A Massive Structured Data Storage Technology for Commodity Screening Applications

Feng XU[1,a] and Wei ZHANG[1]

[1]*School of Computer Science, Beijing Information Science &Technology University, Beijing 100101, China*

Abstract. With the rapid development of e-commerce, the number of goods has become more and more. When commodity screening system is used to store and process mass information, the existing models require all nodes in the distributed system to work in parallel, then the results of each node are integrated to get the final results, the process produces a lot of invalid queries. In order to solve this problem, proposed a new distributed structured data storage method. It statistics the history search results and chooses the high frequency or core columns to be key columns. The data can be stored based key columns and distribute system architecture. Then in the searching stage, only some nodes work when the search refer to key columns. The results show that this method can reduce the tasks and improve the throughout without extra storage consumption.

1 Introduction

With the development of cloud computing and Internet of things, more and more people choose online shopping. According to the thirty-seventh CNNIC China Internet Statistics report shows that as of December 2015, a total of 413 million Internet users online shopping. With the increasing of commodity information, the electronic commerce system needs to store large amount of commodity information, the rapid growth of access requirements and product data has brought great pressure on data query and storage.

At present, the electronic commerce system usually uses the massive data processing model based on Hadoop[1-4] to process the data. In Hadoop processing model, HDFS[5-6] is used to store data and MapReduce[7-9] for distributed computing. However, with the wide application of Hadoop in e-commerce system, the common problems are becoming more and more serious. For example, it is easy to produce invalid query and so on, which reduces the overall throughput of the system and increases the system load. So how to store data and reduce the invalid query is a very worthy of inquiry.

To solve the above problems, this paper presents a distributed structured data storage technology for commodity screening applications. This technique makes the statistical analysis of the history table query, use the field of high frequency or core as a key column, in accordance with the key column data partition. For the query with key columns, the query task can be generated only for the region containing the result information, which can greatly reduce the query data and the number of working nodes, thus effectively improve the throughput of the system.

[a] Corresponding author: fenng.xu@foxmail.com

2 Related research

In the research of Chinese scholars, literature [10] proposed a two-stage data placement strategy based on relevancy, The strategy will be closely related data as much as possible into the same data center, Relatively loose data will be placed in different data center, Ensuring high cohesion between data sets in the same data center and low coupling between data sets in different data center, the task scheduling policy schedules tasks to the largest data dependence of the data center. literature [11] presents a data placement strategy for data intensive applications in a cloud computing environment, This strategy models based on the data dependencies between datasets, The data dependencies between datasets are expressed as clustering matrix, then the clustering matrix is used to classify the data sets, and allocates a storage location for each partition. An improved Hadoop data placement strategy is proposed in [12], the strategy considers both node distance and load, improve the overall performance of the system and balance the node load. The clustering index of the database is described in [13]. The clustering index, the difference between the clustered index and the non-clustered index, and how to select the clustered index are described.

In the research of foreign scholars, literature [14] proposed CoHadoop, modified the underlying HDFS data placement strategy, allowing applications to control the data storage location, as far as possible the relevant data placed in the same group of nodes. In [15], a dynamic data placement strategy is proposed based on the characteristics of actual data, such as the interdependence of data, the actual application and the change of user location. In [16], the processing capacity of each node is taken into account during the data placement, and then the data blocks are allocated according to the node processing capacity, the load is balanced and the throughput of the system is improved.

Based on the above analysis, the domestic and foreign scholars have optimized the default data placement strategy from the aspects of node performance, system performance, data characteristics and data dependency. In this paper, from the data dependency point of view, a data storage strategy based on key column pre-processing is proposed, which can significantly reduce the number of invalid queries and reduce the overall system query times and improve the system throughput.

3 Based on pre-processing key columns distributed data storage technology

In this section, the basic idea and the specific design of the distributed data storage technology based on the key column pre-processing are proposed, reduces the invalid query and improves the throughput of the system effectively.

3.1 The basic idea of distributed data storage technology based on key column pre-processing

In order to solve the problem that the existing e-commerce website is easy to generate invalid query, this paper proposes a distributed structured data storage technology. The technology statistics table query through history, the data in the table in accordance with the query frequency in descending order, then select the core or higher frequency field as the key column. In Figure 1, in the data storage phase, the data is partitioned into multiple storage areas according to the data dependency of the key columns. In the query processing stage, when dealing with the query with key columns, through the key column pre-processing, can only query for some storage area, reduce the invalid query. The experiment results show that this technique can reduce the total number of tasks in each node of the distributed system without the extra storage overhead, and improve the system throughput.

