Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 147 (2019) 596–604

Procedia Computer Science

www.elsevier.com/locate/procedia

2018 International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI 2018

An improvement method of DBSCAN algorithm on cloud computing

Weipeng Jing[a,b] , Chuanyu Zhao [b,*], Chao Jiang [b]

[a] *College of Information & Computer Engineering, Northeast Forestry University, Harbin city, Heilongjiang Province, 150040, China .*
[b] *Heilongjiang Computer Science Center Computing, Harbin Heilongjiang 150040, China.*

Abstract

DBSCAN is a density-based data clustering algorithm, in image processing, data mining, machine learning and other fields are widely used. With the increasing of the size of clusters, the parallel DBSCAN algorithm is widely used. in image processing, data mining, machine learning and other fields are widely used. However, we consider current partitioning method of DBSCAN is too simple and steps of GETNEIGHBORS query repeatedly access the data set on spark. So we proposed DBSCAN-PSM which applies new data partitioning and merging method. In the first stage of our method we import the KD-Tree, combine the partitioning and GETNEIGHBORS query, reduce the number of access to the data set and decrease the influence of I/O in the algorithm. In the second stage of our method we use the feature of points in merging so as to avoid the time costing of the global label. Experimental results showed that our new method can improve the parallel efficiency and the clustering algorithm performance.

Keywords: Big data, DBSCAN, Data partitioning, Data merging ;

1. Introduction

*Weipeng Jing.
E-mail address: weipeng.jing@outlook.com

1877-0509
10.1016/j.procs.2019.01.208

Big data is a hot research direction in the field of computer in recent years, through the cluster can solve the problems such as machine learning, data mining, bioinformatics analysis, and many other big data fields. During the process of collecting data, through clustering analysis samples with the same or similar characteristics can be classified into the same category, and thus irrelevant items will be excluded. Mainstream clustering methods include classifying clustering method, such as K-MEANS; hierarchical clustering method, such as CURE and BIRCH, etc.; statistical model method, such as EM algorithm, etc.; density algorithm, such as DBSCAN, OPTICS, etc. As for the method based on the density, DBSCAN is a relatively typical one. It uses the data size in the super spherical region to measure the density of this area. It is able to find each cluster with any shape and effectively identify noise points.

Traditional clustering methods, based on stand-alone serial mode, cannot satisfy the demand for huge amounts of data analysis. Literature [1] introduces a method called P-DBSCAN algorithm which is based on the principal model. The algorithm achieves the parallel operation of the clustering process. However, due to limitations of master-slave mode, the parallel method is still relatively cumbersome. With the development of Hadoop and other open source platform, clustering algorithm parallelization work which is based on MapReduce [2] has been widely carried out. Literature [3] proposed a DBSCAN algorithm which is based on MapReduce model. The algorithm describes the process of DBSCAN clustering by Map / Reduce and achieves in the Hadoop platform. Literature [4] introduces the MR-DBSCAN algorithm. The algorithm is also based on the MapReduce model by introducing data partitioning strategy to make the algorithm efficiency improve. However, we found that as the amount of data increases exponentially, frequent I/O operations which were produced by MapReduce model iterative process are becoming a key factor influencing algorithm efficiency. In DBSCAN clustering process, because of the access to data sets by data Partitioning and GETNEIGHBORS query operations, the restraining effect is particularly evident.

Apache Spark brings into RDD [5] data structure. RDD is a kind of fault-tolerant and concurrent data structure. It supports internal storage across the cluster, and thus satisfies the low retardance feature. Due to the huge internal storage advantage of Spark, lots of parallelization work of clustering algorithm are carried out based on the platform of Spark. Literature [6] introduces a density based parallel clustering algorithm. DBSCAN is also a density based algorithm. However, nowadays there are few improvement opinions of DBSCAN algorithm based on the Spark platform.

