Research and Implementation of Query Privacy Protection for Spatio-temporal Data Based on Spark

Lin Qian[a], Hengmao Pang[b], Zhu Mei[c], Jun Yu[d], GuangXin Zhu[e], Min Bu[f], Lin Wang[g]

State Grid Electric Power Research Institute (SGEPRI) Nanjing, China

[a]qianlin@sgepri.sgcc.com.cn, [b]panghengmao@sgepri.sgcc.com.cn,
[c]meizhu2016@aliyun.com, [d]yujun@sgepri.sgcc.com.cn,
[e]zhuguangxin@sgepri.sgcc.com.cn, [f]bumin@sgepri.sgcc.com.cn,
[g]wanglin18@sgepri.sgcc.com.cn

Abstract. The traditional CPIR algorithm needs to scan the whole data space with a large amount of computation so it's not suitable for big data. This thesis proposes that the parallel grouping range privacy query algorithm based on Spark. The range privacy query algorithm divides the grid into different groups in order to reduce the amount of computing and parallellzing computation to improve the efficiency based on Spark. It has a big improvement on server execution time, client execution time and communication cost.

1. Algorithm overview

In recent years, cloud computing technology and location-based services technology have made great progress. Cloud computing [1] can be rapidly and automatically expanded to support mobile data stream query processing [2] demand for dynamic change of scale, response time, processing capacity, and so on [3]. Therefore, location-based services are gradually enriched in people's lives, and people pay more attention to personal privacy. However, there is a gap between the degree of privacy protection and the availability of services [4], therefore, this paper studies the protection algorithm of spatial object privacy query based on CPIR [5]. Spatial Range Query Algorithm refers to a certain time or a certain period of time. Range query, nearest neighbor query and K nearest neighbor query are the most basic query types within the range of spatial index. The scope query is also the basis of other queries, including the nearest neighbor and K-nearest neighbor queries, which all find one or several satisfying spatial objects in the range by limiting a smaller range, and then enlarge the range to find the result if no result is found.

This paper is based on the current situation of CPIR protection of privacy query and spatial scope privacy query [6], and the cost of query is based on the analysis of the local space range query algorithm based on private information retrival (NSRQ-PIR). The server has a long calculation time. It is not suitable for the Spark-based CPIR spatial range query privacy protection query algorithm for the lack of large data volume. A Spark-based parallel privacy retrieval packet range query algorithm (Parallel grouping space range query algorithm) is proposed. Based on private information retrival, PGSRQ-PIR), further optimization of the naive algorithm is achieved by partitioning the spatial grid [7].

In the environment of Spark, it is necessary to divide and group the space to apply CPIR algorithm to the spatial range query. In the columnar database, there are a lot of redundancy calculations in the

traditional mesh partition strategy. Therefore, the large mesh is divided into several groups of small mesh by the grouped mesh partition [8], so as to reduce the computational cost and communication cost of the server and client. In the process of meshing, the formula for calculating the amount of data can be expressed in Formula 1, where $Max_{cell}$is the maximum number of objects in each grid in the grid when meshing. g_x is the number of grid columns, g_y is the number of grid rows, n_x is the span of the range of columns. The communication cost is related to the number of rows g_y and Max_{cell} .Therefore, it is necessary to synthetically measure the grid partitioning strategy. In order to achieve a balance between the computational cost and the communication cost, this paper adopts the following partitioning strategy to minimize the number of rows, and divides the query grid into different groups as far as possible, thus implementing Spark-based parallelism. This can minimize the computational cost and communication cost. The calculation method of the cost can be expressed by the formula 2, where g_y is the number of columns of all the calculated groups and Max_{cell} is the maximum number of spatial objects in each grid at the time of mesh division. I is a constant when the secondary residual modulus is constant, g_y indicates the total number of lines.

