OPEN ACCESS

International Journal Of Modern Engineering Research (IJMER)

Range Query on Big Data Based on Map Reduce

Zhang Tingting[1], Qu Haipeng[2]
[1](College of Information Science and Engineering, Ocean University of China, China)
[2](Corresponding Author, College of Information Science and Engineering, Ocean University of China, China)

Abstract: *Range query in P2P system is mainly made by establishing index, such as B+ tree or DST. However when the number of nodes in the system and the amount of data in a single node increases significantly, the above traditional index will become extremely large so it will affect the query efficiency. In the present, enterprises require effective data analysis when making some important decisions, such as user's consumption habits deriving from analyzing user data. This paper aims at optimization of range query in big data. This algorithm introduces MapReduce in P2P system and organizes files by P-Ring in different nodes. When making range query we use P-Ring to find the corresponding files and then search data in the file by B+ tree.*

Keywords: *Range query, Big Data, Map Reduce, P-Ring, B+ tree*

I. Introduction

Today, the amount of data in various fields is increasing. Traditional data processing is transforming to big data. Big data which is firstly put forward by Viktor Mayer-Schönberger 0 is a data set whose content can't be crawled, managed and processed through conventional software tools within a certain time. It has four characteristics which are Volume, Variety, Velocity and Value 0. Google deals with data through large-scale cluster and MapReduce software and the amount of data processed every month is over 400PB 0. Cloud computing platform of Yahoo! has 34 clusters and more than 30000 computers. Its total storage capacity is over 100PB 0. The traditional query algorithm is unable to meet the needs of enterprises. How to quickly find qualified data in such a flood of information becomes a major problem of enterprises.

Traditional range query is mainly based on establishing ordered indexes. Reference 0 presented a range query algorithm in P2P system. Firstly it creates a B+ tree index for each numerical attributes of the file and the index information of the file is only stored in the leaf nodes. When making range the original range is divided into sub-ranges according to the data stored in the root node of the B + tree. Each sub-query is made along the corresponding child node until the leaf node is reached. Then the queried results are returned. However, the performance of this algorithm extended to big data environment is to be improved. Reference 0 proposed a global index structure based on Map Reduce to optimize range query. Firstly sort the indexed attributes. Then the original index table is divided into a number of sub-tables sequentially and these sub-tables are assigned to different nodes of the cluster. Each sub-table of global index stores the location list of files. When making range query the range condition is split to different nodes and each sub-condition is parallel executed. But performance of this algorithm is limited to the speed of the global index.

The main difficulty of data queries on big data environment lies in how to extend the traditional query algorithm to massive data environment while query efficiency is not affected. Establishing a global index will lead to index being algorithm bottleneck of large data environment. Therefore, we need to get rid of the limitations of traditional centralized structure algorithm.

Map Reduce precisely is a model for large-scale data sets. It includes two functions Map and Reduce. Users only need to define these two functions without considering how to achieve the distribution of the underlying system. So this model facilitates programmer to run program in distributed system under the circumstance of not familiar with distributed programming. This model is frequently used to process large-scale distributed systems 0.

This paper presents a range query algorithm processing big data based on Map Reduce and P-Ring **Error! Reference source not found.**. Files on each node are organized by P-Ring and then nodes of distributed system are organized by Map Reduce. Range query is divided into two steps: search files and query data.

The rest of this paper is organized as follows, section II surveys related work. Section III introduces dual-index. Section IV presents basic idea of the algorithm. Section V concludes this paper.

II. Related Work

Range query is achieved mainly based on building ordered indexes. But the index will become very large in big data. So what we need to do is to try to simplify the index structure. At present, the main index structure for range queries is B+ tree. It can sort files according to one attribute so that it is convenient for range query.

In order to simplify the index structure, this paper combines P-Ring and B+ tree to build a dual-index structure. This structure avoids the shortcoming of index being too large caused by establishing B+ tree directly among documents. This paper establishes a P-Ring for files in different nodes of distributed system. The algorithm complexity of exact search by P-Ring is O (logN) **Error! Reference source not found.**. N is the number of nodes in P-Ring 0. Then create a B+ tree for a file based on attribute in the query condition and query data inside a file. This method effectively reduces the height of the B + tree and takes full advantage of the characteristics of P-Ring and B + tree.