Irving Cordova and others propose RDD-DBSCAN in the literature [7]. It is a kind of DBSCAN algorithm based on RDD. RDD-DBSCAN divides algorithm into three steps including data partitioning, local clustering, global mark, and the algorithm is implemented on the Spark platform. But RDD-DBSCAN takes BSP strategy to process the data partitioning problem, the data partitioning is abstracted as the construction of the binary tree. This partitioning process is simple, and the statistical properties of the data set are not considered, so in many cases it cannot meet the demand for effective partition of data sets. Literature [8] introduces a method of data partition in density clustering algorithm, and uses the tree data structure to complete the partition. Yet other partitioning methods of data sets have not been applied to the parallelization work of DBSCAN algorithm. In addition, RDD-DBSCAN separates data partitioning and regional query, increasing the read-write times of data sets and affecting the efficiency of algorithm.

Therefore, on the basis of literature [7], we proposed an improved parallel DBSCAN method (DBSCAN-PSM), improving the way of data partitioning, using the pre-construction strategy of KD tree to simplify the steps of data partitioning and regional query and realizing algorithm parallelization on the Spark platform. The experimental results show that compared with stand-alone way, the efficiency of DBSCAN based on the Spark platform increases significantly, which in return is helpful for processing big data.

2. DBSCAN Algorithm Analysis

DBSCAN algorithm is a clustering method of spatial data based on density, proposed by Ester Martin. It can find a cluster with any shape upon one density condition. DBSCAN has the following basic concepts:

Eps: radius of n DBSCAN Algorithm Analysis eighborhood. It is used to determine core points and noise points.

MinPts: minimum point sets. As a core point, the number of points in its neighborhood must exceed *MinPts*.

Core points: the points, of which the number exceed that of *MinPts* within the neighborhood.

Direct arrived density: there are points p and q. If point q is in the *Eps* neighborhood of point p and point p is the core point, it will be called direct arrived density between point p and point q.

Arrived density: if there are a series of points P_i (i=1, 2, 3, 4... n) in a region, let P=P_1, P_n=Q and the average

density from P_i to P_{i-1} can be reached, then P and Q is thought as arrived density.

DBSCAN is based on the fact that a cluster can be determined by its core points. Accordingly, the algorithm can be expressed as follows:

(1) For any given core point p, there is always a collection $\{o \mid o \in Eps(p)\}$, which is constituted by the object o of density p in the region D. The collection forms a complete cluster C, and there will be.

(2) For a given cluster C and its arbitrary core point p, C is equivalent to the collection $\{o \mid o \in Eps(p)\}$.

In order to determine clusters, DBSCAN arbitrarily takes a point p from region D, and then search all the points in region D which can satisfy *Eps*, *MinPts* and reach density p. If p is a core point, the points in the *Eps* neighborhood of point p is more than that of *MinPts*. If p is a boundary point, the points in the *Eps* neighborhood of point p is less than that of *MinPts*. Namely no object can reach the density from p, p is temporarily marked as a noise point and the algorithm continues to handle the next optional point in region D. During the execution of the algorithm, all arrived density objects of a core point need to be achieved by repeatedly regional query (GetNeighbors).

3. DBSCAN-PSM

As analyzed in the first chapter, the regional query step is an important step to determine the reachable objects of the density of core points. A large part of the I/O spending of DBSCAN algorithm derives from this step. Literature [7] uses R tree. But based on the global query of Eps and MinPts values it will produce a lot of memory and I/O spending, and seriously affect the efficiency of algorithm. As shown in figure 1, when points are extremely unevenly distributed in region D, this phenomenon is particularly serious.

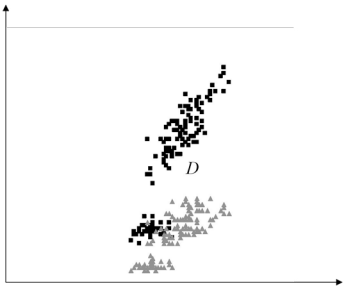

Figure. 1. An extreme distribution.

Because data partitioning step also needs I/O operation on data sets, thus considering the features of DBSCAN algorithm, data partitioning step and regional query step can be merged in the process of parallelization to reasonably avoid time costs in repeated I/O operations. On the other hand, from the angle of the parallelization, if DBSCAN algorithm is simply realized on the Spark platform and all data sets are put into a RDD but not to take data partitioning operation, it is difficult to call in all the internal storage and the low latency characteristic of Spark will not be effectively developed. So, merging data partitioning and regional query steps and adopting new data partitioning strategy are very important.