$$C_{\text{compute}} = \text{Max}_{\text{cell}} \times g_x \times g_y \times n_x \tag{1}$$

$$C_{\text{communication}} = \text{I} \times \text{Max}_{\text{cell}} \times g_y \tag{2}$$

2. algorithm basic ideas

Figure 1 shows the basic calculation flow of the PGSRQ-PIR algorithm on the server. After receiving the query, the server first groups the data. The grouping method is obtained from the client. After the data space is grouped, the result is calculated. The packets are sent to the cluster for CPIR calculation. In Figure 1, the two sets of data blocks are grouped. The two groups of data blocks are executed in parallel. This step is the core of the PGSRQ-PIR algorithm. Finally, the result is sent to the client.

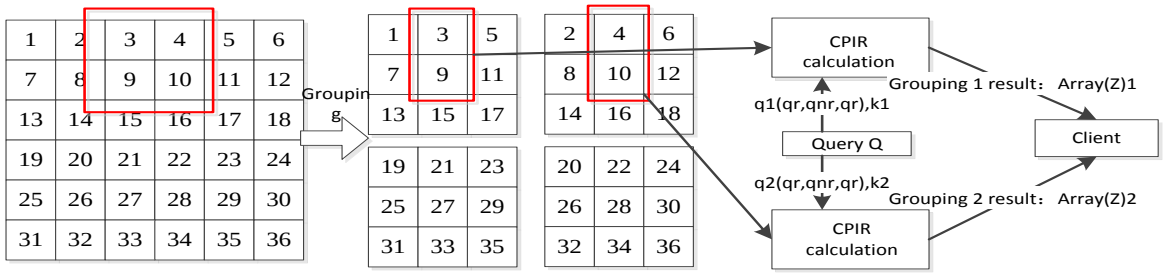

Figure 1. The flow of PGSRQ-PIR

Figure 2 is an architectural diagram showing the operation process of PGSRQ-PIR on the Spark parallel framework. During the query, in the first time, get the matrix of the query from HBase [9] to RDD [10], after the data packet through flat Map operation. After grouping, each packet is assigned to Spark's Executer to perform the CPIR algorithm. The queries q and k corresponding to each group are obtained in the shared variable of Spark, This shared variable query Q has a copy on each Executer and is broadcasted through Spark's broadcast function. After that, each Executer calculation of the packet data will return an array of Z, the size of the array and the number of rows of the group are equal. In this way, the arrays are returned to the client, and the client can obtain the result of the range query performed by the second remaining calculation.