In the data storage phase, the technology divides all data nodes into multiple storage areas according to the specific application requirements. As shown in Figure 1, the original data is preprocessed and the data with the relevant attributes is stored in the same area. For example, in Figure 1, after the historical query data statistics, the key column number N is 3, the number of storage area M is 3, assuming that the original data table has nine records, through the key column pre-processing, these nine records divided into three storage areas. Area 1 is an area having the same shape, Area 2 is an area having the same gray, and Area M is an area having the same shading.

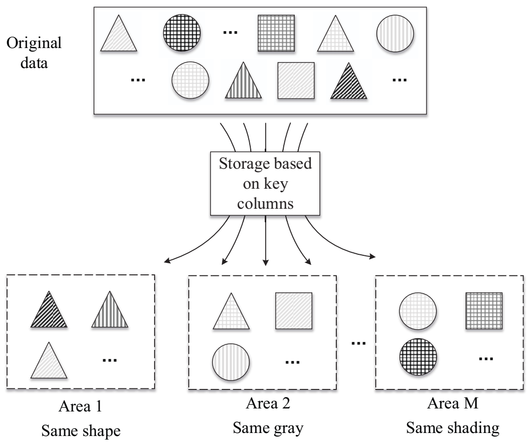

Figure 1. Storage model based on key column pre-processing

In the query processing phase, the query request can generate query tasks only for the region containing the result information by key column pre-processing. When an existing model processes a query request, it traverses all the storage nodes, but some nodes do not contain the result information, resulting in many invalid queries. In the key column pre-processing model, when the query containing the key columns is processed, After the pre-processing phase, the query request condition can be judged, which can only generate valid query task for some nodes and reduce invalid query. For example, in Figure 1, Area2 stores gray-level data. When performing gray-level data query, the existing model will query all the nodes, while the query request can only query Area2 after pre-processing based on key columns.

After the historical query data statistics, this paper assumes that the number of key columns is N, the number of storage areas is M, the data after the key column pretreatment, and then processing the query request, through theoretical analysis and experimental verification of the following rules: When a query request, the query conditions as long as any storage area to meet more than N/2 (down rounding) key columns, you only need to query the storage area, Otherwise, query all storage areas.

Therefore, when there are a large number of query requests, this technology can reduce the number of tasks in the distributed system, reduce the system load and improve the throughput of the system.

3.2 The design of distributed data storage based on key column pre-processing

This section mainly describes the design of the distributed storage technology based on the key column pre-processing.

3.2.1 The design of data storage based on Key Column pre-processing

In the existing storage model, the request for the query requires all nodes in the system all work in parallel, the process will produce a lot of invalid queries. To solve the above problem, this section divides the data into the corresponding storage area according to the data attributes of the key columns, and then the subsequent query request for pretreatment. The results show that the method can effectively reduce the number of nodes query and improve the system throughput.

In this paper, we assume that the number of key columns is N and the data nodes are divided into M storage areas according to the statistics of historical query records, where N and M are not less than

1. Through this design, the original data in the data table through the key column pre-processing, all the original data according to the value of N key columns, re-partitioned into M storage area, any record in the data table will be assigned to a specific region. The newly added data is also partitioned according to the key column data partitioning strategy. In Figure 2, the raw data is divided into each region by the key column pre-processing. There are N possibilities for data in each region. Through the above process the original data through the key column data partitioning strategy to re-partition the regional storage.

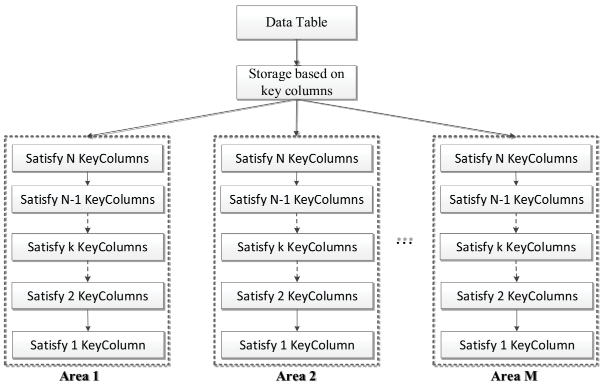

Figure 2. The design of data storage based on Key Column pre-processing

The following is a detailed description of the data storage method based on the key column pre-processing.

