Parallel DBSCAN with Priority R-tree

[1]Min Chen, [2]XueDong Gao
School of Economics and Management
University of Science and Technology Beijing
Beijing, P.R. China
[1]ynn.chenmin@gmail.com

[3]HuiFei Li
Soft Ware Group
IBM Global Services (China) Company Limited
Beijing, P.R. China
Lhf750324@sina.com

***Abstract*—According to the efficiency bottleneck of algorithm DBSCAN, we present P-DBSCAN, a novel parallel version of this algorithm in distributed environment. By separating the database into several parts, the computer nodes carry out clustering independently; after that, the sub-results will be aggregated into one final result. P-DBSCAN achieves good results and much better efficiency than DBSCAN. Experiments show that, P-DBSCAN accelerates more than 40% on one PC, and 60% on two PCs. In addition, the parallel algorithm has much better scalability than DBSCAN, so that it can be used for clustering analysis in huge databases.**

Keywords-Clustering; algorithm DBSCAN; parallel DBSCAN

I. INTRODUCTION

Clustering analysis is one of the most important tasks of data mining. Algorithm DBSCAN is a classic clustering algorithm in spatial databases. The key idea of DBSCAN [1] is that for each point of a cluster the neighborhood of a given radius has to contain at least a minimum number of points, i.e. the density in the neighborhood has to exceed some threshold.

Algorithm DBSCAN can discover clusters of arbitrary shape, whereas, when facing the huge spatial databases, both the memory usage and computational cost are expensive. In addition, the spatial access method R*-tree is not always efficient.

According to the efficiency bottleneck of algorithm DBSCAN, we present P-DBSCAN, a novel parallel version of this algorithm in distributed environment, such as computer clusters. Algorithm P-DBSCAN adopts a better spatial index, Priority R-tree, or PR-tree. It is the first R-tree variant that is not only practically efficient but also provably asymptotically optimal.

By separating the database into several parts, each [1]computational node builds PR-tree and carries out clustering independently; after that, the sub-results will be aggregated into one final result. P-DBSCAN achieves good results and much better efficiency than DBSCAN. In addition, the parallel algorithm has much better scalability than algorithm DBSCAN.

The rest of this paper is organized as follows. In the next section we introduce and analyze algorithm DBSCAN. In Section 3 we present the novel parallel algorithm, P-DBSCAN. Section 4 gives experimental results and their analysis. The paper concludes in Section 5.

II. DBSCAN ALGIRITHM

Here is some basic notion of DBSCAN algorithm [1]. D is the database of points.

[2]Definition 1: (Eps-neighborhood of a point) The Eps neighborhood of a point p, denoted by NEps(p), is defined by $NEps(p) = \{q \in D \mid dist(p,q) \le Eps\}$.

Definition 2: core point) q is a core point wrt. Eps, MinPts if NEps(q) contains at least MinPts points.

Definition 3: (directly density-reachable) A point p is directly density-reachable from a point q wrt. Eps, MinPts if

1) $p \in NEps(q)$

2) $|NEps(q)| \ge MinPts$.

Definition 4: (density-reachable) A point p is density reachable from a point q wrt. Eps and MinPts if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

Definition 5: (density-connected) A point p is density connected to a point q wrt. Eps and MinPts if there is a point o such that both, p and q are density-reachable from o wrt. Eps and MinPts.

Definition 6: (cluster) A cluster C wrt. Eps and MinPts is a non-empty subset of D satisfying the following conditions:

1) $\forall$ p, q: if $p \in C$ and q is density-reachable from p wrt. Eps and MinPts, then $q \in C$

2) $\forall$ p, $q \in C$: p is density-connected to q wrt. Eps and MinPts.

Definition 7: (noise) Let $C_1, \ldots, Ck$ be the clusters of the database D wrt. Epsi and MinPtsi, i = 1, . . ., k. Then we define the noise as the set of points in the database D not belonging to any cluster C_i.

To find a cluster, DBSCAN starts with an arbitrary point p and retrieves all points density-reachable from p wrt. Eps and MinPts. If p is a core point, this procedure yields a cluster C wrt. Eps and MinPts, and all the points of NEps(q) belong to it. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database.

The average run time complexity of DBSCAN is O (n2), where n is the total number of all the subjects. After using

[1]The National Natural Science Foundation of China

978-1-4244-5265-1/10/$26.00 ©2010 IEEE

spatial access methods such as R*-trees to support the region queries, its time complexity descends to O (n * log n).

The efficiency of DBSCAN is not good enough for the two following reasons.

The first one is that before the clustering process begins, algorithm DBSCAN needs to build spatial indexes R*-tree.

The second one is that DBSCAN needs to use k-dist graph to determine the appropriate value of two input parameters, Eps and MinPts .

The above two processes are both very time consuming, when the database is huge, both the memory usage and computational cost are expensive.

In this paper, we present P-DBSCAN, a novel parallel DBSCAN, which is not only much more efficient than DBSCAN algorithm, but has better scalability.