3.1. Data Partitioning Strategies

Literature [7] uses the Binary Space Partitioning strategy [9] in data partitioning. BSP is a kind of space partition method based on binary tree. It is based on the fact that any plane can divide space into two half-spaces. Repeating this process can build a binary tree, and each leaf node is the divided space. The space partitioning strategy of this approach is very casual, basically adopting the typical dichotomy. In the process of partitioning data sets, it is likely to make the space partitioning and the distribution of data sets not completely corresponding to each other, reducing efficiency of parallelization. In addition, the algorithm in the literature [7] needs an independent regional query (GetNeighbors) operation. Although it uses R tree to reduce the complexity of the process and is independently

processed in the divided partition, it still consumes a lot of time.

To solve these problems, this paper builds KD tree in the regional query [10], partitioning according to the division of KD tree. KD tree is a data structure of data points divided in K dimension space. In essence, KD tree is a balanced binary tree and a special case of binary system space partition tree. In two-dimensional space, as shown in figure 2, the dotted lines in the figure are parting lines.

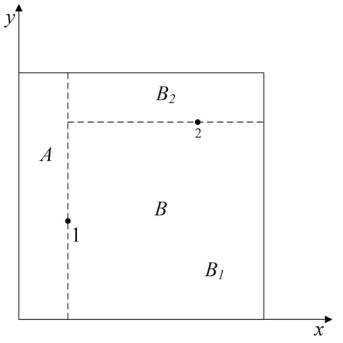

Figure. 2. KD tree determined parting lines

The main task of building KD tree is to determine the segmentation hyperplane in this figure according to certain strategy. The ensuring steps of segmentation hyperplane are as follows:

(1) Suggesting that a space R^n , S_i^2 $(i = 1,2,3...n)$ is the variance of certain dimension. The segmentation domain Split can be satisfied

$$Split = \{\max S_i^2 \mid x = 1,2,3...n\} \tag{1}$$

(2) According to the determined segmentation domain Split, after calculating values of all data points in the domain Split, the breaking point Node can be satisfied

$$Node = \frac{X_{Point(Split)/2} + X_{Point(Split)/2+1}}{2} \tag{2}$$

(the domain *Split* current dimension point is an even number)

$$Node = \frac{X_{Point(Split)}}{2} \tag{3}$$

(the domain *Split* current dimension point is an odd number)

(3) Hyperplane segmentation data sets determined by Node. All data points which are smaller than Node should be regarded as left subtree of KD tree, while all data points which are larger than Node should be regarded as right subtree of KD tree.

(4) Continuing steps (1) - (3) in the left and right subtrees of KD tree until meeting the suspensive conditions of partition.

The suspensive conditions of partition should be associated with properties of DBSCAN and data sets. Because it is necessary to ensure that each divided partition can be independently operated on DBSCAN. Namely all the points within the partition shall be marked as core points or noise points. The suspensive conditions of partition are:

(1) When the partition is less than 2Eps, as shown in the Figure 3, suggest that the partition is p. When the partition length is less than Eps, any point in the partition can obtain the complete clustering results only by combining partitions. That will make the merging operation of partitions very complex, and seriously affect the parallelization efficiency and clustering quality. Therefore, when partition length is less than or equal to 2Eps, stop partition.

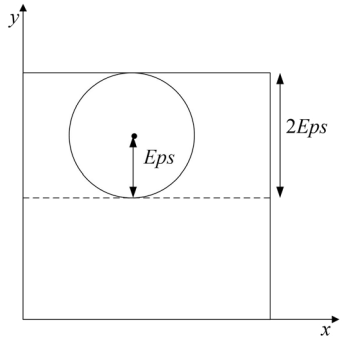

Figure. 3. Partition value

(2) Considering the characteristics of RDD, here introduce MemPts parameters. In order to play the low latency characteristic of Spark, each partition had better not exceed the memory capacity of processing nodes so that all the data can be put in the memory. Therefore, MemPts should be set as the smallest node of memory capacity to adapt to all the nodes.