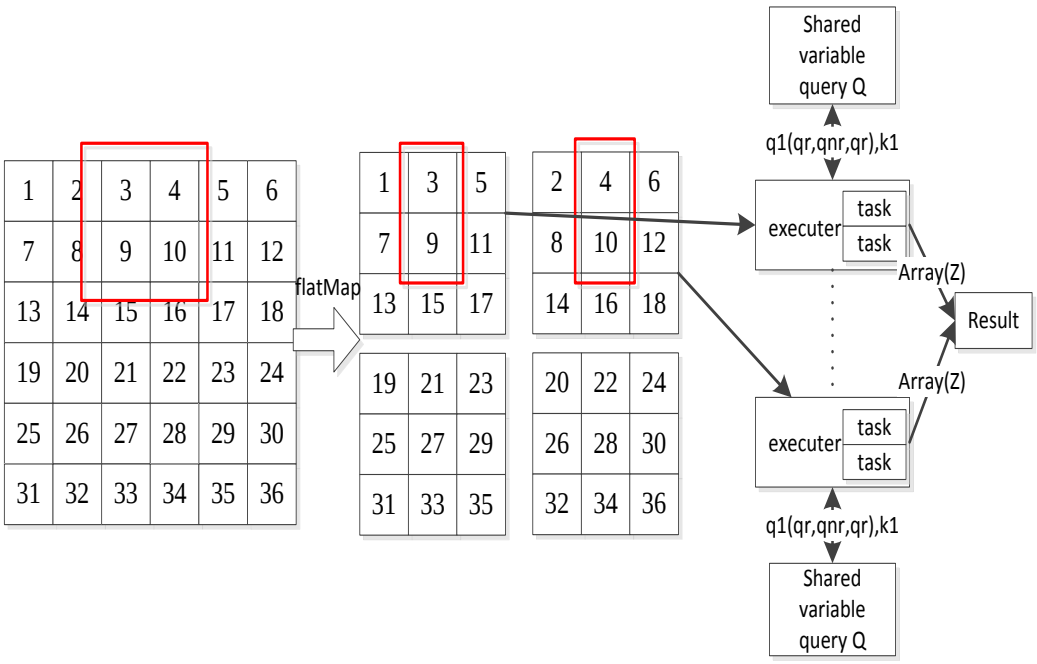

Figure 2. The architecture of PGSRQ-PIR based on Spark

3. Algorithm implementation

Algorithm 1 Spark grouping space range query algorithm at the client PGSRQ-PIR

Input: query range, number of grid rows and columns gx, gy, query space, minimum number of rows per group Miny, minimum number of columns Minx
Output: query result
Algorithm Description:
1: Calculate the row and column coverage of the covered grid based on the query scope xstart, xend; ystart, yend;
2: According to the coverage of the row and column, the column is divided into nc=xend-xstart+1 group, and the row is divided into nr=gy/(yend-ystart+1) group;
3: if(nc>gx/Minx) then
4: nc=gx/Minx;
5: endif
6: if(nr>gy/Miny) then
7: nr=gy/Miny;
8 endif
9: range=nc;
10: ngc=gx/nc;
10: for(k=0;k<nc;k++)
11:Generate $q\left(q_1, q_{1+ngc*range+1} \cdots\cdots q_{1+ngc*(nr-1)*range+1}\right)$ in which there must be a q in the query range, corresponding to the subscript $q_t \in QNR$,and the other subscripts correspond to $q_t \in QR$;
12: send q,nc,k,nr to server ;
13: end for
14: Waiting for the server to return the result;

Algorithm 1 Spark grouping space range query algorithm at the client PGSRQ-PIR

15: Calculate the second remaining result of the result, and obtain the value of the query bit;
16: End.

The algorithm first inputs the size of the parameter space mesh, including the number of rows gy and the number of columns gx. And in order to prevent the snooping of the scope of the customer's query, use Minx and Miny to specify the minimum number of columns and rows per group. Line 1 of the query client algorithm first obtains the range of the grid within the range through the scope of the query. Line 2 finds the optimal number of groups based on the range. The optimal number of tasks in this paper is the column span of the grid within the range, namely xend-xstart+1 group. This allows different meshes to be grouped into different groups. On the line, try to make the grids in the range on the line all in the same group. That is gy/ (yend-ystart+1) group. Lines 3-8 limit the user input parameters and convert the intra-group grids into groups that do not meet the user's security requirements. Lines 10-13 indicate that the secondary residual sequence q for generating a query for each packet is sent to the server along with the partitioned scheme. Lines 14-16, receive the data returned by the server and obtain the result through the second remaining calculation

Algorithm 3.2 Spark grouping space range query server algorithm GSRQ-PIR

Input: Multiple sets of queries, each of the groups contains (query q, column grouping nc, row grouping nr, group k value of query q)
Output: Array(Z) value sent to the client
Algorithm Description:
1: Obtain the CPIR matrix data from HBase and store it in the RDD.
2: Group the data in the RDD according to nc, nr;
3: Spark performs CPIR calculation on the query q corresponding to the k value for each group by k value;
4: Spark summarizes the results and returns them to the client;
5: End.

