Cluster Comput (2017) 20:2833–2844
DOI 10.1007/s10586-017-1081-3

Data storage optimization strategy in distributed column-oriented database by considering spatial adjacency

Kun Zheng[1] · Danpeng Gu[1] · Falin Fang[1] · Miao Zhang[1] · Kang Zheng[1] · Qi Li[1]

Received: 14 March 2017 / Revised: 25 July 2017 / Accepted: 27 July 2017 / Published online: 8 August 2017

Abstract Scan operation will involve many fragments and cause many extra invalid partitioning query operations in distributed column-oriented database which affects query efficiency seriously, especially for spatial data. To solve this question, this paper refers to partitioning strategy in distributed column-oriented database and advocates a spatial data storage optimization strategy named 'SPPS'. This strategy makes adjacent spatial objects stored in the same data fragment with considering spatial adjacency, and reserves the spatial information of each fragment. Thus spatial query operation can locate the relevant fragment on basis of spatial information of fragment, and extra invalid partitioning scan operations would be lighted. Then the storage and query efficiency would be improved. To verify the validity of 'SPPS' optimization strategy, this paper carries on relevant experiments based on HBase and records spatial query efficiency with and without 'SPPS' respectively. The experiments results indicate that 'SPPS' strategy can optimize the storage and query efficiency in distributed column-oriented databases.

Keywords Partitioning · Spatial data · Spatial adjacency · Distributed column-oriented database

✉ Falin Fang
cugwhlin2014@126.com

Kun Zheng
michael_power@sohu.com

[1] Faculty of Information Engineering, China University of Geoscience (WuHan), Wuhan 430074, China

1 Introduction

With the rapid development of sensor technology and network technology, there has been a massive boost in the spatial data. Besides, timely analysis and management of such spatial data become a major challenge in current GIS field [1–3]. To face the challenge, a kind of NOSQL database which named distributed column-oriented database system becomes a main solution [4–7]. In distributed column- oriented database, data is partitioned into multiple fragments with range partitioning or consistent hash partitioning [7–9]. For example, HBase [10] and Bigtable [11] distribute datasets by the range of their keys; Dynamo [12] and Cassandra [13] choose consistent hashing as their partitioning strategies. Whether adopting range partitioning or consistent hash partitioning, it will usually lead to the workload skew of the system, and the data request operation of reading and writing can be focuses on a single node, which decreases the efficiency of distributed system. This paper focuses on range partitioning, and try to optimize the spatial data storage in distributed column-oriented database. Usually range partitioning approach adopts schema-based strategy aiming to generate equi-sized partitions [14], which neglects the characteristic of data. Besides, it provides scan query with start and end primary key [15]. This scan operation may involve some inconsistent data, and cause many invalid fragment queries [6]. Most importantly the invalid fragment scan query will become more frequent with the increase of the number of fragments. Thus spatial query efficiency would be decreased seriously in the distributed column-oriented database [14,16,17].

Different from the general data, spatial data has strong regularity, mainly in the performance of geographical space distribution, that is to say, there is a certain spatial distance relationship among spatial objects, either adjacency or sep-

aration [18]. If we can fully consider these characteristics of spatial data in the process of partitioning, and store spatial adjacent objects into the same data fragment and spatial separate objects into different data fragments, thus to reduce the number of query operation crossing fragments and nodes, which means the number of invalid partitioning scan operation would be decreased. For example, in order to search hotels near Optics Valley, Wuhan, China on the global scale, because the data is stored in a plurality of data fragments by a random method in spite of such a small range of queries, the query operation may involve multiple fragments and lead to lower query efficiency. If we store the information of hotels near Optics Valley into one fragment, and equip the fragment with the spatial position information, the query operation can be implemented on the fragment directly on basis of the spatial information, thus to decrease the system workload of query operation and increase the query efficiency in distributed column-oriented database to a certain extent.

Based on this, this paper presents a data storage optimization strategy named spatial proximity partitioning storage (SPPS). SPPS optimization strategy takes the spatial proximity into account and records the spatial position information in the partitioning process. The most common query method is scan query in distributed column-oriented database. It will retrieve the data in each fragment due to get fragment position information according to the start and end primary key [19]. Thus a question will come out which caused by the operation refers to "division of primary key". This operation can be concluded as below: Each spatial object will be stored into the distributed column-oriented database according to lexicographic order of primary key. Then it divides the primary key and form many fragments in the process of partitioning. So some spatial adjacent objects would be stored into different fragments in the process of partitioning. Then some fragments which do not contain our query result would belong to the range of start and end primary key. Even though some other spatial data storage methods on basis of distributed column-oriented database consider spatial proximity, they will get some uncorrelated fragments using start and end primary key merely if they do not consider the "division of primary key". The SPPS strategy can locate the exact fragments which contain query result with the help of spatial position information contained by fragments. It avoids the question of getting fragment position according to start and end primary key merely. To realize it, our 'SPPS' strategy contains four parts. First, it obtains the spatial distribution principle of spatial data with adopting spatial sampling approach to analyze spatial data, and divides the whole space into many subspaces according to the spatial distribution. Second, we design the partitioning key of each fragment. Then we establish the mapping function relationship between the subspace and the partitioning key. Finally, to store the spatial adjacent objects into the same fragment, we design the primary key based on the partitioning key of fragment, so the spatial position information of fragment can be implied by the primary key. Owning to 'SPPS' strategy, spatial query can be directly located on the data fragment according to spatial mapping relationship and the value of primary key, then it can decrease the number of invalid partitioning query operations and enhance the efficiency of spatial query in the distributed column-oriented database.

2 Related work

In order to optimize the storage of spatial data in distributed column-oriented database and reduce the influence of spatial query caused by invalid partitioning scan, extensive research has been conducted by scholars. Existing methods can be classified into two categories on basis of whether or not considering space partitioning in the data partitioning process. First, it just considers how to implement a relatively more reasonable partitioning method which we called "Data partitioning strategy". Second, it considers the spatial adjacent property, and designs some space partitioning methods which we called "space partitioning strategy". Detailed exposition will be descried as below.