First, get the number of key columns N and the number of storage area M. Key column collection based on history query of data table, and sorted according to the key column heat, using TopK algorithm to solve the problem of selecting key columns, thereby determining the key column and its number N and the storage area and its number M, where M, N are not less than 1. For example, the statistical analysis of the notebook product information table, using TopK algorithm to obtain the table column Top3: brand, price and size. Then the number of key columns N is 3, the brand is the first key column, the price for the second key column, the size of the third key column. Assuming the number of data nodes is greater than 3, and after research found that commodity prices need to be divided into three files to sell, then the number of storage area M 3.

Then, the value of each key column is divided into the storage area. According to different data types, there are two kinds of specific partitioning strategies: data partitioning strategy for continuous fields and data partitioning strategy for discrete fields.

The data partitioning strategy for continuous fields is mainly applicable to the key column for continuous values, and has a certain range of values. When the data of the key columns are continuous values, the strategy divides the values of the key columns into different ranges and assigns them to M storage areas. For example, the price in the data table is the key column, and the value of the price is continuous from 0 to 999, the number of storage area M is 2, then the price in the range of 1 is [0-499], the price in the range of 2 is [500-999].

The data partitioning strategy of the discrete field is mainly applied to the discrete value of the key column, and the value does not have certain regularity. When the data of the key column is not continuous or does not have a certain range of values, the strategy according to the hash algorithm to assign the value of the key column to the storage area. For example, the size of the commodity information table is the key column, and the size of the value of 6, 9, 10, 11, 14 and 16, the number of storage area M is 3, then according to the hash algorithm, the size of the range of values in the region 1 Is [6, 9], the size in the range 2 is [10, 16], the size in the range 3 is [11, 14].

Finally, the data in the data table is divided into various storage areas. For any record in the data table, after all the key column points are judged, the record is divided into the area with the highest drop point, as shown in Figure 3, the specific strategy is as follows:

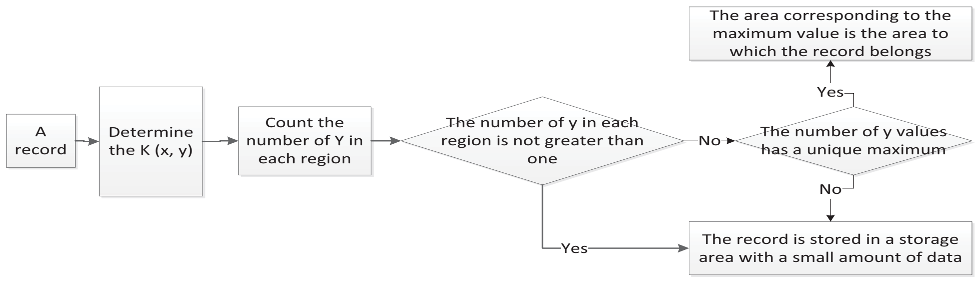
A record
Determine the K (x, y)
Count the number of Y in each region
The number of y in each region is not greater than one
No
Yes
The number of y values has a unique maximum
Yes
No
The area corresponding to the maximum value is the area to which the record belongs
The record is stored in a storage area with a small amount of data

Figure 3. The data placement strategy based on key columns

1. Determine the K (x, y) for all the key columns in the record. K (x, y) indicates that the value of the x-th key column in a data record belongs to the y-th region, For example, K (1, 2) of a record, then the value of the first key column of the record belongs to the second region.
2. Statistics the number of y in each area, that is, the number of key columns in the same area.
3. If the number of y in each region is not greater than 1, it means that each key column in the record belongs to a different region. Then, according to the storage load balancing, the record is stored in a storage area with a small amount of data
4. If a region or a number of regions within the y is greater than 1, that there are a number of key columns belong to the same region. If the number of y within a region has a unique maximum value, which means that the key column in the region is the largest, then the record will be divided into the region; If the number of y in each region does not have a unique maximum value, which means that the number of y in the plurality of regions is the same and maximized, the record is stored in the storage region with a small amount of data according to the storage load balancing. For example, the number of key columns N is 4, assuming that all the key columns K (x, y) of a record are K (1, 3), K (2, 3), K (3, 4),K(4, 4), That is, the first and second key columns belong to the region 3, the third and fourth key columns belong to the region 4, the region 3 and the region 4 have two key columns, the number of all regions y does not have a unique maximum value, And the number of the key columns of the regions 3 and 4 is the largest, the record is stored in a storage area with a small amount of data according to the storage load balancing.