In the frist time, the algorithm obtains the query content and the grouping strategy sent by the query client. And broadcast multiple sets of queries through Spark's variable broadcast function. Ensure that each node of Spark can get the content of the query. The first line of the algorithm reads the data stored in HBase. Convert to Spark flexible data set RDD for RDD transformation (Action) and action (Action) after storage. The second line is a one-to-many conversion of RDD through a conversion function flat Map of RDD. The second line is a one-to-many conversion of RDD through a conversion function flat Map of RDD. Mainly divide each row of data into groups according to the client's policy. It is divided into nr groups on the line, divided into nc groups on the column, and converted into a new RDD. The third line of the algorithm mainly performs the calculation of the CPIR algorithm. The corresponding q and k are obtained from the broadcast variables mainly by the labels of the group. Therefore, using q to perform CPIR calculation of data grouping, the Z value of the bank group is finally obtained. The algorithm finally summarizes the results of the group calculations and sends them to the client through the RDD reduce action.

4. Performance analysis and experimental results

NSRQ-PIR will convert each range query into multiple single-grid queries. The number of queries in the range of grids, The complexity of the algorithm for deriving NSRQ-PIR from Equation 1 is $O(nc \times g_x \times g_y \times Max_{cell})$, nc is the number of columns in the grid occupied by the query, g_x is the number of columns of the grid, $g_y$is the number of grid lines, Max_{cell} is the largest number of spatial objects in each grid, Therefore, the server time of NSRQ-PIR increases linearly with the increase of the number of grids included in the range. The number of returned results of NSRQ-PIR is $nc \times g_y$, the number of results is the number of columns occupied by the grid of the query multiplied by the number of rows of the query matrix. Figure 2 shows the results of the NSRQ-PIR query, Assuming that only 1 Bit of information is stored in each grid, Then the NSRQ-PIR query result of the CPIR matrix graph on the left of Figure 2 can be represented as the graph in Figure 3, and the query is also a row 1-2, 3-4 column, $q1(QR,QR,QNR,QR,QR,QR)$ and $q2(QR,QR,QNR,QR,QR,QR)$ are query sequence, Query the information in

columns 3 and 4 respectively. Z1(QNR,QNR,QR,QNR,QNR,QNR) And Z2(QNR,QR,QR,QNR,QR,QNR) is the result for two queries The PGSRQ-PIR algorithm groups range queries and data. The number of rows and columns of the query matrix are reduced. Thereby reducing the amount of data and communication costs of the calculation, at the same time, the query is broadcast through Spark. And parallel processing different groups, Assign each group's calculations to each node of the cluster. Thereby further improving the efficiency of the calculation and reducing the computing time of the server. There is also a big advantage when the number of returned data is small in the range of the query. Figure 3 shows the results of the PGSRQ-PIR query. PGSRQ-PIR only performs CPIR query on 2 small packets. q1(QR,QNR,QR) And q2(QR,QNR,QR) are the lower side of the figure and they also are the two sets of query sequences. Z1(QNR,QNR,1) And Z2(QNR,QR,QR) are on the right side of the picture and they also are the result of the query. Finally, by comparison, the ratio of the calculation cost of NSRQ-PIR and PGSRQ-PIR is the ratio of the number of grids required for the query, which is $6\times6+6\times6=72:\ 3\times3+3\times3=18=4:1$, the ratio of communication costs is the ratio of the number of returned results, which is $(6+6):(3+3)=2:1$.