2.1 Data partitioning strategy

Data partitioning strategy holds that the partitioning methods in distributed column- oriented database is inefficiency and stupid and has shortcomings of hot spot problem and low performance. So many scholars want to design a more reasonable data partitioning method and optimize data storage and management in distributed column-oriented database.

A reasonable data partitioning method should disperse the partitioning fragments into system evenly so as to ease the hot spot problem and improve query efficiency. Chen et al. [7] took full account of the problems like the hot spot and inefficiency caused by the existing partitioning method in NoSQL database, and designed a more reasonable partitioning method which named HRCH which combines together the consistent hash and the range partitioning method. HRCH can improve the system's scalability, and avoid the hot spot problem as far as possible; Cruz et al. [20] proposed an automated workload-aware table segmentation mechanism in order to select a reasonable primary key division point for implementing data partition process. It takes full account of the workload of the system and makes the data fragments of the table be more evenly distributed into the cluster, so the workload of searching can be evenly allocated to each node. This way it can lead to a better overall load balance across

regions, and the query efficiency can be improved. Ye and Li [21] considered the question of request skew, and proposed a heterogeneous distributed storage system based on Cassandra [13] which is well known as a NoSQL database. It can reduce total time-consuming of read requests and write requests to a certain extent, also the storage utilization of each node can be balanced after applying this distributed storage system; Elghamrawy [22] proposed an adaptive-rendezvous hashing partitioning module named ARHPM. It can make up the drawback with which Cassandra ignores the nature of nodes in the process of assigning data to Cassandra nodes, and optimize the partitioning results in Cassandra database.

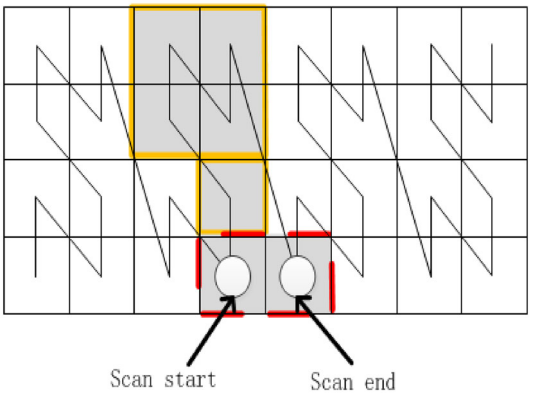


Fig. 1 Situation of invalid partitioning scan

2.2 Space partitioning strategy

This strategy tries to implement a reasonable space partitioning strategy on basis of the adjacent characteristics of spatial data, thereby reduce the number of invalid partition scan operations and enhance the efficiency of spatial data query [3,3,6,16,23–26]. To implement this target, many scholars adopt space filling curve to construct primary key. Space filling curve is a linearization technique transforming high dimensions to one dimension, and it can keep adjacent property for spatial data. Besides, it will store the spatial data according to the lexicographic order of primary key in distributed column-oriented database. So the adjacent objects would be stored as adjacent key-value pairs owing to the property of space filling curve and lexicographic order.

For example, Nishimura et al. [6] presented a multi-dimensions spatial data partitioning method and applied it into LBS (location-based services). This method adopts space filling curve to partition the space, and encodes the primary key of spatial object according to the result of space partitioning. Then, it stores the spatial data into HBase, and designs a multi-dimensions index construct combining index mechanism of HBase. Then the times of invalid partitioning scan can be decreased, and the query efficiency would be enhanced with this index structure. Also some scholars adopted geohash [27] in the construction of primary key [3,25,26,28,29]. For example, Le and Hong [3] adopted geohash [27] to construct primary key, and the spatial objects which contain the same prefix encoding value will store together in physical storage. To make up the limitations of geohash, it advocates BGRP tree which is based on R tree. Besides, some other scholars adopted other methods to consider adjacent proximately. Zhang et al. [5] adopted the grid index to partition the space, and established a spatial index which can decrease the operations of invalid partitioning scan and support high efficient query for spatial data; Vo et al. [16] presented a spatial partitioning framework, which is based on Mapreduce processing and has been demonstrated in Hadoop-GIS framework.

Nevertheless, most of them just consider the question of space partition instead of the partitioning storage strategy in distributed column-oriented database, which can result in invalid partitioning query in the process of scan operation in distributed column-oriented database as before. For example, a case of adopting geohash [27] to partition space and encode the spatial object, which is shown as Fig. 1.

In Fig. 1, each grid represents one fragment in distributed column-oriented database. And the red range represents actual query area. However, it will scan not only the range of red color but also unnecessary range of yellow color, which generates many invalid scan operations and decreases the efficiency of spatial query.

In our viewpoint, it is difficult to decrease the number of invalid partitioning scan operation by means of data partitioning strategy which separates the redundancy of invalid partitioning scan operation into each nodes. Besides, space partitioning strategy tries to partition the spatial data in an appropriate way, and designs the encoding value on basis of space partitioning results. Then it can avoid unnecessary partitioning scan operation to some degree. However, many space partitioning strategies presented by scholars does not consider the partitioning strategy of distribution column-oriented databases and lose the spatial information of each fragment, so the data in each fragment is still disorder and unsystematic. The invalid partitioning scan operations still exist in distributed column-oriented database, which influence the storing and query efficiency of spatial data. To avoid these, we store the spatial data of the same scope of area into same fragment, and record the spatial scope of each fragment by designing ‘SPPS’ strategy. It can locate the storage location which contains spatial data according to the spatial information of fragments. Then it can decrease invalid partitioning scan operations to a certain extent, and improve the efficiency of spatial data querying.

