香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

Parallel DBSCAN on Spark

Group 12: Wang Jianming
Zhang Xinyue
Xing Zhenghao
Fu Chennan

2020/11/24

Content

- **Introduction to DBSCAN**

- **Naive parallel DBSCAN**
  - Naive partition
  - Local DBSCAN
  - Merge

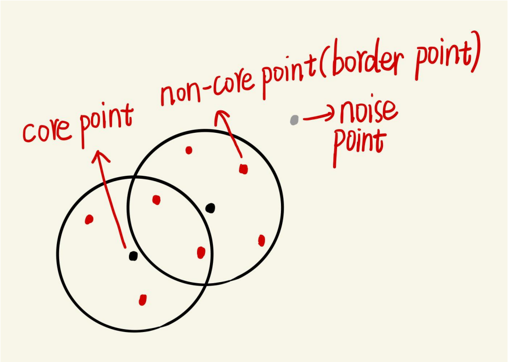
- **Optimization on partition**
  - Reduce-boundry based partition
  - Content based partition

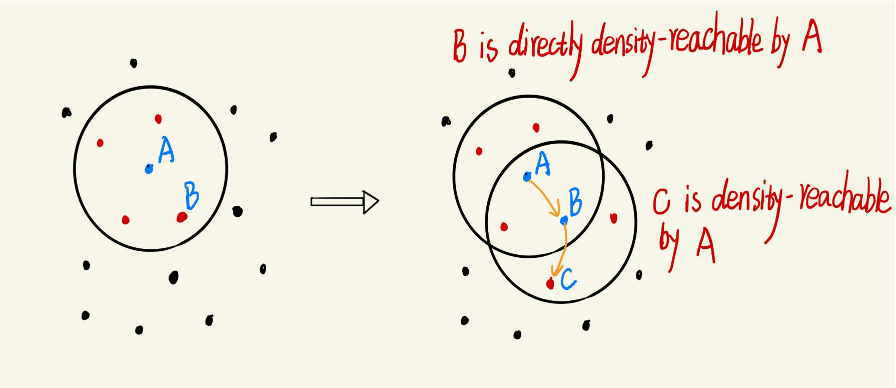
- **Conclusion**

Local DBSCAN algorithm

Concepts:

ε(Eps) — the given radius
MinPts — the given minimal points
if MinPts = 4:





Core idea:

1. select a data point p arbitrarily from the data set.
2. determine whether p is a core point by Eps and MinPts.If p is a core point,find all the data point which are **density-reachable from p** to form a cluster. If not, select another point to do the same steps as before.

Matrix DBSCAN algorithm

1. Introduction: Build a matrix to hold all those point-pair distance value instead of computing every time.

2. Pros: In principle, the computational cost could be reduced by half, which is not big algorithm-wise improvement, but a giant increase in engineering, especially it comes to big data problem.

example

	Point 1	2	3	4	5
point 1	X	$D_{1,2}$	$D_{1,3}$	$D_{1,4}$	$D_{1,5}$
2		X	$D_{2,3}$	$D_{2,4}$	$D_{2,5}$
3			X	$D_{3,4}$	$D_{3,5}$
4				X	$D_{4,5}$
5					X

Distance from P_1 to P_5

Parallel Matrix DBSCAN algorithm

Two Dimensional Example: (x, y)

Partition:

1. define partition number
2. define partition block position in the space
   a. find the minimum x and y and find the maximum x and y
   b. add eps to upper bound and minus eps to lower bound
   c. divide the whole area into corresponding partition id (p_id)
3. build rdds (pid, dataset_points) by judging whether the points is in the partition area or n

Local DBSCAN:

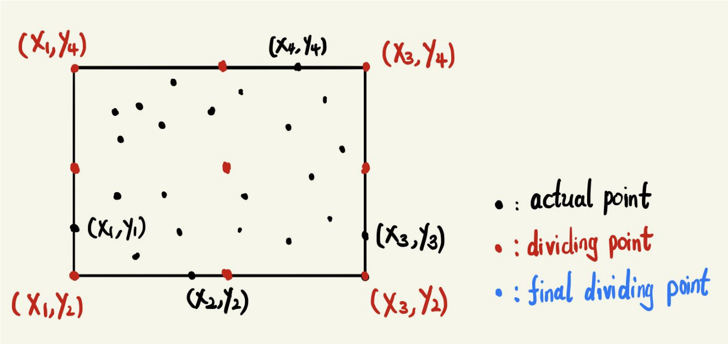
1. each rdd will execute a matrix DBSCAN
2. each execution will return ((partition_dataset, core_point list), local_tags)