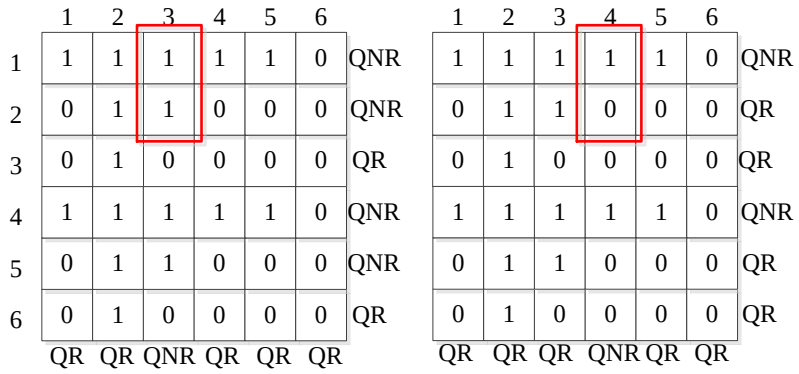

Figure 3. Schema of NSRQ-PIR result

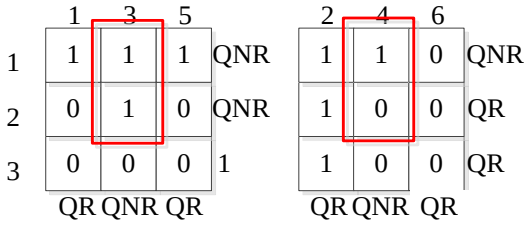

Figure 4. Schema of PGSRQ-PIR result

The security of the algorithm is directly related to the minimum column value Miny entered by the user. Security increases as Miny increases. When it comes to $\text{Miny}=g_y$, the safety reached the same level as CPIR. Assuming that the intensity of CPIR is 1, then the security strength of PGSRQ-PIR can be expressed by Equation 3.

$$S=\frac{\text{Miny}}{g_y} \tag{3}$$

Figure 5, Figure 6, and Figure 7 show the effect of different query ranges on uniformly distributed dataset range queries. The formula for CPIR calculation uses Equation 1.

Figure 5 is a comparison of server time between different single-scope scope query (NSRQ-PIR) and Spark-based privacy search packet range query algorithm (PGSRQ-PIR) under different query ranges. It can be seen from the figure that the simple range query algorithm and the Spark-based privacy search

group-wide query algorithm will slow down as the query range grows larger. But PGSRQ-PIR due to the grouping strategy and parallel computing of Spark, the calculation time is always lower than NSRQ-PIR, And as the scope of the query grows, the advantages of PGSRQ-PIR become more and more obvious. This is because the data in the first query range is divided into different groups, and different groups are executed in parallel on the Spark cluster, and the data volume of each packet is effectively compressed and reduced, which improves the calculation speed. With the increase of the range, the time growth of PGSRQ-PIR is very slow, which fully reflects that under the current test scale, Spark cluster has better scalability and does not increase significantly with the increase of the range.

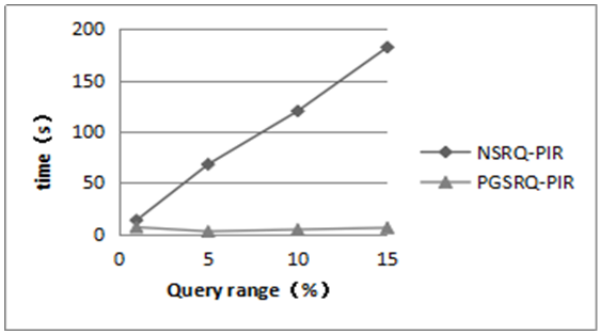

Figure 5. The server time of different query ranges

Figure 6 is a comparison of communication costs between a simple spatial range query (NSRQ-PIR) and a Spark-based privacy search packet range query algorithm (PGSRQ-PIR) in different query ranges. The communication cost refers to the number of QR and QNR in the result returned by the server. The minimum number of rows used in this experiment is the number of rows in the range to be queried plus 1.It can be seen from the figure that the communication cost of the two algorithms will increase with the increase of the range, but PGSRQ-PIR can achieve a relatively small communication cost by splitting the query range into groups. This picture can reflect the trend of analysis well.