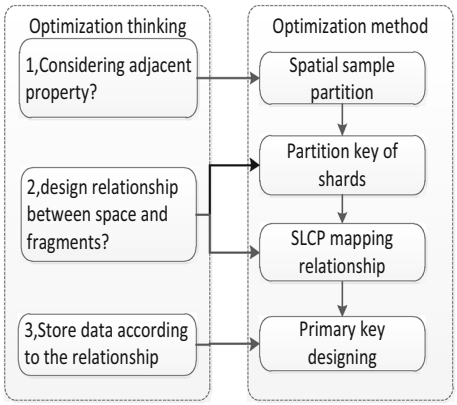


Fig. 2 Optimization thinking and method

3 Partitioning storage optimization strategy

As mentioned above, the most common method to deal with spatial data in column-oriented database is adopting space filling curve and geohash to consider adjacent of spatial data adequately. However this method cannot consider the adjacent character of spatial data among fragments, and fragments do not contain spatial message, so spatial query operation just can be carried out on basis of start and end primary key, which can generate many invalid partitioning scan operations and influence spatial query efficiency. However our optimization strategy which called SPPS partitions space with consideration of spatial distribution situation of spatial object, and confirms the spatial range of each fragment in the process of partitioning. Thus the fragment can contain enough spatial information to locate the specific fragment in the process of spatial query. Besides, we adopt Geohash [27] to consider spatial adjacent in each fragment. And spatial range scan querying can be conducted in each fragment which contains spatial objects separately. Then the invalid partitioning scan can be decreased, and the spatial query can be accelerated. Our optimization thinking contains three parts: First, how to consider spatial adjacent of spatial objects; second, how to design the function relationship between subspace and fragment; lastly, how to store spatial data according to this function relationship. To solve these questions, our 'SPPS' strategy contains four steps which can be showed as Fig. 2.

First, we will carry on the spatial sampling partitioning operation and get distribution principle of spatial objects; Second, we will conduct the design of partitioning key of fragment, and then define the mapping function relationship between subspaces and partitioning key which is called spatial longest common prefix (SLCP); Lastly, we will carry out the designing of primary key to store spatial object into the correspond fragment. Now we will present the optimization strategy in detail.

3.1 Spatial sampling partitioning

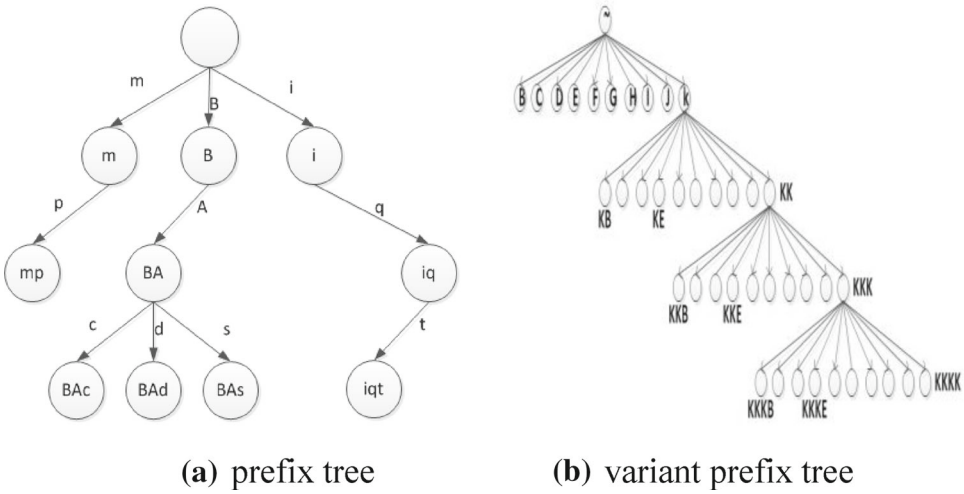
The partitioning strategy in distributed column-oriented database would generate unreasonable fragment distribution. Besides, the data in each fragment will be stored into database according to a random method which can lead to the disorder storage in every fragment. Therefore, we must consider how to disperse spatial data according to spatial distribution principle and make each fragment has an equal sum of spatial objects approximately in the process of spatial data storage. To achieve this, this paper adopts spatial sampling method to obtain the general distribution of spatial data. In the process of spatial sampling, we adopt random sampling method to analyze spatial object, and confirm the partitioning number combining the size of spatial data, lastly, spatial data is divided by STR [30] method on basis of the sampling distribution regularities. STR divides space range into a number of partitions in vertical direction, and then divides each vertical strip into several partitions in horizontal direction. It will insure all the partitions of spatial data is roughly equal. And the space range of each subspace can be obtained at the end of the space partitioning. Then, we will number each subspace following the column and row successively, and get a mapping relationship between number and subspace range, which can be showed as Formula (1)

$$< Number, Range < Point1, Point2 >> Formula1$$

In this formulation, Number is the serial number of subspace; $Range < Point1, Point2 >$ expresses the space range of subspace; $Point1$and$Point2$represent the lower left and upper right point of rectangular range. Thus, there will have a unique number which can map into each subspace, and we can ensure that the spatial data in the same subspace is adjacent. Most importantly, we can confirm the serial number of the subspace by the space range of each subspace. It will be used in the query process.

3.2 Partitioning key of fragment

Data is dispersed into cluster in fragmentation in the distributed column-oriented database. Each data fragment would have its own storage range. The storage range of each fragment is some segment in hash ring, when distributed column-oriented database adopts consistent hash partitioning method to disperse data. Otherwise, the storage range of each data fragment is the start and end primary key, when distributed column-oriented database adopts range partitioning method to disperse data. Whether adopt consistent hash or range partitioning, the data fragment of partitioning key should meet the following rules: first, ensuring there are no overlaps between data fragments of partitioning key domain. Second, the collection of all fragments is equal to the whole



(a) prefix tree **(b)** variant prefix tree

Fig. 3 The designing of partitioning key

of key value scope in size. Lastly, the fragmentation of the partitioning key value is orderly [8]. To ensure neighboring storage of spatial data, we should store spatial data into the distributed column-oriented database according to the result of space sampling partitioning. That is to say, we should store the spatial data in the same subspace into the same fragment.

To ensure this point, this paper designs the partitioning key of fragment on basis of prefix tree [31]. Prefix tree is a highly efficient searching tree, which is widely used in search engines. Each parent node encoding is the prefix code value of the corresponding child node in the prefix tree, and composition of the parent node codes usually contain different letters, each prefix encoding value of parent node is not the same as his brother's. It can be showed as Fig. 3a.