(3) Considering in extreme cases KD tree can cause the imbalance of both sides, so here the data partitioning layer should also be limited. According to optimization KD tree search strategy, here set the layer as 10.

Such a partitioning strategy completes the operation of regional query and data partitioning during operating the data sets, avoiding the repetition of data sets. The partition divided according to KD tree features can maximally ensure that the numbers of all the data points in each partition are similar to each other, and at the same time during partitioning the characteristics of a group of data sets it can avoid lowering the clustering quality caused by partitioning operation. Eventually the level of parallelization can be improve.

3.2. Data Fusion Strategy

After partitioning, the algorithm independently adopts DBSCAN clustering operation in each partition. In order to get a global clustering value, it is necessary to take fusion operation after an independent DBSCAN operation. The literature [7] uses similar steps to process RDD-DBSCAN, implementing the fusion through marking the partitions and data points at the same time. Because partitioning fusion operation involves many partitioning edge points, this paper uses a relatively simple method which ignores partition marks and directly determines partitioning edge points.

(1) Merging Two Clusters

Considering that during partitioning process it is likely to separate the points which belong to the same cluster into two adjacent partitions, as shown in Figure 4, therefore in the process of fusion these points should be merged.

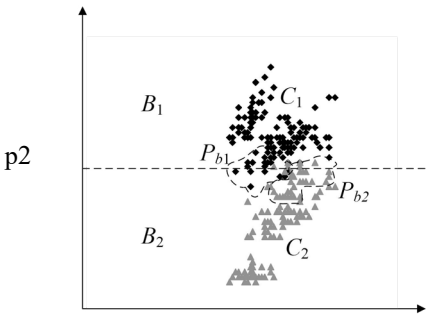

Figure. 4. Same clusters in different partitions

If and only if:

①a, b are located in adjacent partitions B1, B2. C1, C2 being as two clusters. $a \in C_1$, $b \in C_2$.

② $DISTANCE(a,b) \le Eps$. Eps is the neighborhood value of the clustering, Pb1,Pb2 are the boundary point sets of partitions B_1, B_2.

C1, C2 can be regarded as the same clustering. The above condition ② is only an approximation. Based on the DBSCAN density condition, it can be strengthened

$\forall p_i \in P_{b1}$, $\forall q_i \in P_{b2}$, $N_{b1} = |P_{b1}|$, $N_{b2} = |P_{b2}|$,

$$\overline{d} = \frac{\sum_i \sum_j DISTANCE(p_i, q_j)}{N_{b1} \times N_{b2}} \le Eps \tag{4}$$

(2) Noise Points Produce a New Cluster

During the partitioning process, there may be some points around partition B. In the clustering process of the partition, these points are marked as noise points. But after merging, the number of data points in Eps radius of the points satisfy MinPts. Therefore, these points will form a new cluster. In the process of partitioning fusion, these points should also be considered.

Suggesting that B1, B2 are two adjacent partitions. Pb1, Pb2 are boundary point sets of B1, B2. Noise point sets are PN in P_{b1}, P_{b2}. If and only if

① $\exists p_0 \in PN$, $\exists p_i \in PN (i = 1,2...n, n \ge MinPts)$.

② $DISTANCE(p_0, p_i) \le Eps$

It can be considered p0 as a new core point, and p0， pi builds a new cluster C.

3.3. Algorithm Realization

After introducing the data partitioning strategy, DBSCAN parallelization efficiency can be further improved. Based on the Spark platform, DBSCAN is built on the basis of data sets partitioning. After partitioning of data sets, DBSCAN independently runs on each divided data set. So it can avoid large modifications to DBSCAN algorithm, and maximally increase parallelization degree.

4. Experiments

4.1. Test Platform and Data Sets

The paper uses 5 Inspur I8000 blade servers (Xeon E5-2620 v26 nuclear 2.10 GHz processor, 8 GB memory, 200 GB hard disk drive), and uses Tenda TEG1024G gigabit Ethernet switch to connect, running the 1.5.0 version of Spark.

