be running in parallel at the same time in a cluster. Each map call is given a key-value pair (k1,v1) and produces a list of (k2,v2) pairs. The output of the map calls is transferred to the reduce nodes (shuffle phase). All the intermediate records with the same intermediate key (k2) are sent to the same reducer node. At each reduce node, the received intermediate records are sorted and grouped (all the intermediate records with the same key form a single group). Each group is processed in a single reduce call. The data processing [4-6] can be summarized as follows:

Map $(k1, v1) \longrightarrow list(k2, v2)$

Reduce $(k2, list(v2)) \longrightarrow list(k3, v3)$

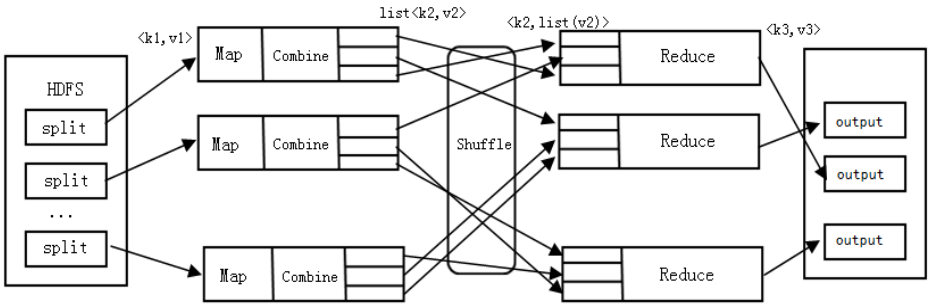

Figure 1. Data Processing Flow of MapReduce

3. DBSCAN Clustering Algorithm

The purpose of clustering algorithm is to convert large amounts of raw data into separate clusters in order to better and faster access. DBSCAN and K-means are two major algorithm for processing clustering problem**.** DBSCAN is a density-based clustering algorithm [7-8], which can generate any number of clusters, and also for the distribution of spatial data. K-Means algorithm is based on the prototype, which can find the approximate class for the given value. Compared to K-means algorithm, DBSCAN does not need to know the number of classes to be formed in advance. it can not only find freeform 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. The idea of the algorithm is：for all of the unmarked objects in data set D, select object P and marked P as visited. Region query for P to determine whether it is a core object. If P is not a core object, then mark it as noise and re-select an other object that is not marked. If P is a core object, then establish class C for the core object P and generals the objects within P as seed objects to region query to expanding the class C until no new object join class C, clustering process over. That is when the number of objects in the given radius (ε) region not less

than the density threshold (MinPts), then clustering. Because of taking the density distribution of data object into account, so it can mining for freeform datasets. The following are related terms, definitions [9] of DBSCAN algorithm:

(1)The area in ε of the specified object P is called ε region of P.

(2)If the number of objects in the ε region of object P is not less than the given density threshold MinPts, say P is a core object.

(3)The given dataset D, if object P is included in the ε region of object Q, and Q is a core object, say P starting from Q is directly density-reachable.

(4)Object list {P1, P2, ..., Pn}, P1 = Q, Pn = P, for Pi $\in$ D ($1 \leq i \leq n$), if Pi +1 starting from Pi is directly density-reachable on ε and MinPts, then P starting from Q is density-reachable on ε and MinPts.

(5)For the object O in the dataset D, if object P starting from O is density-reachable on ε and MinPts and object Q starting from O is also density-reachable on ε and MinPts, then call P to Q is density-linked on ε and MinPts.

(6)If object P does not belong to any class, P is said to noise.

DBSCAN discover classes by querying the ε region of unmarked objects in the dataset. If the ε region of an object P has the number of objects is more than MinPts, then create a new class based on the core object P, all of the objects in the ε region of P and P are belong to the same class. objects in the ε region of P are treated as seeds and region query for these seeds to expanding the class until there is no new object join class. Algorithm pseudo-code is as follows:

```
Input:dataset D, radius Eps, density threshold MinPts
Output:class C
1.DBSCAN（D, Eps, MinPts）
2.Begin
3.init C=0;// The number of classes is initialized to 0
4.for each unvisited point p in D
5.  mark p as visited; //marked P as accessed
6.  N = getNeighbours (p, Eps);
7.  if sizeOf(N) < MinPts then
8.     mark p as Noise; //if sizeOf(N) < MinPts,   then mark P as noise
9.  else
10.    C= next cluster; //create a new class C
11. ExpandCluster (p, N, C, Eps, MinPts);//expand class C
12. end if
13.end for
14.End
```

DBSCAN algorithm has two parameters: the radius ε and density threshold MinPts, Determination of parameters determine the final clustering result. On the computational complexity, If using spatial index, it can be analyzed the time complexity of DBSCAN is O (nlogn), n is the number of objects in the database. Otherwise, the time complexity rises to O (n^2) [10]. As we know, DBSCAN algorithm firstly determine whether an

object is the core object, if it is ,then continue to expand the class with the object as the center. In this process, with the increasing of core objects, the objects which are not marked are stored in memory, if there is a very large clustering in database, it will require a lot of memory space to store the core object information, and will lead to high I/O overhead, as a result, the clustering speed will be seriously affected. So this paper proposes a parallel DBSCAN clustering algorithm based on MapReduce to reduce I/O overhead and improve the clustering speed.