It is difficult to build the mapping relationship between partitioning key of data fragment and subspace when adopting prefix tree directory, because the prefix code values for each node of the tree lack certain regularity. So this paper will adopt variant prefix tree to design the partitioning key of fragment which is showed in Fig. 3b. The characteristics of the variant prefix tree can be concluded as two points: First, the encoding value of each father node will have the same letter with other father nodes, except for the last letter. Second, each layer will have ten child nodes in variant prefix tree, and only the tenth node in each layer has child tree nodes. Then the partitioning key will hold a certain regularity, and the mapping function relationship can be easily built through this variant prefix tree.

3.3 SLCP mapping relationship

We can confirm the serial number of subspace which indicates the location of spatial object through the mapping relationship which is shown as $< Number, Range < Point1, Point2 >>$. In addition, we can get the data range of each fragment after designing partitioning key of fragment. However, it lacks related function mapping relationship between them, which makes it difficult to confirm the storage location of fragment where the spatial object should store. Based on it, this paper designs a mapping relationship which called 'SLCP' code to confirm the relationship between serial number of subspace and data range of each fragment. It helps to form one-one mapping function relationship between subspace and data fragment. The specific approach can be concluded as follow: fetching the node encoding value according to the method of Layer traversal, and then building the 'SLCP' mapping relationship with the serial number of subspace. The 'SLCP' mapping relationship can be shown as Formula (2)

$$F1(M) = \underbrace{K \ldots K}_{(Number/10)} \; F2(Number\%10) Formula2$$

In this formula, the mapping relationship can be divided two parts. First part, we will build some same letters, and the number of same letters is $Number/10$. Second part, 'SLCP' mapping relationship will contain a function which is called $F2(Number\%10)$, and we will splice the mapping value when getting the value of two parts. Besides, $Number$ is the serial number of subspace, and $F2(Number\%10)$ is a very simple mapping function, which can be shown as formula (3):

$$F2(Number\%10) = \left\{ \begin{array}{l} < 0, B >, < 1, C >, < 2, D >, < 3, E >, < 4, F > \\ , < 5, G >, < 6, H >, < 7, I >, < 8, J >, < 9, K > \end{array} \right\} Formula3$$

In this formula, the return value will correspond to the node encoding value in the second layer of variant prefix tree.

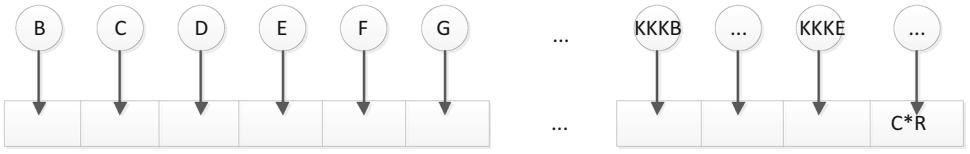


Fig. 4 SLCP mapping relationship

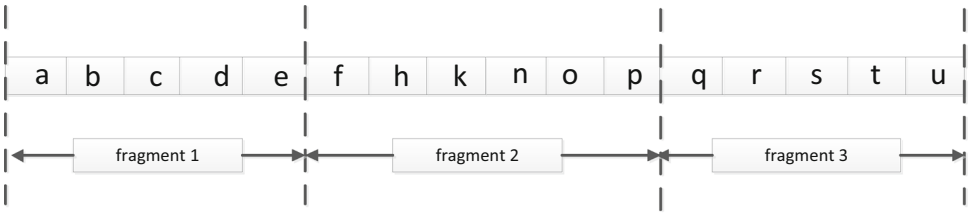


Fig. 5 Process of data partitioning

Lastly, the results of SLCP mapping relationship can be shown as Fig. 4.

Now, it forms a one-one mapping relationship between each subspace and fragment. Thus partitioning key of each data fragment can represent a certain space range, that is to say, each data fragment holds a certain space range, which can be used in the future spatial query. Besides, in the process of storing spatial data, we can confirm the serial number of subspace which is the spatial object belong to, then get the data range of data fragments according to the serial number and SLCP mapping relationship, and store spatial object into corresponding fragment according to spatial sampling partitioning.

3.4 Designing of primary key

Primary key is the uniqueness identifying of spatial objects in distributed column-oriented database. It not only has the capacity of rapid data retrieving, but also decides the storage location of spatial data in distributed column-oriented databases [32,33]. So how to design a reasonable primary key for each spatial object is very important. In this paper, we should ensure that spatial object should be stored into distributed column-oriented database according to 'SLCP' mapping relationship. In the process of spatial data storage, it will sort the data according to the primary key of lexicographic order, and then partition the primary key according to default partitioning method. Lastly, the spatial data will form some data fragments and disperse them into distributed column-oriented database. The data partitioning process can be shown as Fig. 5.

Based on it, 'SPPS' optimization strategy confirms the primary key range of each data fragment based on the mapping encoding value of 'SLCP', and to support spatial query, we adopt geohash encoding method to build primary key of each spatial object. Besides, to differentiate every spatial object, we also use the unique encoding value of spatial object, so the designing of primary key can be divided into three parts in this paper, which is shown as Fig. 6.

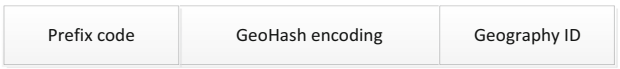


Fig. 6 Designing of primary key

In Fig. 6, the first part is prefix encoding value which is constituted by the partitioning key of data fragmentings. It would make spatial object stored into corresponding data fragmentings. The process can be expressed as below, firstly we should estimate the serial number of subspace where the spatial object locates, and then confirm the range of partitioning key corresponding to data fragment according to 'SLCP' mapping relationship, lastly, employ the start primary key of partitioning key as prefix key of spatial object which can make spatial object stored into data fragmentings on basis of spatial partitioning rule. The second part is spatial encoding which employs geohash encoding to decrease the dimensions of geographical spatial location information from two to one, thus to support spatial range query operation. The third part is the unique geographical ID which can identify the role of geographical object. On the whole, it can consider adjacent property of spatial data in the process of partitioning, make each fragment of spatial data in a certain space range, and keep the spatial range information of each data fragment. Then we can locate related data fragments according to the spatial range information of data fragment, so the scan query can be conducted in specific fragment which contains spatial object we searched,and invalid data fragments scan operation can be decreased, and the query efficiency of spatial data can be increased.