Our paper uses the tool----samples generator [11] provided by scilearn's [12] to generate testing data sets. In order to effectively test clustering effect under different scales of data, our paper selects different data sets including 100 thousand and 5 million data points. The condition of the data set is shown in table 1.

Table 1. The condition of data sets.

No.	Data Sets Lines	Clustering Amount	No.	Data Sets Lines	Clustering Amount
1	100000	4	6	2500000	10
2	500000	3	7	3000000	8
3	1000000	5	8	3500000	7
4	1500000	2	9	4500000	3
5	2000000	5	10	5000000	4

4.2. Clustering result of DBSCAN-PSM

Serial DBSCAN, RDD-DBSCAN and DBSCAN-PSM were used to test 100 thousand lines, 1 million 500 thousand lines, 2 million 500 thousand lines and 3 million 500 thousand row iris data sets. We introduction Purity and Entropy evaluation index.

$$Purity = \sum_{i=1}^{K} \frac{m_i}{m} P_i \tag{5}$$

$$Entropy = -\sum_{i=1}^{K} \frac{m_i}{m} \sum_{j=1}^{L} P_{ij} \log_2 P_{ij} \tag{6}$$

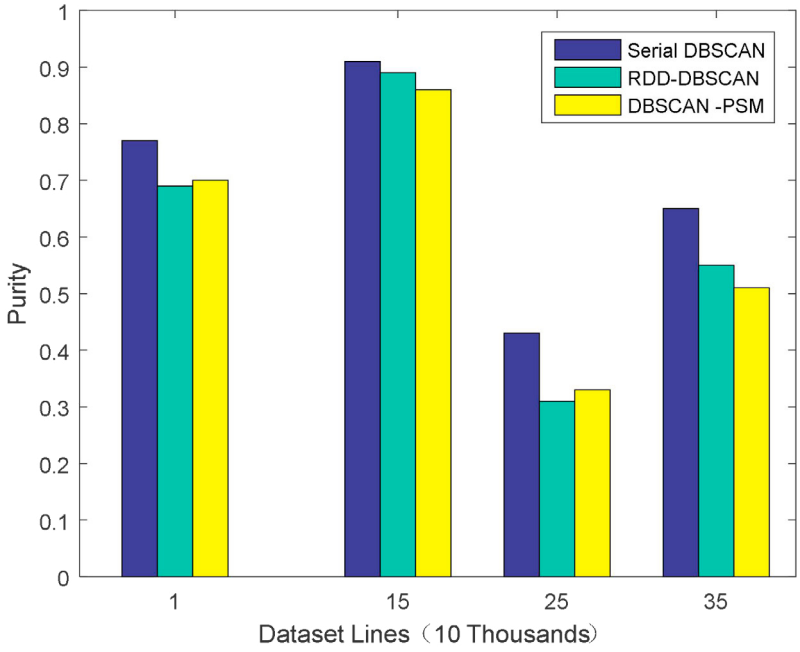

Figure. 5. Purity

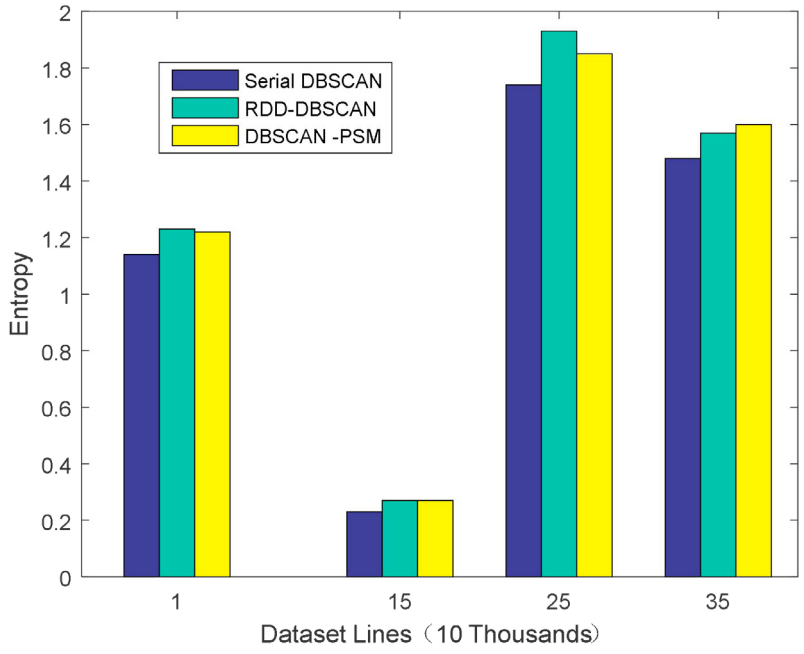

Figure. 6. Entropy

When the number of clusters is small, both RDD-DBSCAN and have better clustering quality, and the difference between DBSCAN-PSM and DBSCAN is small. However, in the 2 million 500 thousand and 3 million 500 thousand row data, because of the number of clusters, RDD-DBSCAN and DBSCAN-PSM clustering quality compared to the serial algorithm has a decline. This is mainly due to the more clustering data, the impact of data partitioning on clustering results is greater. But compared with RDD-DBSCAN, DBSCAN-PSM has no obvious change in the quality of clustering. The data partitioning method based on KD tree has no further effect on the quality of clustering.

5. Conclusion

We propose DBSCAN-PSM algorithm based on data partitioning, which further improves the efficiency of the algorithm, and handle the clustering operations of huge amounts of data and to make it more effectively applied into the cloud computing environment.

The emphasis of the next job lies on: (1) use the existing way of partitioning to study different Eps values adopted in different partitions and further improving the quality of clustering. (2) use the statistical characteristics of data sets, automatically select Eps and MinPts values to improve the automation level of the algorithm. (3) apply different partitioning ways for different data sets types to further improving the level of parallelization.

