Research on Parallel Design of DBSCAN Clustering Algorithm in Spatial Data Mining

Gong-jian ZHOU
Xiamen University Tan Kah Kee College, Zhangzhou, Fujian, China

Keywords: Distributed computing, DBSCAN algorithm, Spark, MapReduce, Data segmentation.

Abstract. DBSCAN clustering algorithm uses fixed Eps and Minpts. When density distribution is uneven, the effect of clustering is not ideal, and the time complexity of the algorithm is $O(n^2)$. To solve the above problems, this paper proposes a parallel grid clustering algorithm and two cluster merging strategies of DBSCAN based on Spark platform, will find the Eps neighborhood to narrow the scope of the eight adjacent cells within the data object, and the parallel execution of the local clustering data with fast global clustering. The experiment shows that the improved DBSCAN parallel algorithm has better acceleration ratio and extensibility.

Introduction

The rapid development of the Internet and information technology has led to an exponential growth in data volume. How to cluster these massive data quickly and efficiently is a hotspot of data mining in the field of artificial intelligence and machine learning. From the perspective of machine learning, clustering is unsupervised learning, which can automatically discover the similarity between data sources, aggregate data with high similarity into a cluster, and separate dissimilar data as much as possible. However, in the face of large-scale data, traditional single-machine clustering algorithms have not been able to meet the processing needs of massive information in terms of efficiency and computational complexity.

In recent years, with the open source of cloud computing platforms such as Hadoop and Spark, the cloud computing parallel programming framework is adopted, and the traditional data mining algorithm is distributed and parallelized to improve the efficiency of big data mining. It has become a hotspot of data mining [1,4], this paper introduces the Yarn resource manager to improve the traditional DBSCAN algorithm in parallel, so that it can run on the Spark platform effectively, and improve the clustering efficiency and application scope of the algorithm.

Spark Parallel Computing Framework

Spark is a master-slave architecture platform. The master node is used to control the cluster to ensure the normal operation of the cluster. The slave node acts as a compute node and accepts commands and status reports from the master node. The user submits the application through the client, the master node finds a slave node to start the Driver program, and the driver applies the resource to the resource manager (Cluster Manager) to convert the application into a directed acyclic graph (RDD DAG), and The task information is generated after being converted to the DAG Scheduler, and the Task Scheduler is responsible for submitting the task to the control computing node (Worker), and the Worker starts multiple Executors to execute each task in parallel [3,5].

DBSCAN Algorithm

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an algorithm proposed by Ester Martin et al. to spatially cluster data object density as a similarity index [2]. DBSCAN refers to the largest set of density-connected points as clusters. Clusters can generate clusters of arbitrary
shapes, have good support for spatial spherical type data and non-spherical data types, and can also filter isolated noise points.

Related Concepts
The DBSCAN algorithm identifies a class based on a given density threshold (Eps and MinPts), and Eps and MinPts represent the cluster radius and the minimum number of objects in the neighborhood, respectively. The following are some related concepts used in the execution of the algorithm [4]:

1) Eps neighborhood: the data object P in the d-dimensional data set D to be processed is the center of the circle, and Eps is the radius region. \( N_{\text{Eps}}(p) = \{ q \in D | \text{dist}(p, q) \leq \text{Eps} \} \) is included in the Eps neighborhood; where dist\((p, q)\) is the Euclidean distance between objects q and p in D.

2) Density value: For the point p in the data set D, the number of points Num contained in the Eps neighborhood is its density value.

3) Directly density-reachable: given the object p, q \( \in \) D, p is the core point, and the Eps neighborhood of p contains q, then the q to p departure is directly reachable. Recorded as: q \( \in \) \( N_{\text{Eps}}(p) \), |\( N_{\text{Eps}}(p) \)| \( \geq \) MinPts.

4) Density-reachable: given objects p_1, p_2, p_3, p_4,...,p_n \( \in \) D, where p_1=q, p_n=q, if p_{i+1} is reached from p_i, direct density If it is reachable, then p is said to have a density from q.

5) Density-connected: Given the object p, q \( \in \) D, if there is a point o \( \in \) D, satisfying p, q to o, the average density is reachable, then the p to q density is connected.

![Figure 1. Eps neighborhood, direct density reachable, density reachable, density connected.](image)

Algorithm Description
For the spatial data set D to be processed, the algorithm clustering radius is set to Eps, and the minimum number of objects in the neighborhood is MinPts.

1) Check the data object p that has not been accessed in the data set. If p has not been processed (printed to a cluster or marked as noise), search for its Eps neighborhood. If the number of objects included is greater than or equal to MinPts, then Create a new cluster C and put all the points into the candidate set N;

2) Checking the Eps neighborhood of the object q that has not yet been processed in the candidate set N. If at least MinPts data objects are included, these objects are added to the data set N; if q is not yet classified into a cluster, then q is placed into the data set C;

3) Repeat step (2) to continue checking the unprocessed objects in N until the current candidate set N is empty;

4) Repeat steps (1) ~ (3) until all data objects are classified into a cluster or marked as noise.