4 Experiments

To investigate the effect of SPPS's optimal decisions, we design the experiments based on Hbase, and compare the storing efficiency between our 'SPPS' strategy and default strategy firstly. In first part, we take HBase default partitioning strategy to store spatial data, and the primary key is

composed by Geohash and geography ID. In second part, store the spatial data using our 'SPPS' optimization strategy. The process can be shown as the follows:

- Sampling and analyzing the spatial data, and then obtaining the distribution rule of spatial data. Lastly, finishing the process of splitting spatial area according to spatial distribution rule.
- Computing the subspace where the spatial object locates according to the coordinates of each spatial object, and getting serial number of subspace.
- Confirming prefix-code of data fragment where spatial objects store, and computing the geohash value of this spatial object. Then, get the primary key of spatial object contains prefix encoding, object's geohash and object's ID.

After finishing the storage efficiency comparison of spatial data, we will compare query efficiency between 'SPPS' storage strategy and default storage strategy by implementing some spatial query experiments in HBase.

4.1 Experimental environment

In our laboratory, we have deployed an HBase cluster, which is consisted of two master nodes and six slave nodes and each node equipped with 12GB of memory and 4 core CPU. Besdises, there are 100 million spatial points in our experiment data whose size is about 40GB.

4.2 Results and discussion

- **Case one: comparison of storing efficiency**

In the processing of storing spatial data, we take parallel framework–MapReduce to import the spatial data, and compare the cost time of these two storage strategies (as shown in Fig. 7).

Rectangle within blue color shows the time consumption of SPPS store pattern, while red color shows that of the default store pattern. As we can see, time consumption of 2651s showed in blue is less than that of 2728s showed in red, so SPPS is more efficient in storing spatial data.

- **Case two: comparisons of database querying efficiency**

To demonstrate the efficiency of 'SPPS' optimization strategy, we have compared query efficiency of this two storing strategies include default storing strategy and 'SPPS' storing strategy. We conduct common range query using scan query method and record the RPC numbers of querying as an index, different size of data returned by RPC, and time consumption of querying. Additionally, spatial query operation would be divided into two parts caused by the different situations of query ranges which can be shown as Fig. 8.

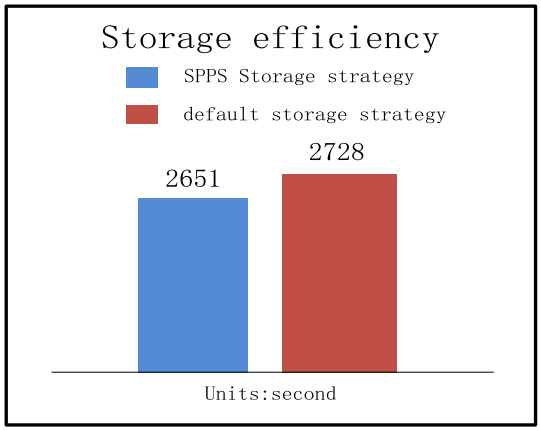


Fig. 7 Comparison of storage efficiency

In Fig. 8, one rectangle represents one fragment which corresponded to one subspace, and red rectangle represent query range. (a) Shows that query range covers only one fragment and (b) indicates that query range covers many fragments. Our experiment will be conducted in these two situations.

When the spatial query situation is (a), the query operation will be implemented in single fragment before and after optimization. The query operation can be directed on affirmatory fragment with start and end primary key in scan operation. We record the RPC number at different data size, which is shown in Fig. 9, the size of data that RPC returns, which is shown as Fig. 10, and cost time of querying, which is shown as Fig. 11.

When spatial query situation is (b), it cannot be implemented in single fragment, as we can get the start and end primary key in the process of querying. And spatial data is stored into distributed column-oriented database based on lexicographic order of primary key. Then start and end primary key may be involved in irrelevant fragments. However, 'SPPS' can calculate the set of fragments which obtain query result according to the spatial position relationship implied by the primary key. To verify our theory, we conduct spatial query before and after optimization. We also record the query records of RPC number which are shown in Fig. 12, size of data that RPC returns which is shown as Fig. 13, and cost time of querying which is shown as Fig. 14.

After finishing the query experiments and analyzing the experiment results, we can conclude two points as follow:

- When the scope of query covers a few subspaces, the query processing will execute in the same fragment, and do not cover other data fragments. So the efficiency of query does not improve obviously, but the efficient of 'SPPS' query keep the same level with default strategy.