III. P-DBSCAN ALGORITHM

A. Spatial Access Method and parameters determination

1) Priority R-tree

In algorithm DBSCAN, Region query is supported by R*-tree [3], well, R*-tree and most R-tree [2] variants do not historically guarantee good worst-case performance [4]. So, in algorithm P-DBSCAN, we take Priority R-tree or PR-tree as the spatial index, which is the first R-tree variant that always answers a window query worst-case optimally.

PR-tree was proposed by Lars Arge in 2004. It is the first R-tree variant that is not only practically efficient but also provably asymptotically optimal. It performs similar to the best known R-tree variants on real-life and relatively nicely distributed data, but outperforms them significantly on more extreme data [4]. So we use it to support region query in algorithm P-DBSCAN.

2) Shape of Neighborhood

The shape of neighborhood in most previous studies about algorithm DBSCAN is round, whereas, whether R-tree, R*-tree or PR-tree, all these data structures split space with minimum bounding rectangles. In fact, region query is executed within these rectangles. So in algorithm P-DBSCAN, we adopt the rectangular neighborhood instead.

3) Determination of Parameters

Algorithm DBSCAN determine the two input parameters, Eps and MinPts by k-dist graph, actually, it eliminates the parameter MinPts by setting it to 4 for all 2-dimensional databases. However, our experiments indicate that when MinPts is set to 4, we could not get accurate clustering results in some cases with noise. Well, if we set MinPts to 7, we can get the precise results.

Furthermore, our experiments also show that it is the value of Eps that affects the clustering speed dramatically, the higher the value is, the more computation is needed. And the factor Eps has nothing to do with the computation. The related experimental result is shown in session 4.

B. Procedure of P-DBSCAN Algorithm

- Step 1, Parameters determination
  P-DBSCAN sets MinPts to 7, and then use 7-dist graph to determine the parameter Eps.
- Step 2, Database partition
  Before clustering, P-DBSCAN partitions the database into several parts with the following method,

P-DBSCAN projects the database on each dimension coordinate; then, it partitions the database according to the distribution characteristics of the database. Here is an example.

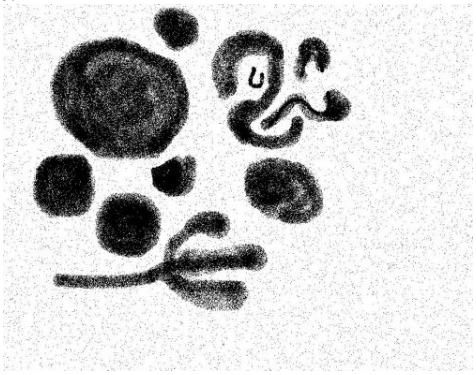

Figure 1. Database

Set of points is depicted in Figure 1, P-DBSCAN projects the database on X and Y coordinates separately, the result is shown in Figure 2.

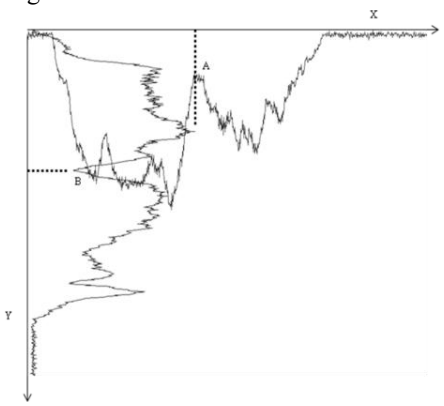

Figure 2. Projection on coordinates

If there are only two computational nodes, the database can be partitioned at point A and point B into 4 parts; and then the 4 parts are deployed to the 2 nodes (4 CPU).

- Step 3, Clustering independently
  Each node builds PR-tree and carries out the clustering independently. No further communication is necessary throughout this step.
- Step 4, Sub-results aggregation
  The correct clustering result is determined by merging the locally detected clusters according to the merging policy of [5].
  The procedure of P-DBSCAN algorithm is shown in Figure 3.

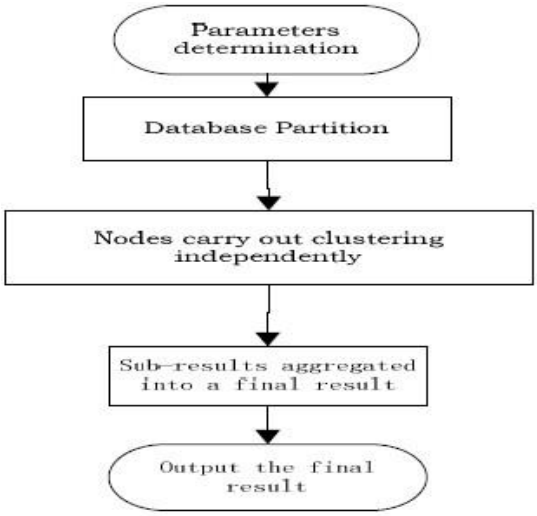

Figure 3. Procedure of P-DBSCAN

IV. Implementation And Performance Evaluation

In this section, we compare the performance between P-DBSCAN and DBSCAN. The configuration for experiments includes 2 nodes in the computer cluster, interconnected by a 100 Mbps LAN. Each node runs the windows XP operating system on 2.0 GHZ with 3 GB of main memory. The programs are written in Java.