At present P-Ring is mainly used in large-scale distributed P2P systems to search data. At present index structures of range query about P2P have some defects in a certain extent. P-trees can only handle a single data item per peer, and hence, are not well-suited for large data sets. The index proposed by Gupta et al. 0 only provides approximate answers to range queries and can miss results. Mercury [4] and P-Grid 00 provide probabilistic (as opposed to absolute) guarantees on search and load-balancing, even when the P2P system is fully consistent. Reference 0 pointed out that the main advantage of P-Ring relative to existing index structures of range query is high accuracy and it can be extended to environment where there are a large number of nodes and data.

III. Dual-Index

The index structure proposed in this paper firstly indexes to the file by improved P-Ring, and then to specific data by B+ tree.

A. Defect of current P-Ring

Traditional P-Ring mainly index to data. Under the environment of big data, data is distributed different nodes by the unit of file. So if we want to execute range query, the file including corresponding data must be found. In this circumstance traditional P-Ring can't meet our needs.

B. Improved P-Ring Index

P-Ring is an index based on sort. Data on P-Ring is strictly sequential. Each node stores a value and the location between two nodes stores values ranging from previous node to next one. In P-Ring every node saves hierarchically information about its successor node and the value of hierarchy is differentials of value between successor node and current node. Improved P-Ring is improved to index by files through creating index for ranges of attribute values in a file. Value segment between two nodes can store the index information for one or more files.

The index structure proposed in this article organizes files between different nodes by improved P-Ring. Every node stores one value according to ranges of attributes in file. Different nodes are arranged sequentially and everyone stores a table of index information named infoTable. The infoTable stores range segments, file name and file address of corresponding files of these segments.

The following are examples of file data:

Table 1 File information of node 1

File name	Range of attribute value	File address	Node of P-Ring
F1	10-15	A1	P0
F2	5-10	A2	P1
F3	6-8	A3	P1
F4	10-18	A4	P0
F5	1-8	A5	P1,P2

Table 2 File information of node 2

File name	Range of attribute value	File address	Node of P-Ring
F6	2-10	A6	P1,P2
F7	13-15	A7	P0
F8	4-6	A8	P1,P2
F9	2-10	A9	P1,P2
F10	16-20	A10	P3

Process of establishing P-Ring index on the above documents is as follows.

1) Store F1. Firstly define two nodes P0 and P1 on the ring, respectively stores 10 and 15. Then save the address and attribute ranges of F1 in infoTable of P0. Assuming the name of range is R, R0=(10,15);
2) Store F2. Because attribute range of F2 is 5 ~ 10, it is not included in the scope of R0, so create a new node P storing value 5. Because 5 is smaller than 10 stored in P0, so the P0 and P1, respectively, plus 1 to P1 and P2. The newly inserted node is P0. Save the address and attribute ranges R2=(5,10) of F2 in infoTable of P0;
3) Store F3. The attribute range of F3 is 6~8, so its name and address is stored in P0 and divide the range of P0 which is R2 into r21=(5,6), r22=(6,8) and r23=(8,10). Then corresponding files of every sub range are stored its infoTable. In this example the corresponding file of r21, r22 and r23 is respectively F2, F2 and F3, F2;
4) Store F4. The procedure is similar to 3). When storing file address we need to save the corresponding machine number or address of file;
5) Follow this step to store the remaining files.

The P-Ring created in accordance with above steps using files in Table1 and Table2 is as shown below.