Merge:

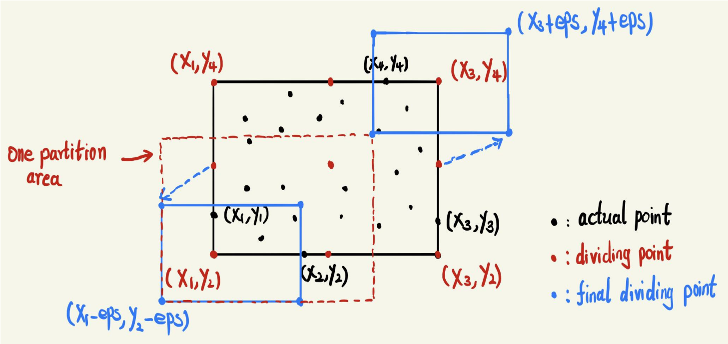
1. update local tag list into global tag list
2. return final global tag list

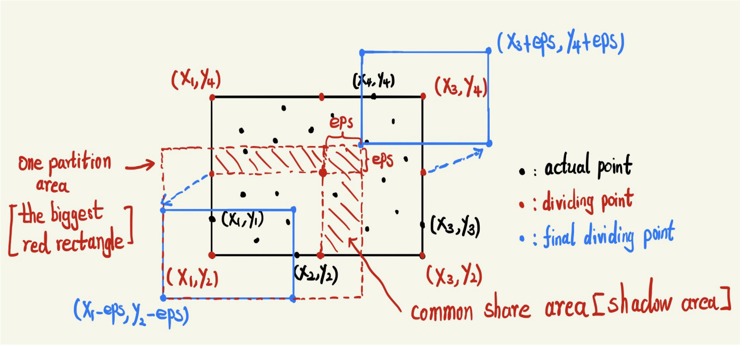
Parallel Matrix DBSCAN algorithm——Partition