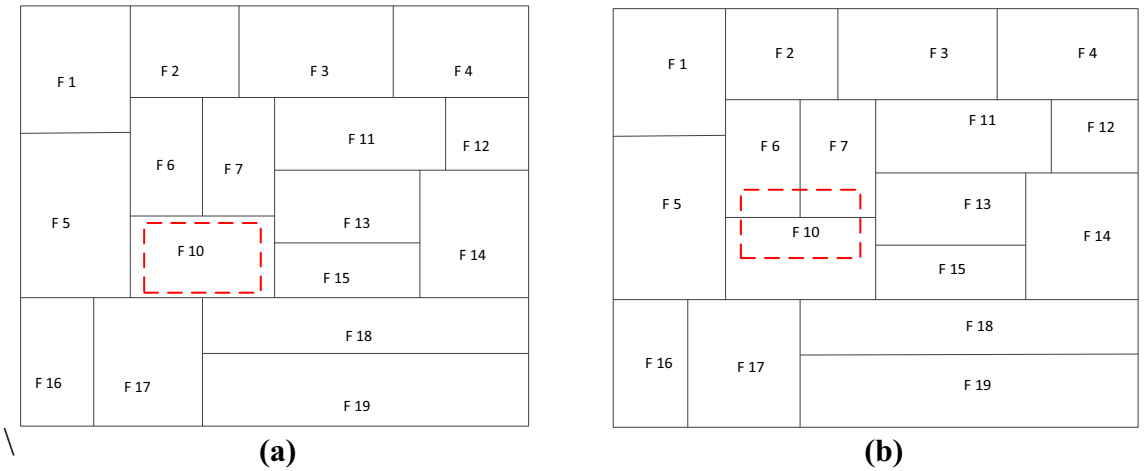


Fig. 8 Two different situations of spatial query

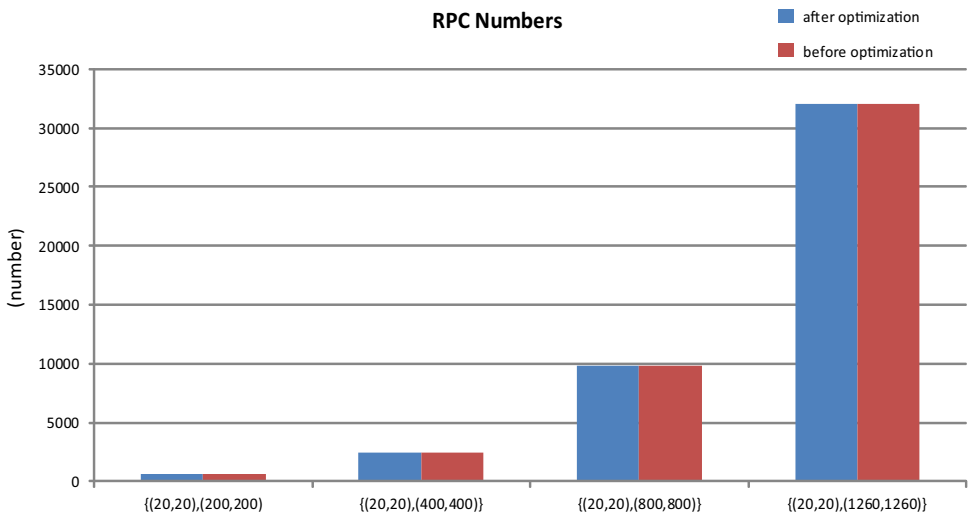


Fig. 9 Information of RPC numbers when scopes of query cover a few subspaces

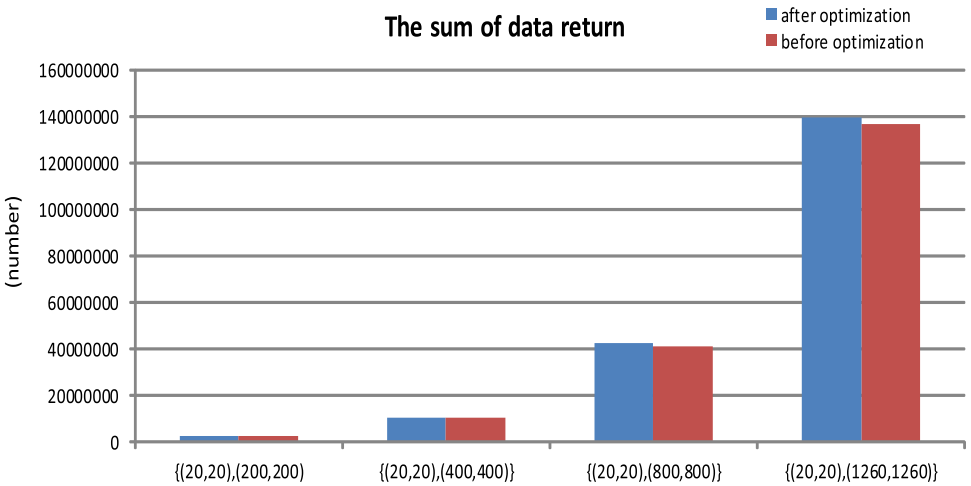


Fig. 10 Data size of RPC return when scopes of query cover a few subspaces

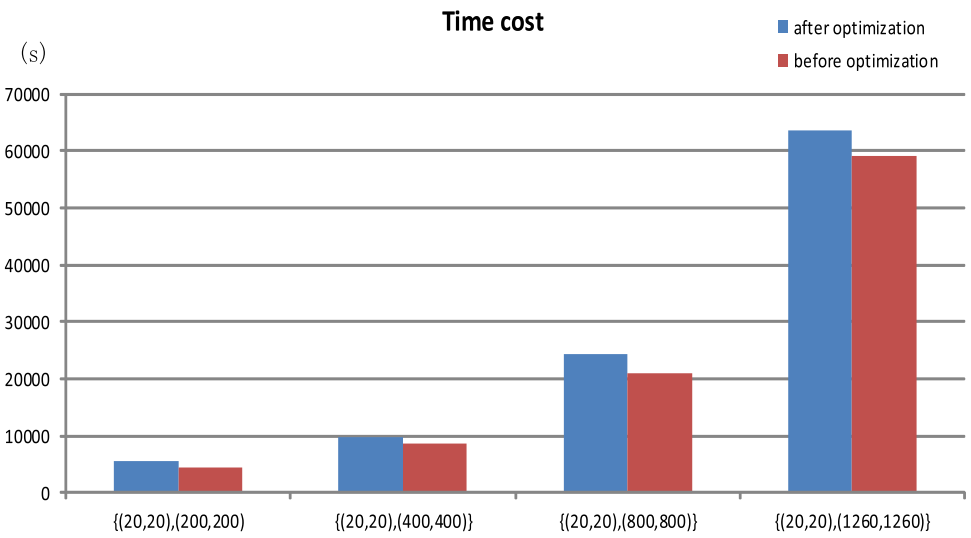


Fig. 11 Time cost when scopes of query cover a few subspaces

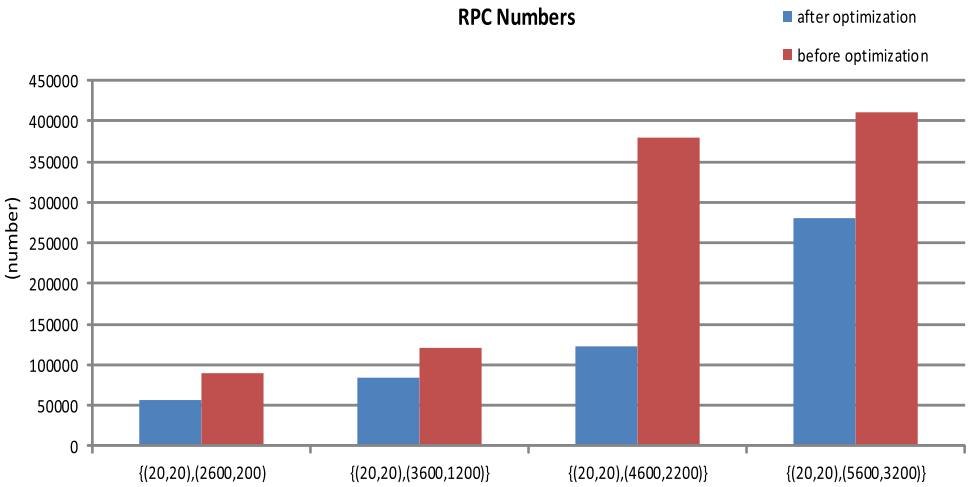


Fig. 12 Information of RPC numbers when scopes of query cover many subspaces

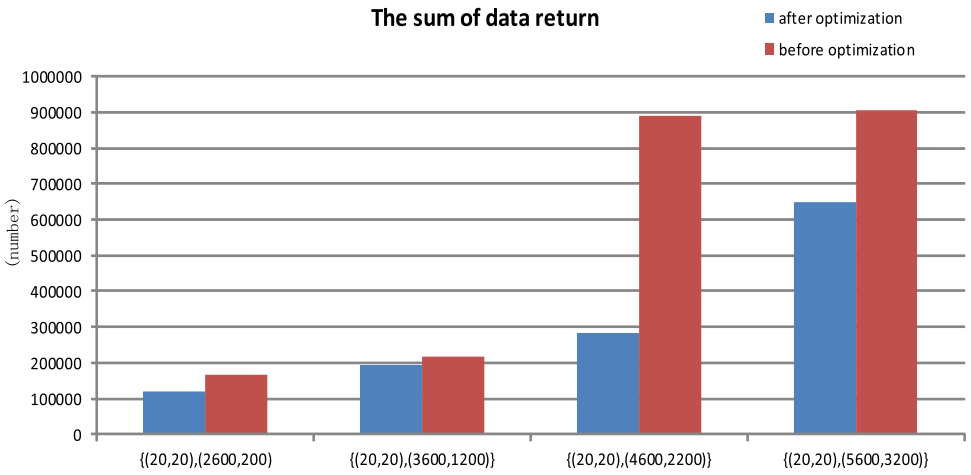


Fig. 13 Data size of RPC return when scopes of query cover many subspaces

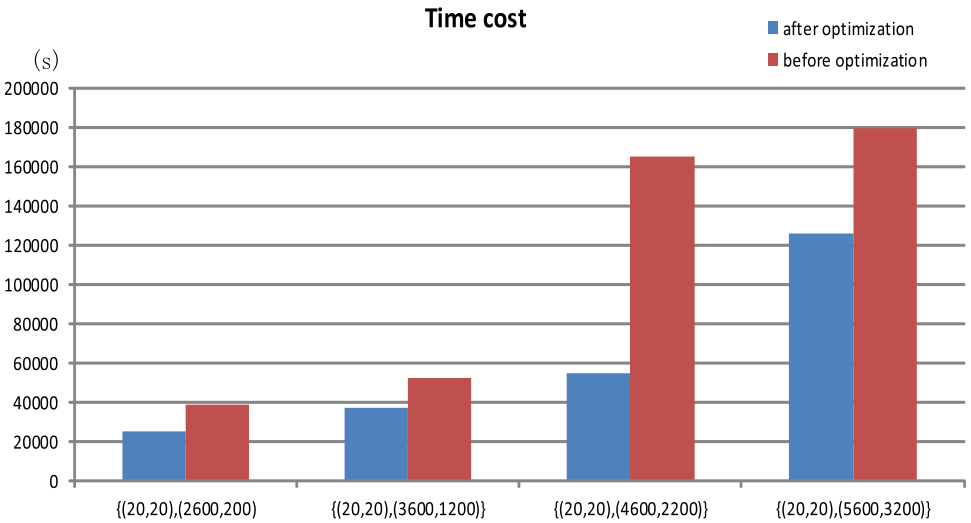


Fig. 14 Time cost when scopes of query cover many subspaces

- When scope of query covers many subspaces, the query processing will execute in many fragments. Then 'SPPS' can locate query fragment immediately, and avoid invalid fragment scan operations, so the RPC invoking times, RPC return data and query times can be decreased. It represents that the efficiency of spatial query is improved significantly.

Overall, when querying scopes cover many subspaces, the 'SPPS' optimization strategy can decrease the workload of query, and improve the efficiency of query obviously.

5 Conclusion

In this paper, we proposed 'SPPS' optimization strategy which considers spatial adjacent property. Different from other spatial data storage method, the 'SPPS' strategy can hold geospatial information of data fragments, so it can directly locate the fragments which contain spatial object. Then invalid partitioning scan operation can be decreased and the query efficiency of spatial data will be improved. nevertheless, our 'SPPS' optimization strategy can be used into other spatial object with using MBR(Minimum Bounding Rectangle), and yet we did not test the storage and query efficiency in this paper. So I will continue to study this question in future work. Beside, we plan to continue to study the question of the storage and management of spatial-temporal data in distributed column-oriented database and try to optimize the storage of spatial-temporal data.

Acknowledgements The authors would like to thank the following foundations for support: the National Key Research and Development Program of China (No. 2016YFB0502603), the National Key Research and Development Program of China (No. 2017YFB0503704), the Natural Science Foundation of Hubei Province of China (No. ZRY2015001543) and Fundamental Research Founds for National University, China University of Geosciences (Wuhan) (1610491B20).