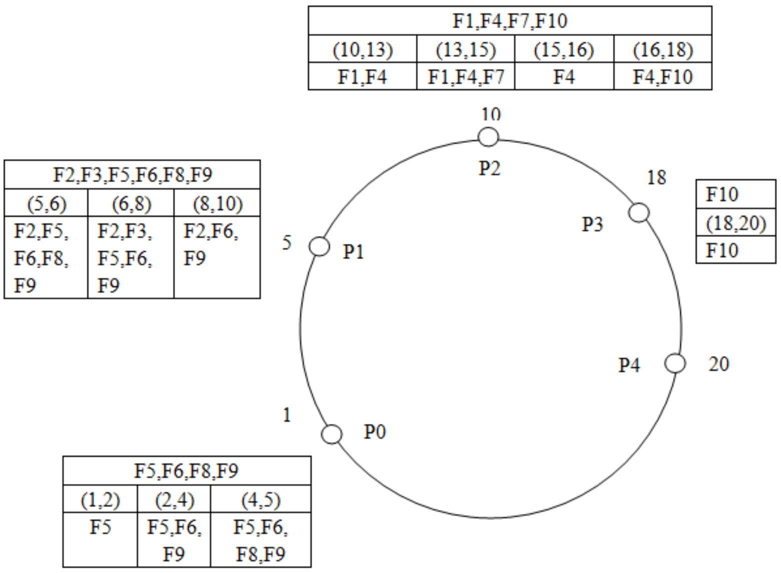

Figure 1 P-Ring index based on files

The first line of info Table in the above Fig.1 stores file name and file address. The second line stores every sub ranges. Corresponding file name of every range is stored in third line. Node of this ring can be Map of Map Reduce.

The central issue of P2P is routing. So after index structure is completed, we need to build the routing table for nodes. Traditional P-Ring structure uses hierarchical routing algorithm and the value difference of successor node and the current node in level n is 2^{n-1}. This article draws lessons from hierarchical algorithm and improves it on the basis of current one. Specific algorithm is as follows:

Routing of every node includes its predecessor and successor node. The definition of level is same to traditional P-Ring. As the number of levels in traditional P-Ring is $\lceil \log_2 N \rceil$ and in this article predecessor nodes are also indexed as similar with successor nodes so the number of levels is smaller than traditional P-Ring. It is $\lceil \log_2 \frac{N}{2} \rceil$. N is the number of nodes.

Routing table of nodes in Fig.1 according to above algorithm is as Fig.2.

Because the query complexity of current P-Ring is $\log N$. In the modified routing algorithm predecessor and successor nodes respectively index half of all nodes in the ring. So as long as traversing $\lceil \log_2 \frac{N}{2} \rceil$ nodes we can find information we need. Therefore query complexity of this algorithm is $\log \frac{N}{2}$.

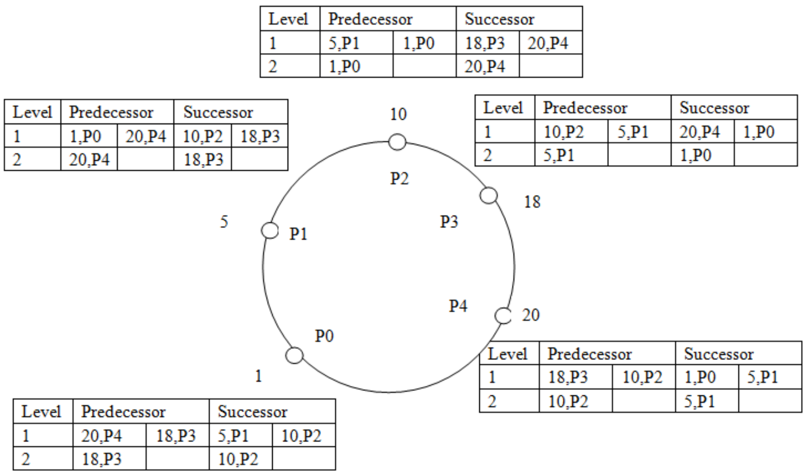
Level | Predecessor | Successor
1 | 5,P1 | 1,P0 | 18,P3 | 20,P4
2 | 1,P0 | | 20,P4 |
Level | Predecessor | Successor
1 | 1,P0 | 20,P4 | 10,P2 | 18,P3
2 | 20,P4 | | 18,P3 |
Level | Predecessor | Successor
1 | 10,P2 | 5,P1 | 20,P4 | 1,P0
2 | 5,P1 | | 1,P0 |
10
P2
18
P3
5
P1
P4
20
P0
1
Level | Predecessor | Successor
1 | 18,P3 | 10,P2 | 1,P0 | 5,P1
2 | 10,P2 | | 5,P1 |
Level | Predecessor | Successor
1 | 20,P4 | 18,P3 | 5,P1 | 10,P2
2 | 18,P3 | | 10,P2 |

Figure 2 Routing table of nodes

When a range query is coming firstly we need to determine whether the scope of query contains sub ranges. If a range is split into several sub-ranges we can firstly record corresponding file name of each sub-range, and then merge them.