partition number: 4   eps (epsilon): 3   MinPts (minimum points): 2



Define partition block position in the space

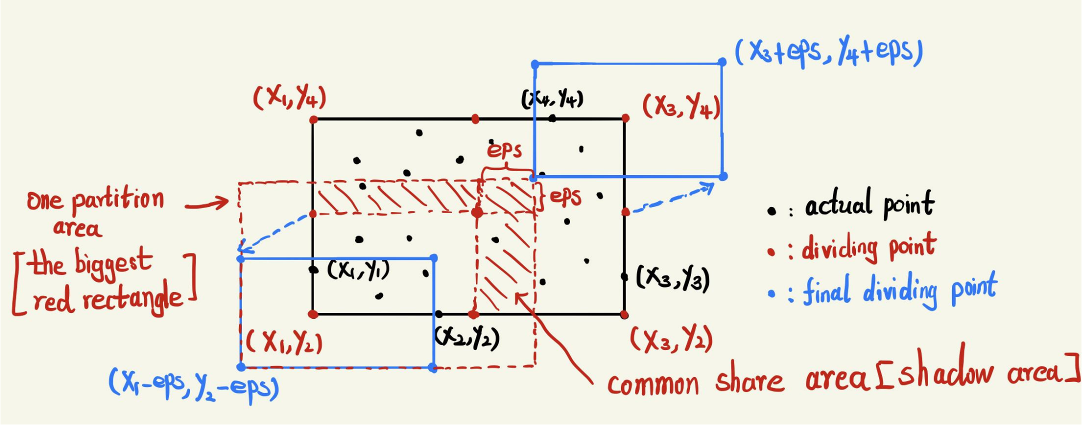


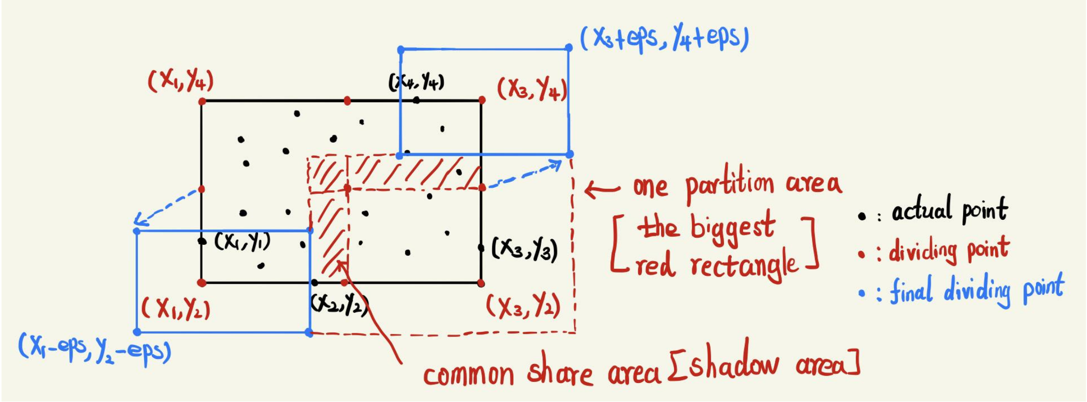


Parallel Matrix DBSCAN algorithm——Local Matrix DBSCAN

partition number: 4   eps (epsilon): 3   MinPts (minimum points): 2

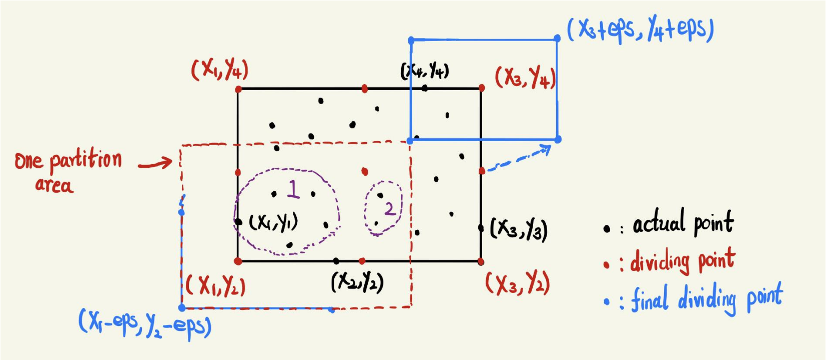
Points in the red shadow area will have multiple labels