References

1. Lu, F., Zhang, H.: Big data and generalized GIS. Geomat. Inf. Sci. Wuhan Univ. **39**(6), 645–654 (2014)
2. Zhang, X., Song, W., Liu, L.: An implementation approach to store GIS spatial data on NoSQL database. In: Hu, S., Ye, X. (eds.) International Conference on Geoinformatics (2014)
3. Le, H.V., Takasu, A.: An Efficient Distributed Index for Geospatial Databases, pp. 28–42. Springer, Heidelberg (2015)
4. Alvanaki, F., et al.: GIS navigation boosted by column stores. Proc. Vldb Endow. **8**(12), 1956–1959 (2015)
5. Zhang, N., et al. HBaseSpatial: a scalable spatial data storage based on HBase. In: IEEE International Conference on Trust, Security and Privacy in Computing and Communications (2014)
6. Nishimura, S., et al.: MD-HBase: design and implementation of an elastic data infrastructure for cloud-scale location services. Distrib. Parallel Databases **31**(2SI), 289–319 (2013)
7. Chen, Z., et al.: Hybrid Range Consistent Hash Partitioning Strategy—A New Data Partition Strategy for NoSQL Database, pp. 1161–1169. IEEE, New York (2013)
8. Qi, W., Song, J., Bao, Y.B.: Near-uniform range partition approach for increased partitioning in large database. In: IEEE International Conference on Information Management and Service (IMS) (2010)
9. Kumar, A., Yadav, J.S.: A review on partitioning techniques. Database **35**(3), 342–347342 (2014)
10. George, L.: HBase schema design—things you need to know—O'Reilly Media Free. Live Events (2017)
11. Chang, F., et al.: Bigtable: a distributed storage system for structured data, pp. 205–218. USENIX Association, Berkeley (2006)
12. IBM. https://en.wikipedia.org/wiki/Dynamo_(storage_system)
13. Cassandra. https://cassandra.apache.org/doc/latest/
14. Akdogan, A., et al.: Cost-efficient partitioning of spatial data on cloud. In: International Conference on Big Data (2015)
15. Xia, C., Wang, T.: Cached Index of HBase based on coprocessor. In: International Conference on Computer Science and Communication Engineering (CSCE 2015), pp. 123–129 (2015)
16. Vo, H., Aji, A., Wang, F.: SATO: a spatial data partitioning framework for scalable query processing. In: Proceedings of IEEE

International Conference on Computer Science & Software Engineering (2015)
17. Zhuang, H., et al.: Design of a more scalable database system. In: IEEE-ACM International Symposium on Cluster Cloud and Grid Computing, pp. 1213–1216. IEEE, New York (2015)
18. Zhong, Y., Liu, D.: The application of K-means clustering algorithm based on Hadoop. In: Proceedings of 2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA 2016), pp. 88–92 (2016)
19. George, L.: HBase The Definitive Guide. O'Reilly Media, Newton (2011)
20. Cruz, F., et al.: Workload-Aware Table Splitting for NoSQL, pp. 399–404. Aurora Construction Materials, Rockbank (2014)
21. Ye, Z., Li, S.: A request skew aware heterogeneous distributed storage system based on Cassandra. In: International Conference on Computer and Management (2011)
22. Elghamrawy, S.M.: An adaptive load-balanced partitioning module in Cassandra using rendezvous hashing. In: International Conference on Advanced Intelligent Systems and Information (2016)
23. Eldawy, A., Alarabi, L., Mokbel, M.F.: Spatial partitioning techniques in SpatialHadoop. Proc. Vldb Endow. **8**(12), 1602–1605 (2015)
24. Han, D., Stroulia, E.: HGrid: a data model for large geospatial data sets in HBase. In: IEEE Sixth International Conference on Cloud Computing (2013)
25. Fox, A., et al.: Spatio-temporal Indexing in Non-relational Distributed Databases. IEEE, New York (2013)
26. Hughes, J.N., et al.: A survey of techniques and open-source tools for processing streams of spatio-temporal events. In: Proceedings of the 7th ACM SIGSPATIAL International Workshop on GeoStreaming (IWGS), pp. 39–42 (2016)
27. Geohash. https://en.wikipedia.org/wiki/Geohash
28. Lee, K., et al.: Efficient spatial query processing for big data. In: ACM Sigspatial International Conference on Advances in Geographic Information Systems (2014)
29. Pal, S., et al.: Embedding an Extra Layer of Data Compression Scheme for Efficient Management of Big-Data, pp. 699–708. Springer, New Delhi (2015)
30. Leutenegger, S.T., Lopez, M.A., Edgington, J.: STR: a simple and efficient algorithm for R-tree packing. In: Proceedings of the International Conference on Data Engineering (Series), pp. 497–506. Computer Soc Press, Los Alamitos (1997)
31. https://en.wikipedia.org/wiki/Trie
32. HBase. http://hbase.apache.org/book.html
33. Chang, F., et al.: Bigtable: a distributed storage system for structured data. ACM Trans. Comput. Syst. **26**(2), 4 (2008)

Kun Zheng received his Ph.D. from China University of geoscience. He is currently an associate professor with the Faculty of Information Engineering, China University of Geosciences, Wuhan, China. His current research interests involve Spatio-Temporal data Visual Analytics, Storage and management of spatial big data.

Danpeng Gu received the B.E. degree from China University of Geosciences. He is currently master degree candidate with Faculty of Information Engineering, China University of Geosciences, Wuhan, China. His current research interest is Spatio-Temporal data Visual Analytics and parallel processing for massive spatial data.

Falin Fang received the B.E. degree from China University of Geosciences. He is currently master degree candidate with Faculty of Information Engineering, China University of Geosciences, Wuhan, China. His current research interest is the storage and management of spatial data and parallel processing for massive spatial data.

Miao Zhang received the B.E degree from North China University of Water Resources and Electric Power. She is studying for a master's degree in Information Engineering at China University of Geosciences. Her research interests are spatial index, spatial relation extraction.

Kang Zheng received the B.E. degree from China University of Geosciences. He is currently master degree candidate in major of Geographic Information System, in China University of Geosciences, Wuhan. His current research interest is Geographic Information System in cloud computing environment and parallel processing for massive spatial data.

Qi Li received her Bachelor degree in China University of Geoscience. She is currently a graduate student majoring in GIS at Faculty of Information Engineering, China University of Geosciences, Wuhan, China. Her current research interest is spatio-temporal data 3D visualization.