4. DBSCAN Algorithm based on MapReduce

4.1. Feasibility Analysis

The approach of DBSCAN algorithm extending class is by verifying whether the given object in the dataset D is a core at the specified radius ε. The Algorithm takes a large part of the time spending on the region query of object. When the dataset is very large, the inputted data objects are many, serial DBSCAN algorithm to determine whether each of object is core object will consume a high I/O overhead. Researchers found that the determination of core object can be parallelized, and we can get a conclusion by analyzing the algorithm that if an object exists in two different classes, and it is a core object, then these two classes can be merged into a new class. Otherwise the object belongs to one of classes, there is no relationship between the two classes. This paper introduces a new concept - sharing object.

- sharing object

Core objects P and Q belong to different classes, if the object O starting from P, Q are directly density-reachable, then called object O is a sharing object. And if O is a core object, called O is a sharing core object. If there exist a sharing core object in different classes, these classes can be combined into a new class. as shown in Figure 2, object O is a sharing core object:

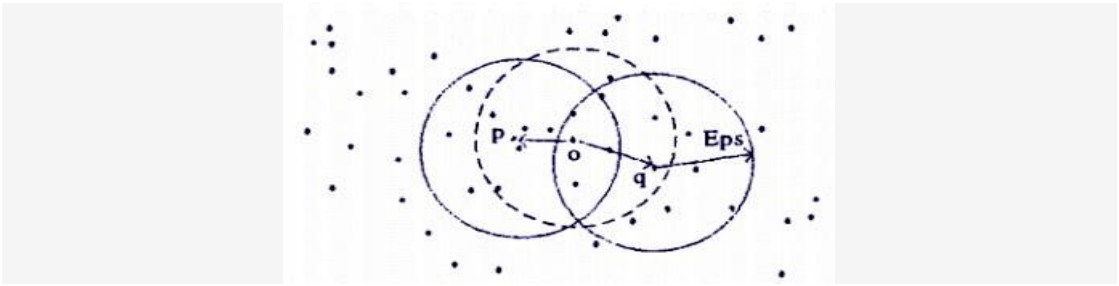

Figure 2. Clustering of Overlapping Objects in Region

Obviously we can take advantage of the MapReduce programming model to parallelize the whole process to save clustering time and resources. The basic idea of DBSCAN algorithm based on MapReduce is divided into four steps:

1. Step one: The data in the dataset cut into small blocks which are equal size.

2. Step two: the blocks are distributed to the nodes in the cluster, so that all of nodes in the cluster can run the Map function of Themselves in parallel to calculate and process those blocks.

(1) Initial Cluster. For any unlabeled object p in the dataset, using Map-Reduce parallel programming model to calculate the number of objects in its region to determine if it is a core object. If is, P and all the objects in the region of P constitute an initial class (C), and marked those objects with the same cluster identifier (Cid). Conversely, if p is not a core object, and there is no other object in its region, marked P as noise. Otherwise, detect whether there has a core object q in the region of non-core object p. If has, given the object p and the objects in the region of q with the same cluster identifier. Repeat until all of the objects in the dataset are identified. After the process is completed, get an initial class clusters and a noise set.

(2) Merge Result. Class merging is to consider these objects which exist in more than two classes. if there is a sharing core object, merging the two classes. Else classify the object as the proximity side. It will be given a new class name if there have different classes are combined. When there is no object exist in different classes, mergering initialize class completed.

3. Step three: merge the result of each processor.

4. Step four: Output clustering results.

Figure 3 shows the ideas of DBSCAN algorithm based on MapReduce, the data exchange format of each stage is <key,value>, and the object identifier Oid as the key, value is filled with (core_tag, used_tag, Cid, x, y, z). core_tag indicates whether it is a core object, used_tag identify whether there had been clustering, Cid means class logo.