For example supposing that P4 receives a query whose attribute range is 1~8. Firstly find the minimum value 1 in the infoTable of P4. It is located in P0. Because stored values of range in every node of P-Ring are no smaller than the value of this node. As 8 is smaller than 10 which is the value of P2 so the maximum value 8 of the query range is located in P1. Then the search goes on in P2 and P1. The locating based on nodes in this example needs only once.

When completing locating nodes we need concretely search based on values in located nodes. In the above example we firstly query P0. Since the range stored in P0 is 1~5 and our query range is 1~8 which includes all values in P0. So all files stored in P0 need be queried in detail which are F5, F6, F8 and F9. Then query P1 to find files meeting with 5~8. There are two sub ranges in P1 meeting conditions which are (5, 6) and (6, 8). So record corresponding files of these two ranges and they are F2, F3, F5, F6, F8 and F9. Summarizing files meeting with query condition of the above two ranges (The repeated files in two ranges don't need to search twice.). The final results are that F2, F3, F5, F6, F8 and F9 need to be searched in detail. According to index information in infoTable find corresponding files to query concretely based on attribute values. That is B+ tree index.

C. B+ tree index

After finding corresponding files, we need to seek data meeting the condition in each file. Such search based on data requires building a B+ tree. That is to say the logical organization structure among data is B + tree. This paper takes F4 as an example. Assuming that data stored in F4 shows as the following table.

Table 3 Data in F4

Order Number	Attribute	Order Number	Attribute
1	43	11	54
2	46	12	58
3	51	13	43
4	44	14	46
5	42	15	52
6	50	16	56
7	55	17	59
8	48	18	57
9	49	19	47
10	41	20	60

B+ trees established based on attribute of this file show as follow.

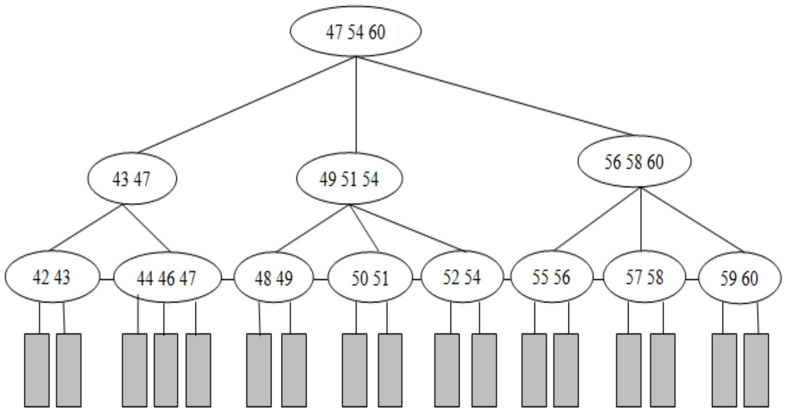

Figure 3 B+ tree based on attribute of file 4

In B+ tree only leaf nodes store keywords (In this paper they refer to data in the table). Non-leaf nodes store index only containing the biggest keyword of its sub-tree.

In the dual-index proposed in this paper, files found in Chord consist of information about B+ tree pointed to these files. For example, when the query range is 47<a<51 F4 is the found file in Chord and then next step is making specific search in B+ tree of F4. Query results are 48、49、50、51.

B + tree index is created before the first query and the following queries can use it directly unless data is update.

D. Map Reduce

The Master node needs to know about information stored on each node so as to avoid sending query to a node which will return null results. That will affect query efficiency. Therefore each node needs to send its own ID, the range of fileID in this node and numeric attributes (Before receiving the query which attribute will be searched is not determined.) in each file to Master node which stores these index information. When deciding Map node we can choose nodes which don't execute query.

When Master node receives query request it firstly checked index information stored in it. Then Master node sends this request to a node containing this queried attribute. The chord will retrieve files containing the query attribute and attributes whose values range meets the query condition.

IV. Range Query Processing

When one query comes the Master node of Map Reduce decides which Map node the query will be sent to. After Determining queried node the search based on file begins in each node to find qualified files. Then search data inside the file. Finally data found is sent to the Reduce node to merge results.