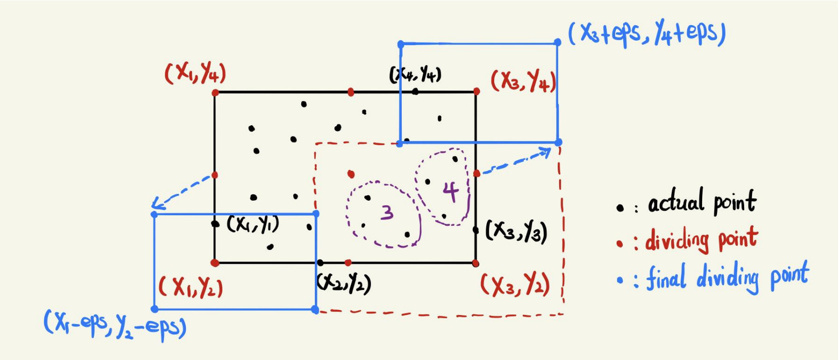


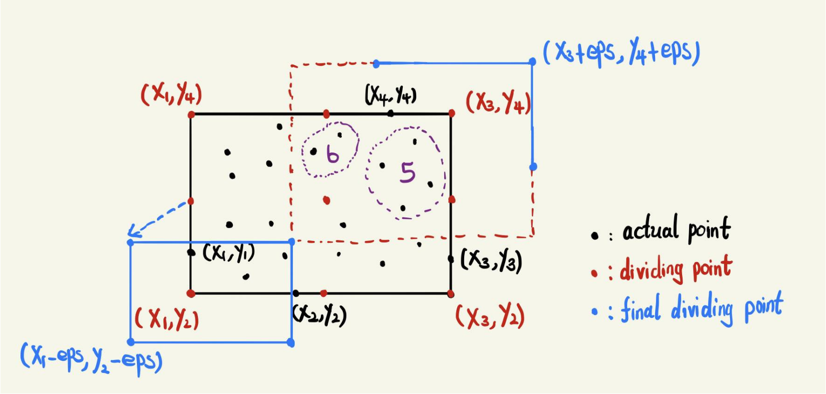


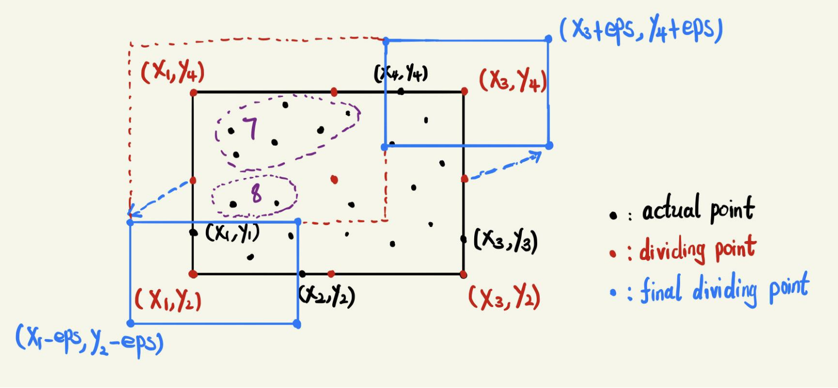
Parallel Matrix DBSCAN algorithm——Local Matrix DBSCAN