Algorithm Analysis
(1) When the index is not used, the time complexity of the DBSCAN algorithm is O(n^2), so that the algorithm requires a large amount of memory support and I/O overhead when clustering massive data.

(2) The DBSCAN algorithm needs to pass in two fixed global parameters Eps and MinPts, and is sensitive to these two parameters, especially in the case of uneven density distribution, their changes will lead to large differences in clustering results.
(3) For the boundary object, the DBSCAN algorithm often determines which cluster belongs to according to the principle of “who first finds who belongs to it”, so that the accuracy of the cluster is greatly affected by the data processing order.

**DBSCAN Algorithm Parallel Design**

**DBSCAN Algorithm Overall Parallelization Strategy**

The current common parallelization methods are divided into data parallelism and task parallelism. In the data parallel, the data is divided into several small data blocks, which are respectively mapped to different processors, and each processor runs the same processing program to process the assigned data blocks. Each thread in the task parallel performs an assigned task, and these threads are assigned to the various compute nodes of the parallel computing system.

Considering that Spark has the memory abstract RDD feature, data parallelism can be used to make full use of Spark's memory-based computing features. RDD is used to perform local clustering and global merging of data to be processed according to custom data partitioning. In this paper, the grid data partitioning and quadratic clustering strategy are applied to the Spark platform for data parallelization design of DBSCAN algorithm. Read and format the data set from HDFS, cache it to memory, allocate small block data to each computing node according to the grid data partitioning strategy, and perform local DBSCAN clustering. After the local data clustering of each computing node is completed, the clustering result is globally merged and remarked to generate a global cluster class.

![DBSCAN parallel algorithm flow chart](image)

Figure 2. DBSCAN parallel algorithm flow chart.
DBSCAN Algorithm Parallelization Process Implementation

**Related Concepts.** Data segmentation is the most important step in the parallelization of DBSCAN clustering algorithm. The mesh data partitioning strategy divides the sample data into a finite number of n rectangular cells with the Eps radius of DBSCAN as the side length. The local clustering operation is performed on the block data set contained in the contiguous grid composed of rectangular units.

1. Related Concepts
   (1) Grid unit: based on the Eps neighborhood set by the user, the data set is divided into intervals of length Eps in each data dimension, and the entire data space is divided into an Eps grid with Eps as the side length of Eps, which is recorded as a grid unit: 
   
   \( \text{Cell} = \{c_1, c_2, \ldots, c_n\}, c_i = \{p_{ij} | 1 \leq i \leq n, l_{ij} = Eps \} \)

   \( i \) denotes a dimension, \( j \) denotes a spatial ordinal number. If the edge length of the last grid unit is less than Eps, it is calculated according to the actual side length.

   (2) Adjacent mesh set: A mesh set of rectangular regions formed by eight cells along the upper, lower, left, and right directions of the cell \( \text{Cell} \).

   (3) Boundary point: A type of data object with a specific meaning in a data set. It is defined as being located at the edge of one or more partitioned data. It may belong to a cluster or belong to multiple clusters, or it may be an isolated data object. The attribution of its clusters is not certain.

2. Description of Operation Steps
   The DBSCAN algorithm based on the grid unit first obtains the collection of the data object and its adjacent grid. If the number of data objects in the nine cells is less than MinPts, the data object is marked as noise, otherwise the next operation is continued. The specific steps are as follows:
   (1) dividing the space into n grid cells;
   (2) establishing a mapping relationship between the data object and the grid unit;
   (3) Find the object in the data set that is not identified as cluster or noise. If \( p \) is not processed, check the number of objects in the grid and the adjacent grid in a total of 9 grids. If the number of objects is greater than or equal to MinPts, then it is marked as a new cluster \( C \), and all points in it are placed in the candidate set \( M \), otherwise the data object is marked as noise;

   (4) Selecting the object \( q \) in the candidate set \( M \) that has not been processed, and checking the adjacent mesh set. If the number of objects included is greater than or equal to MinPts, put them into the candidate set \( M \); if \( q \) is not classified into a certain cluster, then mark \( q \) as cluster \( C \).

   (5) If the candidate set \( M \) is not empty, repeat step (4)

   (6) Repeat steps (3)~(5) until all data objects have been classified into a cluster or marked as noise.

Local Clustering Stage. After the data segmentation is completed, the DBSCAN clustering algorithm can be parallelized by using the MapReduce programming model on the Spark platform. Since the previous algorithm guarantees that the size of each data block is not much different, and the size of the data block is guaranteed to be smaller than the available memory of the host; therefore, the DBSCAN algorithm can perform well at each node. After each local data set is locally clustered, all data objects are marked as core, boundary, or noise.