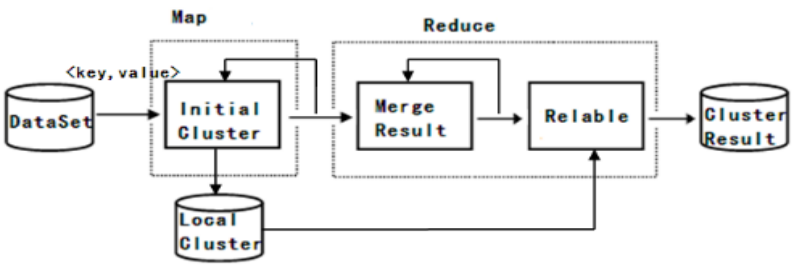

Figure 3. The DBSCAN Flow Chart based on MapReduce

4.2. The Realization of DBSCAN Algorithm based on MapReduce

Above, have conducted an analysis of feasibility of DBSCAN algorithm parallelization, the following will detail the implementation process of DBSCAN algorithm based on MapReduce, as shown in Figure 4:

(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” is (core_tag, used_tag, Cid, x, y, z). 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 execute Reduce process. when the dataset is large, 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. For example, each of Map function generates thousands of records like <oid,Cid>, these records are transmitted over the network to a specified Reduce function will consume a lot of network resources, increase the delay and reduce I / O performance. Here, Combiner functions combine the results that the local Map functions output and get intermediate key-value pairs <oid,list(Cid)>. Then partition functions will split the middle key-value pairs to generate partitions one by one. At last, these partitions will be distributed to the designated Reduce function.

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

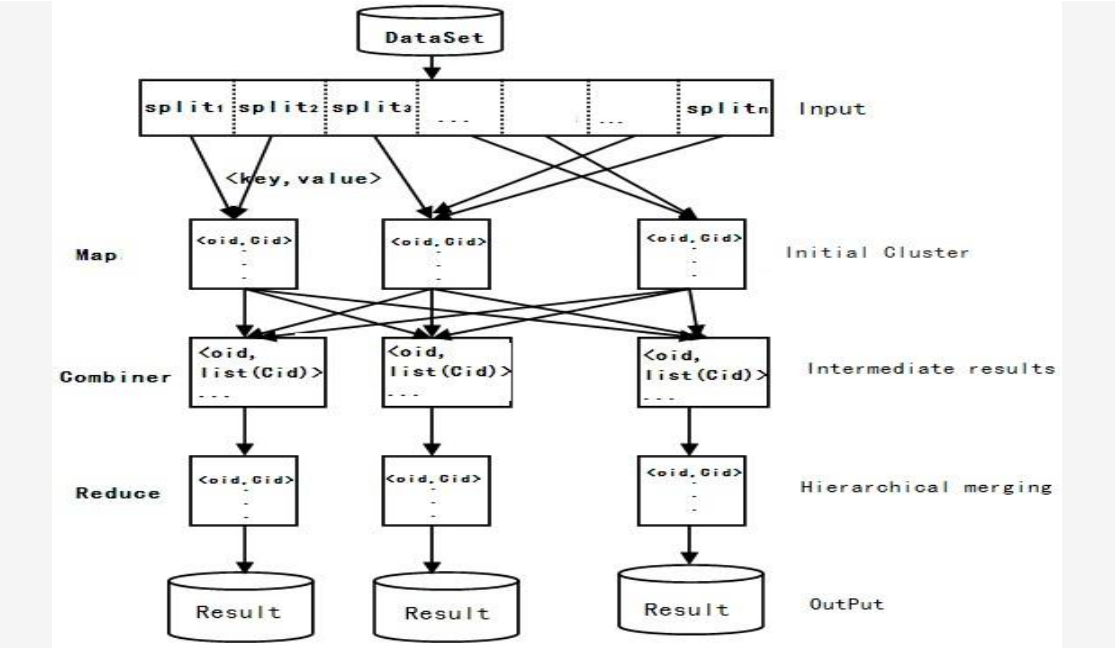

Figure 4. Data Processing Flow Chart

5. Analysis of Experimental Results

For this test, first to configure and deploy the cloud computing platform of Hadoop on the server. Hadoop are deployed on four PC Server, the port and role of which is shown as Table 1. One is for the NameNode and JobTracker service node, the others are for the DataNode and TaskTracker service node. Every machine configuration is as follows: two Physical CPU, twenty-four Logical CPU,the model of CPU was Intel (R) Xeon (R) CPU E5-26300@2.30GHZ,Memory 64G,RAID disk array, thirteen disk ,each two TB, all was twenty-six TB. The operating system is Redhat Linux version 2.6.32-279.e16.x86_64.