partition number: 4 eps (epsilon): 3 MinPts (minimum points): 2



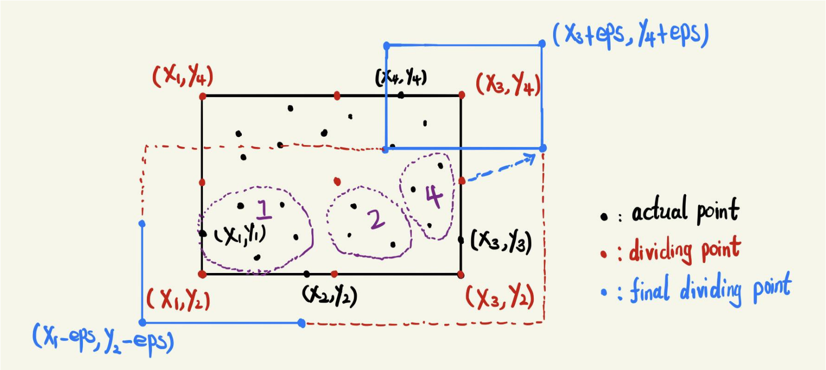


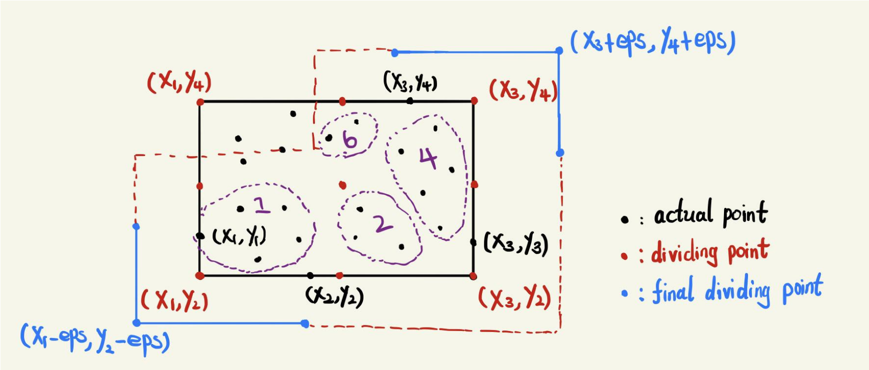


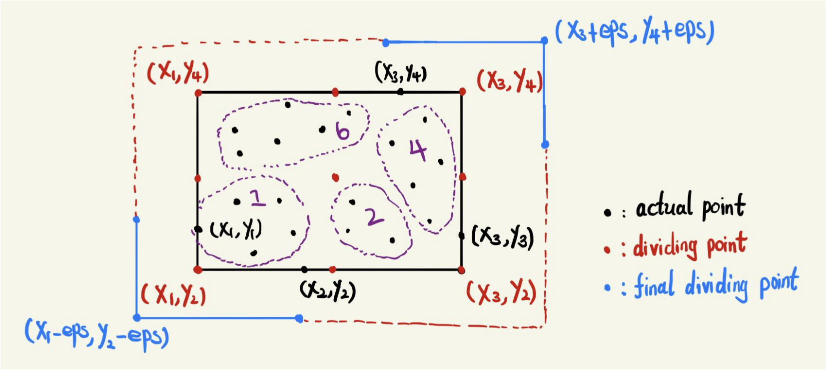


Parallel Matrix DBSCAN algorithm——Merge

partition number: 4   eps (epsilon): 3   MinPts (minimum points): 2



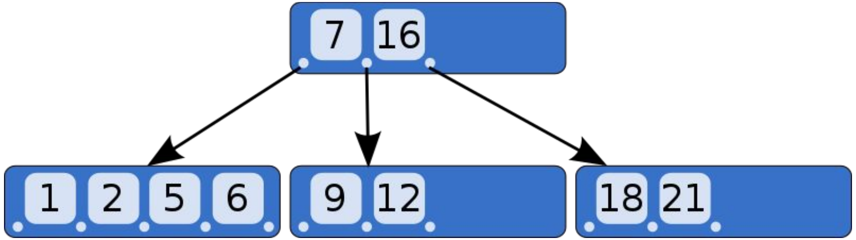




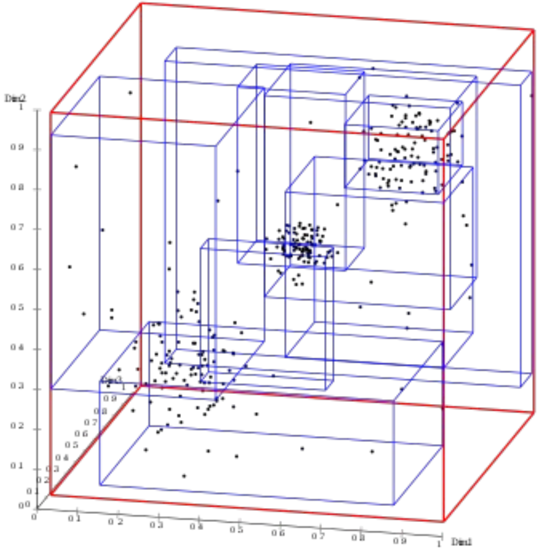
Using local labels to update global labels for all clusters

Optimization on partition——R tree

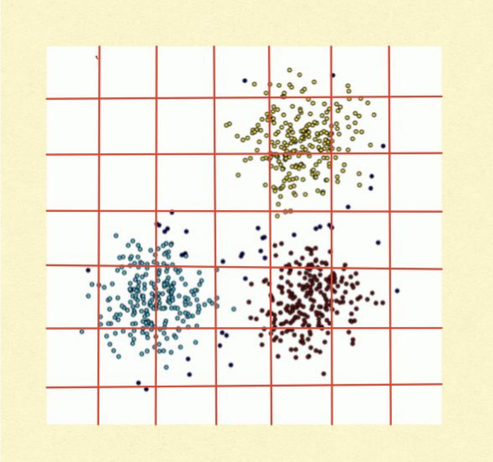
B-tree：
A self-balancing tree data structure

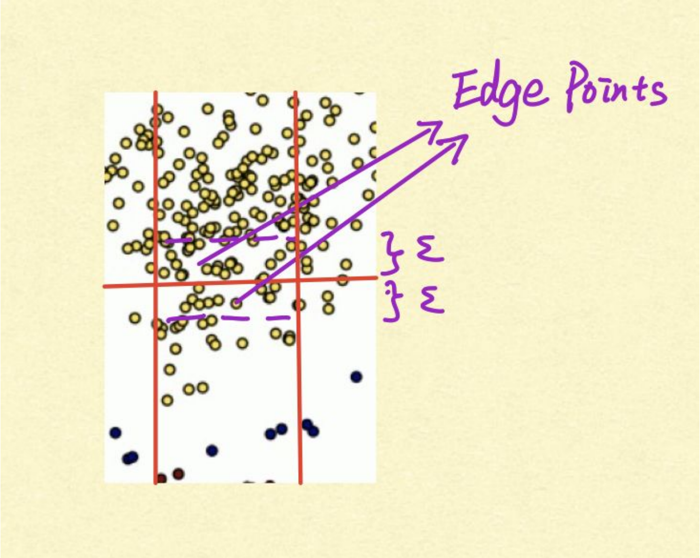


R-tree：
A tree data structures used for spatial access methods



Optimization on partition——R tree (Boundary-based Strategy)