The concrete steps show as following:

1. The query is sent firstly to the Master node. As the Master node stores attributes value range of files of the other nodes. It maintains a P-Ring index including all files in other nodes. So the Master node can determine which one or more nodes the query will be executed in. In this article we use P-Ring to organize nodes of MapReduce. Every node under query is Map.
2. When Master receives query it firstly needs to determine which node the maximum and minimum value of range is located in. Files in the node which storing values meeting with query range. There are following circumstances.

1) The maximum and minimum value is exactly the value of node. If this happens, infoTable of all nodes (including node which minimum value is located in) between these two nodes need to be searched concretely;

2) The minimum value is exactly equal to one node assuming it is P1 and the maximum value is between two nodes assuming P0 and P3. We need to determine which sub range of infoTable of P0 the maximum value is located in. So all files in P1 and files which are located in range from the first sub range of P0 to sub range includes the maximum need to be query.

3) The maximum value is exactly equal to one node or two values of query range are between two nodes in ring. Locate files according to step 2) and return file name and address needing to search concretely. If two values of query range are between two nodes we can search files in two Map nodes at the same time using the parallelism of MapReduce.

1. Query B+ tree of each file in parallel in the list of found files according to file name and file address in infoTable. The search begins with root node of B+ tree. If child nodes fit with query condition the search will continue from this branch until the query arrives at leaf nodes. Finally results are returned to Reduce nodes designated by Master.
2. On the end of each Map node query, it sends a message to the Master node. Master decides which Reduce node to merge these results will send to.

V. Conclusion

This paper proposed a range query algorithm mainly based on big data which applies dual- index structure. In order to simplify B + tree we combined P-Ring with it. Files are stored in different nodes. The range between two nodes is corresponding to files containing these values. MapReduce dispatches tasks among nodes. The time complexity of this algorithm is finding files in P-Ring adding up to finding data in B+ tree. That is $\log\frac{N}{2}$+logP .N is the number of nodes in P-Ring and P is the maximum number of items in one file.

REFERENCES

[1] Viktor Mayer-Schönberger, Big Data: A Revolution That Will Transform How We Live, Work, and Think.

[2] Wang Dan, Li Maozeng, A Range Query Model based on DHT in P2P System, 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing.

[3] Hui Zhao, Shuqiang Yang, Zhikun Chen, Songcang Jin, Hong Yin, Long Li, MapReduce model-based optimization of range queries, 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012).

[4] Adina Crainiceanu, Prakash Linga, Ashwin Machanavajjhala, P-Ring: An Efficient and Robust P2P Range Index Structure, SIGMOD '07 Proceedings of the 2007 ACM SIGMOD international conference on Management of data Pages 223-234 New York 2007 .

[5] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. SIGCOMM Comput. Commun. Rev., 34(4), 2004.

[6] Michael Cardosa, Chenyu Wang, Anshuman Nangia, Abhishek Chandra, Jon Weissman, Exploring MapReduce efficiency with highly-distributed data， MapReduce '11 Proceedings of the second international workshop on Map Reduce and its applications, Pages 27-34 , New York 2011.

[7] Samet Ayhan, Johnathan Pesce, Paul Comitz, Gary Gerberick, Steve Bliesner, Predictive analytics with surveillance big data, BigSpatioal’12 Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, Pages 81-90, New York 2012.

[8] Edward Z. Yang, Robert J. Simmons, Profile Jeff Dean: Big data at Google, XRDS: Crossroads, The ACM Magazine for Students - Big Data, Volume 19 Issue 1, fall 2012, Pages 69-69

[9] Jens Dittrich, Jorge-Arnulfo Quiane-Rioz, Efficient big data processing in Hadoop MapReduce, Proceedings of the VLDB Endowment, Volume 5 Issue 12, August 2012 , Pages 2014-2015.

[10] A. Gupta, D. Agrawal, and A. El Abbadi. Approximate range selection queries in peer-to-peer systems. In CIDR, 2003.

[11] A. Datta, M. Hauswirth, R. John, R. Schmidt, and K. Aberer. Range-queries in trie-structured overlays. In P2P Comp. 2005.

[12] P. Ganesan, M. Bawa, and H. Garcia-Molina. Online balancing of range-partitioned data with applications to peer-to-peer systems. In VLDB, 2004.