A. Efficiency of P-DBSCAN Algorithm

The database with noise is depicted in Figure 1. The database contains 126,862 points, that is, 11 clusters. The parameters, Eps and MinPts are set to 3 and 7 respectively. The clustering result is indicated in Figure 4.

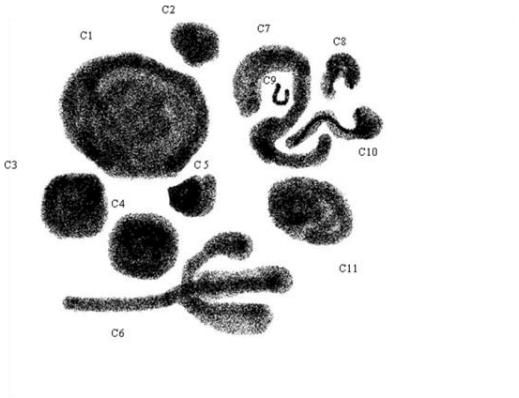

Figure 4. Clustering result

The experimental clustering result is correct, besides P-DBSCAN is much more efficient, that is shown in Table I.

TABLE I. EFFICIENCY COMPARISON

algorithm	No. of PC	Run time s	Run time ratio	result
DBSCAN	1	97.015	100%	11 clusters
P-DBSCAN	1	55.422	57%	11 clusters
P-DBSCAN	2	31.812	33%	11 clusters

The experiments show that, the execution time of P-DBSCAN is only 60% of DBSCAN when they are executed on the same computer; and 30% of it on two computers.

B. Scalability of P-DBSCAN Algorithm

To test P-DBSCAN's scalability with the number of points, we report results on three different size datasets. The results are shown in Figure 5. We can see that when the number of points increases, the run time of P-DBSCAN increases in a much more conservative way compared to DBSCAN algorithm.

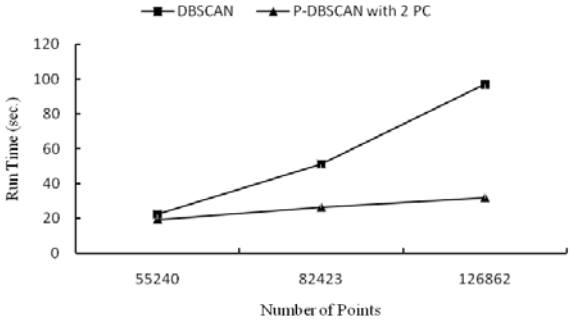

Figure 5. Scalability comparison

C. Parameter influence

We use the database depicted in Figure 1 to indicate how the parameters MinPts and Eps influence the computational cost. P-DBSCAN is executed on one computer, and the result is depicted in Table II.

When Eps is set to 3, even though MinPts varies, the execution time almost does not change; on the contrast, when MinPts is set to 7, various values of Eps lead to very different computational costs. So we can conclude that it's Eps that affects the computational cost much.

TABLE II. INFLUENCE OF PARAMETERS

Eps / MinPts	1	2	3
4			54750ms
7	9297ms	23703ms	55422ms
10			56406ms

V. Conclusion

In this paper, we presented a novel parallel DBSCAN algorithm in distributed environment, P-DBSCAN, which

adopts the spatial access method Priority R-tree other than R*-tree to support region queries. The results of these experiments demonstrate that P-DBSCAN is much more efficient than original algorithm DBSCAN. It accelerates more than 40% on one PC, and 60% on two PCs. In addition, the parallel algorithm has much better scalability than DBSCAN, so that P-DBSCAN can be used for clustering analysis in huge databases.

ACKNOWLEDGMENT

This work was supported in part by a grant from the National Science Foundation, the name is High-Dimensional Sparse Data Clustering, with the number of 70771007.

REFERENCES

[1] Easter M, Kriegek H-P, Sander J, etc, "A density-based algorithm for discovering clusters in large databases," Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD 96), AAAI Press, Aug.1996, pp. 226-231.

[2] A. Guttman, "R-trees: A dynamic index structure for spatial searching," Proc. SIGMOD International Conference on Management of Data, (SIGMOD 84) ACM Press, June 1984, pp. 47-57

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," Proc. of the ACM SIGMOD International Conference on Management of Data(SIGMOD 90), ACM Press, May, 1990, pp. 322-331.

[4] Lars Arge, Mark de Berg, Herman J, Ke Yi, "The Priority R-Tree: A Practically Efficient and Worst Case Optimal R-Tree," Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD 04), ACM Press, June 2004, pp.347-358.

[5] Zhou ShuiGeng, Zhou AoYing, Cao Jing, "A data-partitioning-based DBSCAN algorithm," Journal of computer research & development, Vol. 37, Oct. 2000, pp. 1153-1159.