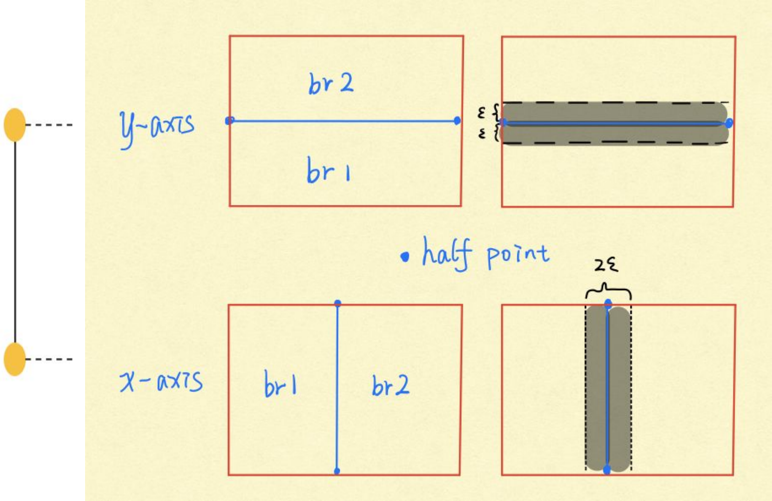




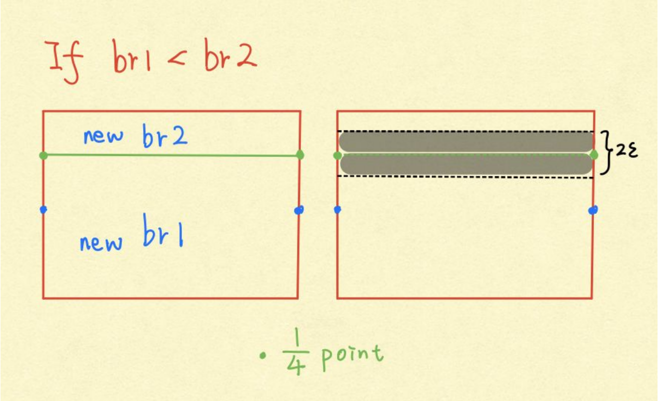
- Reduce the amount of edge points
- Make sure workload of each worker as balanced as possible

Optimization on partition——R tree (Boundary-based Strategy)

Set max points number in partition - maxpoints



As for y-axis:



minimize **Score = |# in br1 — # in br2| * (# in Black part)**

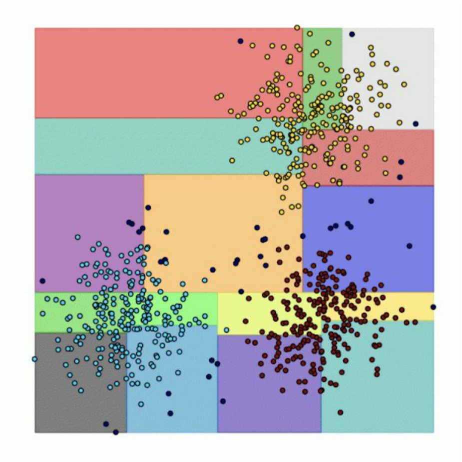
Optimization on partition——R tree (Boundary-based Strategy)

Max points number in partition - maxpoints

x-axis → x_minScore

y-axis → y_minScore

smaller one -> divide



Put final br1 & br2 to queue（right hand）

Take out the leftmost mbr in the queue in turn to do the above calculation

Optimization on partition——R tree (Cost-based Strategy)

Goal: Make the workload (# of points) of partitioned region as balanced as possible.

Input: **S** (The rectangle to be split)