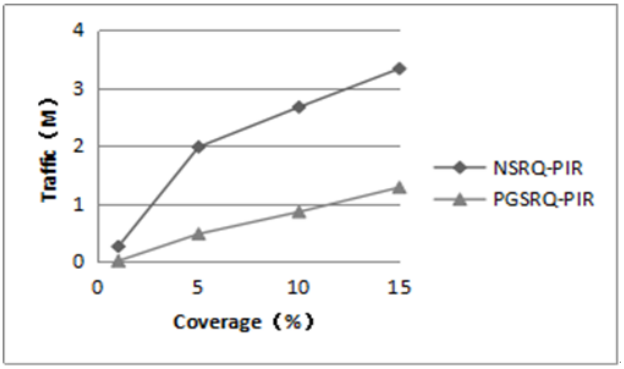

Figure 6. The communication cost of different query ranges

Figure 7 is a comparison of the client computing time between the simple spatial range query (NSRQ-PIR) and the Spark-based privacy search packet range query algorithm (PGSRQ-PIR) under different query ranges. The grouping in the row direction by PGSRQ-PIR can reduce the number of results returned, but the client only decrypts one of the required rows, and the number of rows decrypted by the two is the same, so the calculation time is basically the same.

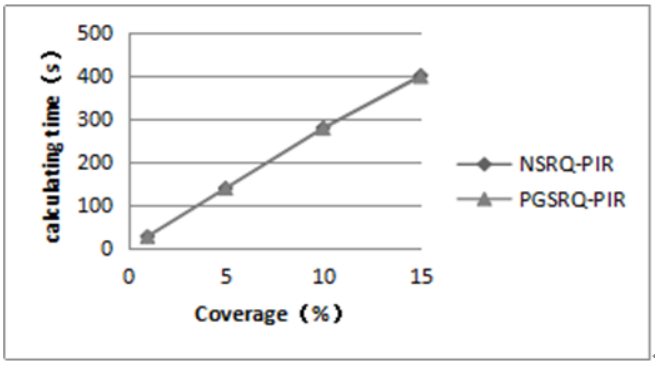

Figure 7. The client time of different query ranges

5. Conclusion

In this paper, we propose a range privacy query protection algorithm based on Spark and CPIR. Compared with the simple range query, this algorithm reduces the cost of client computation and communication, and has two orders of magnitude improvement in the server query time. Because PGSRQ-PIR algorithm is a parallel design algorithm based on Spark, there is room for performance improvement in redundancy computation.

Acknowledgments

This work was financially supported by the State Grid Corporation of Science and Technology (WBS number: 521104170019).

References

[1] LI Rui-xuan,DONG Xin-hua, GU Xi-wu,ZHOU Wan-wan,WANG Cong. Overview of the data security and privacy-preserving of mobile cloud services [J]. Journal on Communications, 2013, 34 (12): 158 - 166.

[2] Xie D, Li F,Yao B,et al. Simba:Efficient In-Memory Spatial Analytics. ACM SIGMOD International Conference on Management of Data . 2016

[3] WANG Lu, MENG Xiao-Feng. Location Privacy Preservation in Big Data Era: A Survey [J]. Journal of Software, 2014, 25 (4): 693 - 712.

[4] Pan Xiao,Hao Xing,Meng Xiaofeng. Privacy Preserving Towards Continuous Query in Location-Based Services [J].Journal of computer research and development. 2010 (01).

[5] Settimi V. A Study of Computational Private Information Retrieval Schemes and Oblivious Transfer [J]. 2007.

[6] Wei W, Xu F, Li Q. Mobishare: Flexible privacy-preserving location sharing in mobile online social networks [C]//INFOCOM 2012 Proceedings IEEE. IEEE, 2012: 2616 - 2620.

[7] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system [C]// Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, 2010:1 - 10.

[8] Pan X, Xu J, Meng X. Protecting location privacy against location-dependent attacks in mobile services [J]. Knowledge and Data Engineering, IEEE Transactions on, 2012, 24 (8): 1506 - 1519.

[9] Taylor R C. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics.[J]. Bmc Bioinformatics, 2010, 11 suppl 12 (6): 3395 - 3407.

[10] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing[J]. In-Memory Cluster Computing". USENIX Symposium on Networked Systems Design and Implementation (NSDI, 2012, 70 (2): 141 – 146.