Result Consolidation Phase. After the local clustering is completed, the data objects on each partition block have been marked with clusters. For non-boundary points on each block, the cluster has ended; for any one of the boundary points, there are three possible types of points: core points, non-core points, and noise points; extracting boundary points on each partition. According to the block number, the second clustering operation is performed to obtain the secondary cluster mark of the boundary point, and the two family tags are compared before and after, and different processing is performed according to the different conditions of the two cluster marks. There are two cases at this time: one is two independent clusters without intersections, and there is no need to merge the two clusters; the other is that there are intersections, and the secondary clustering result of the boundary points determines whether to merge. The specific steps are as follows:

(1) selecting the boundary points that have been clustered on the partition;
(2) Forming each boundary point into a new boundary point block according to the block mark;
(3) Clustering new boundary point blocks;
(4) Judging the new and old family marks of the boundary points;
(5) processing the cluster according to the judgment result;

Performance Testing and Analysis

Experimental Data and Platform Configuration

The experimental data is selected from the 3D-Road-Network dataset provided by the UCI database. The dataset consists of attribute fields such as Road-ID, Longitude, Latitude and Altitude. There are 434874 data records, and the sample data set size is 10K, 20K, 40K, 80K, 100K;

The Spark physical cluster is built and the Yarn resource manager is introduced to experimentally deploy the DBSCAN parallel algorithm. The software and hardware environment of the platform is configured as follows:

| Platform name   | Hardware information                                      | Software information | Number of nodes |
|-----------------|-----------------------------------------------------------|----------------------|-----------------|
| Spark cluster   | CPU: Intel(R) Xeon(R) ES-2650 v2 @2.60GHz, 8 core Memory: 64G Cache: 20480KB | CentOS 7.0, Spark 1.6.0, Hadoop 2.6.0, Java 1.8.0, Scala 2.10.4, Tachyon 0.7 | 4               |

Analysis of Experimental Results

Speed Ratio Analysis. Speedup is the ratio of the time it takes an algorithm to run in a serial system and a parallel system to measure the effectiveness of parallelization of a distributed system or algorithm. It is defined as: $S_p = T_1/T_p$, where $S_p$ is the acceleration ratio, $T_1$ represents the processing time in serial, and $T_p$ represents the processing time in parallel. When the acceleration ratio index $S_p$ is larger, the algorithm is more efficient.

![Figure 3. Spark on Yarn based DBSCAN parallel algorithm speedup.](image)

It can be seen that when the data set is on the scale of 10K and 40K, the acceleration ratio curve tends to be flat as the number of nodes increases. This is because the data volume of the 10K and 40K data sets is relatively small, and the processing time is relatively short. As the number of nodes increases, the overhead of data transmission, task scheduling, resource management and the like reduces the execution efficiency of the algorithm. The acceleration ratio obtained when dealing with 80K and 100K data sets is basically linear growth, which indicates that the DBSCAN parallel algorithm based on Spark on Yarn mode can effectively improve the efficiency of the algorithm when the data volume is large enough and cannot be processed by a single machine. And the acceleration effect of the cluster is better when the data sets are larger.
**Scalability Analysis.** Scalability is used to test the practicability of the parallelization algorithm, which can reflect the impact of the number of cluster computing nodes on the parallelization efficiency. It is defined as: $E = \frac{S_p}{P}$, where $E$ is scalability, $S_p$ is the speedup ratio, and $P$ is the number of cluster computing nodes. If $E$ is larger, the scalability of the algorithm is better.

![Figure 4. Spark on Yarn-based DBSCAN parallel algorithm scalability.](image)

It can be seen that when the size of the data set is small, when the number of nodes in the cluster increases to a certain number, the expansion ratio will drop sharply. This is because when the data set is small, when the node reaches a certain number, its resources are enough to process the data set. Increasing the number of nodes will increase the overhead of task scheduling, network communication, resource management, etc., but it is not too efficient. However, when the data set size is large, the parallel expansion ratio of the DBSCAN algorithm is smoother. As the number of nodes increases, the efficiency of the algorithm on the platform remains basically stable.

**Conclusion**

Aiming at the problem that the traditional DBSCAN algorithm deals with the excessive clustering time of massive spatial data, this paper conducts parallel research and design on DBSCAN algorithm based on Spark platform, and tests and verifies it through Spark on Yarn physical cluster. The improved DBSCAN parallel algorithm has better Good speedup and scalability, and it also improves the execution efficiency of DBSCAN clustering algorithms and the field of big data applications.

**Acknowledgement**

This research was financially supported by the Fujian Provincial 13th Five-Year Education Science Planning Key Project (FJJKCGZ16-165).

**Reference**

[1] Wang Gui-zhi, Wang Guang-liang. Improved fast DBSCAN algorithm[J]. Journal of Computer Applications, 2009, 29(9):2505-2508.

[2] Feng Shao-rong, Xiao Wen-jun. New method to improve DBSCAN clustering algorithm quality[J].Journal of Xidian University, 2008, 35(3):523-529.

[3] Sun Yu-fen. Study on grid-based clustering algorithms[D]. Wuhan: Huazhong University of Science and Technology, 2006.

[4] N. Gupta, R. L. Ujjwal. An efficient incremental clustering algorithm[J]. World of Computer Science & Information Technology Journal, 2013, 3(5): 23-29.

[5] M. Kryszkiewicz, P. Lasek. TI-DBSCAN: Clustering with DBSCAN by Means of the Triangle Inequality[C]. Rough Sets and Current Trends in Computing. Springer Berlin Heidelberg, 2010:60-69.