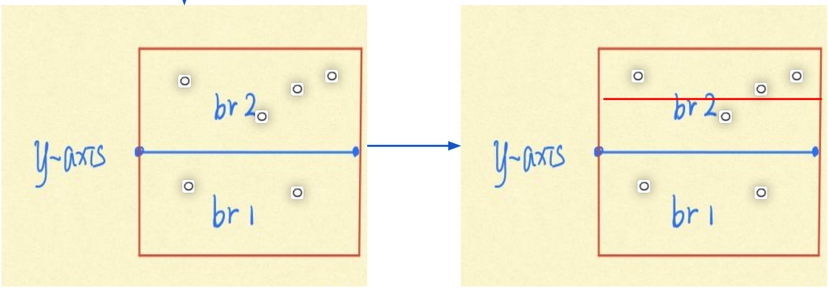
1. Min Cost Differience ← ∞

2. Calculate the cost difference between S1 and S2 ***(after applyng the selected split line)***

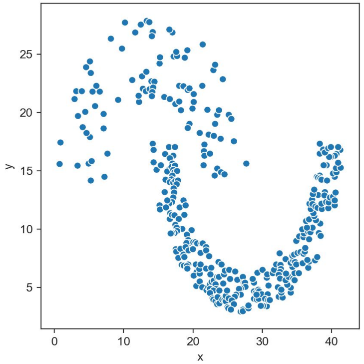
3. Split line + eps * 2 beyond the boudary

   (Get the best split line from the record!)

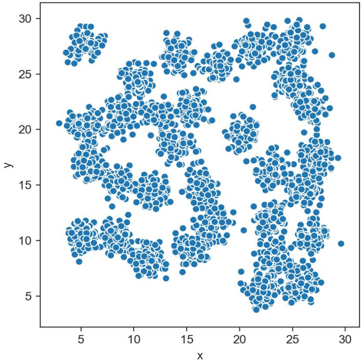
Output: **S1** and **S2** such that **S1 ∪ S S2 = S**



Conclusion



Jain



D31

Partition Number	**4**	**8**	**10**	**Partition Number**	**4**	**8**	**10**
Naive	1179ms	797ms	808ms	**Naive**	26410ms	19415ms	18483ms
Parallel Matrix DBSCAN	422ms	385ms	380ms	**Parallel Matrix DBSCAN**	6730ms	5590ms	4564ms
Rtree - rbp	360ms	330ms	289ms	**Rtree - rbp**	7993ms	5722ms	5328ms
Rtree - cbp	308ms	284ms	271ms	**Rtree - cbp**	6977ms	5652ms	5164ms

Conclusion

- Performance
- Sclability
- Bottle neck (Master)
- Know your data

Future Work:

Now: using number of points to measure workload when doing partition

Future: using the real cost (maybe considering density) of DBSCAN to measure workload

More expreiments on large volume datasets with large partitions

Partition Number	4	8	10	Partition Number	4	8	10
Naive	1179ms	797ms	808ms	**Naive**	26410ms	19415ms	18483ms
Parallel Matrix DBSCAN	422ms	385ms	380ms	**Parallel Matrix DBSCAN**	6730ms	5590ms	4564ms
Rtree - rbp	360ms	330ms	289ms	**Rtree - rbp**	7993ms	5722ms	5628ms
Rtree - cbp	308ms	284ms	271ms	**Rtree - cbp**	6977ms	5652ms	5264ms

Reference

[1] DBSCAN Wikipedia. https://en.wikipedia.org/wiki/DBSCAN

[2] Guttman, A. (1984). R-trees: a dynamic index structure for spatial searching. ACM SIGMOD International Conference on Management of Data.

[3] Huang, Fang &. Zhu, Qiang &. Zhou, Ji &. Tao, Jian &. Zhou, Xiaocheng &. Jin, Du &. Tan, Xicheng &. Wang, Lizhe. (2017). Research on the Parallelization of the DBSCAN Clustering Algorithm for Spatial Data Mining Based on the Spark Platform. Remote Sensing. 9. 10.3390/rs9121301.

[4] He, Yaobin &. Tan, Haoyu &. Luo, Wuman &. Feng, Shengzhong &. Fan, Jianping. (2014). MR-DBSCAN: a scalable MapReduce-based DBSCAN algorithm for heavily skewed data. Frontiers of Computer Science. 8. 10.1007/s11704-013-3158-3.

[5] Chernishev, G &. Smirnov, K &. Fedotovsky, P &. Erokhin, G &. Cherednik, K. (2013). To Sort or not to Sort: The Evaluation of R-Tree and B+-Tree in Transactional Environment with Ordered Result Set Requirement. SYRCoDIS.

[6] Clustering basic benchmark. http://cs.joensuu.fi/sipu/datasets/

[7] Phnix-wu, Full analysis of spatial data index RTree, https://blog.csdn.net/wzf1993/article/details/79547037, 2018/03/12.