Table 1. Hadoop Cloud Computing Platform

Machine name	Port number	Role

hss01	10.21.17.101	Namenode,jobtracker
hss02	10.21.14.106	Datanode,tasktracker
hss03	10.21.14.108	Datanode,tasktracker
hss04	10.21.14.109	Datanode,tasktracker

In order to verify the timeliness of DBSCAN MapReduce algorithm , the paper makes experiments on different sizes of large data sets and runs on a single machine and Hadoop cloud computing platforms respectively. The results are shown as Table 2 and Figure 5.

Table 2. Execution Time Table

Records		20000	50000	100000	200000	300000
Execution time（s）	Standalone	5.26	18.49	34.75	117.38	486.98
	Cluster	2.85	4.23	14.62	25.49	79.12

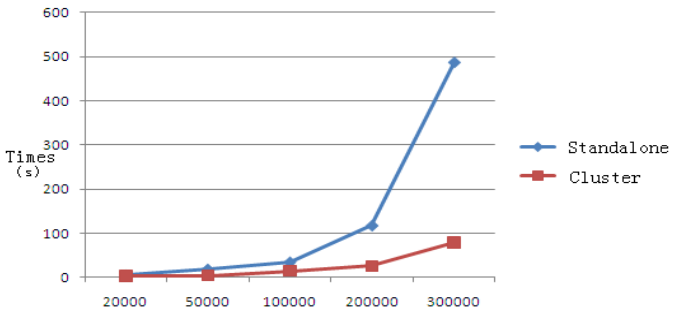

Figure 5. Execution Time

6. Conclusion

The paper comes up with a DBSCAN algorithm based on the MapReduce, which faces in scalabily problems when dealing with large data sets. Large data sets are cut into small set of data , and it processes the small set of data parallelly. Experimental results show that the DBSCAN algorithm based on the MapReduce alleviated the problem of time delay caused by large data sets and have good timeliness. The next task is to study how to solve the problem of mining efficiency of dynamic massive database furtherly. And applications of DBSCAN algorithm based on MapReduce in other fields.

Acknowledgements

This work was supported by the National Natural Science Foundation of China(61300107), the Guangdong Provincial Natural Science Foundation (No.10451009001004804 & 9151009001000007).

References

[1] F. Bo, H. WenNing, C. Gang and Z. Donghui, “K-means algorithm to optimize the initial cluster centers chosen”, Computer Engineering and Applications, vol. 49, no. 14, **(2013)**.

[2] D. Kellner, J. Klappstein and K. Dietmayer, "Grid-based DBSCAN for clustering extended objects in radar data", Intelligent Vehicles Symposium (IV), IEEE Digital Object Identifier, **(2012)**, pp. 365-370.
[3] J. Dean and S. Ghemawat, "MapReduce: a flexible data processing tool", Communications of the ACM, vol. 53, no. 1, **(2010)**, pp. 72-77.
[4] R. D. Chiara and U. Erra, "Massive simulation using GPU of a distributed behavioral model of a flock with obstacle avoidance", Proceedings of Vision, Modeling and Visualization. Stanford: [s.n.], **(2004)**, pp. 233 - 240.
[5] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters", Communications of the ACM 51.1, **(2008)**, pp. 107-113.
[6] P. Richmond and D. Romano, "Agent -based GPU,a real -time 3 D simulation and interactive visualisation framework for massive Agent-based modelling on the GPU", Proceedings International Workshop on Super Visualization. New York: ACM Press, **(2008)**, pp. 185-192.
[7] H. P. Kriegel, P. Kröger, J. Sander and A. Zimek, "Density-based clustering", Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 1, no. 3, **(2011)**, pp. 231-240.
[8] S. Kisilevich, F. Mansmann and D. Keim, "P-DBSCAN: a density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos", Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application. ACM, vol. 38, **(2010)**.
[9] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial–temporal data", Data & Knowledge Engineering, vol. 60, no. 1, **(2007)**, pp. 208-221.
[10] C. Böhm, K. Kailing and P. Kröger, "Computing clusters of correlation connected objects", Proceedings of the 2004 ACM SIGMOD international conference on Management of data. ACM, **(2004)**, pp. 455-466.

Authors

Xiufen Fu, she is a professor at Guangdong University of Technology, P.R China. She is a senior member of China Computer Federation (CCF) and a member of Collaborative Computing Professional Committee of (CCF); is the Vice-Director of Professional Committee of Database of Guangdong Province Computer Society. Her research interests include collaborative computing, database applications, knowledge management, network security. She chaired and participated in more than thirty research projects and published more than 130 papers.

Shanshan Hu, she will receive her master degree in computer science from Guangdong University of Technology, P.R China in 2014. Her research interests include collaborative computing, Cloud Computing, knowledge management, and so on.

Yaguang Wang, he receive his master degree in computer science from Guangdong University of Technology, P.R China in 2013.His research interests include collaborative computing, Data Mining, knowledge management, and so on.