ACKNOWLEDGMENT

The work described in this paper is supported by Natural Science Foundation of Heilongjiang Province of China (ZD201403) ， Special Fund for Forest Scientific Research in the Public Welfare(201504307 ） and the Technological innovation talent research project in Harbin (2014RFQXJ132).

References

[1] M. Chen, X. Gao, and H. Li.: Parallel dbscan with priority r-tree. Information Management and Engineering(ICIME). IEEE, 2010.

[2] Wikipedia. (2015, April, 4) Mapreduce, http://en.wikipedia.org/wiki/MapReduce

[3] B.-R. Dai and I.-C. Lin.: Efficient map/reduce-based dbscan algorithm with optimized data partition. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, 2012, pp. 59–66.

[4] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan.: Mr-dbscan: an efficient parallel density-based clustering algorithm usingmapreduce. In: Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on. IEEE, 2011, pp. 473–480

[5] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica.: Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012, pp. 2–2.

[6] L. Li, X. Jiang, L. Liao.: Density Based Clustering on Large Scale Spatial Data Using Resilient Distributed Dataset. Journal of Hunan University(Natural Sciences), vol. 42, no. 8, pp. 116-124, 2015.

[7] Irving. Cordova, Teng-Sheng. Moh.: DBSCAN on Resilient Distributed Datasets. In High Performance Computing & Simulation (HPCS), 2015 International Conference on, 2015 pp. 531-540.

[8] Y. Yu, A. Zhou.: An Improved Algorithm of DBSCAN. Computer Technology and Development, vol. 21, no. 2, pp.30-33, 2011.

[9] M. J. Berger and S. H. Bokhari.; A partitioning strategy for nonuniform problems on multiprocessors. Computers, IEEE Transactions on, vol. 100, no. 5, pp. 570–580, 1987.

[10] Wikipedia. (2015, April, 4) KD-Tree, https://en.wikipedia.org/wiki/K-d_tree

[11] Dataset loading utilities, http://scikit-learn.org/stable/datasets/

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher,M. Perrot, and E. Duchesnay.: Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.