3.2.2 The design of data query based on key column pre-processing

This section describes the data query strategy based on key column pre-processing. In this strategy, all the query requests through the key column pre-processing, can only contain the results of the storage area to generate query tasks, the specific strategy shown in Figure 4.

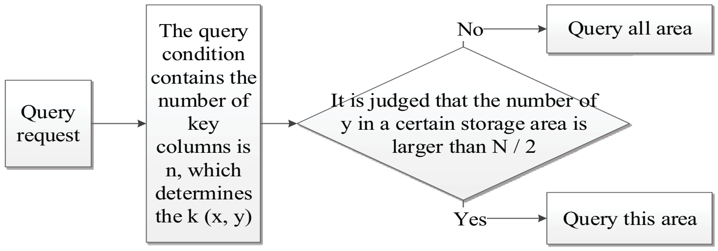
Query request
The query condition contains the number of key columns is n, which determines the k (x, y)
It is judged that the number of y in a certain storage area is larger than N / 2
No
Query all area
Yes
Query this area

Figure 4. The data query strategy based on key column

Assuming that the total number of key columns is N and the total number of storage areas is M, the query condition of a query task contains n key columns, and n is not greater than N. Specific strategies are as follows:

1. Determine K (x, y) for all the key columns in the query condition;
2. Statistics of the number of y in each storage area, that is, Statistics the number of key columns in the same area;
3. It is judged in each storage area that if the number of y in a storage area is larger than N/2 (rounding down), the storage area is directly inquired, otherwise, all the storage areas are queried.

4 Case verification

In order to verify the performance of the distributed data storage technology based on the key column pre-processing, the verification experiment is carried out on Hadoop and Hive, and compared with the existing system. The experimental platform is CDH version Hadoop and Hive, the data transmission tool is Sqoop. There are 4 nodes in the cluster, including 1 NameNode and 3 DataNodes, so the number of memory area is M is 3 and the number of key columns N is 3 in this experiment. Memory area 0 corresponds to DataNode0, memory area 1 corresponds to DataNode1, and storage area 2 corresponds to DataNode2, as shown in Figure 5.

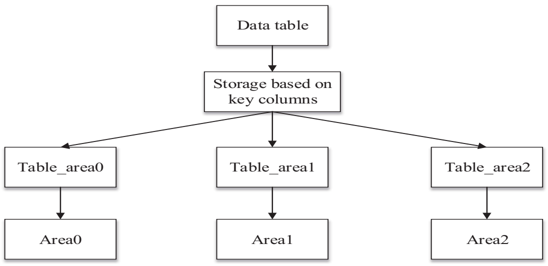

Figure 5. The experiment process

Experiment the first step to get the key columns. The key columns are collected for the historical query log of the data table. The MapReduce program is programmed to calculate the number of occurrences of each column in the query condition. The TopK algorithm is used to get the most K columns, which are the key columns. Because K = N = 3, so find Top 3, the highest occurrence of the three columns, these three columns is the key column.

The second step of the experiment divides the value of three key columns into three storage areas, and obtains the range of the key columns in each storage area. After inspection and analysis, the test data are discrete integer data, through the key column in section 3.2.1 value division method, the value of the three key columns are divided into three storage areas.

In the third step of the experiment, the data in the original data table is divided into the sub tables corresponding to each storage area. According to the data division strategy in 3.2.1 section, the original data table is divided into 3 sub tables: table_area0, table_area1, table_area2. The 3 sub tables correspond to 3 storage areas: area0, area1, area2.

In the fourth step, the 3 sub data tables are transmitted to the corresponding storage area in HDFS. In the Hadoop cluster, through the configuration of the hdfs-site.xml file, the configuration file to add the dfs.hosts.exclude blacklist of DataNode, dynamic control delete/add DataNode. When the sub table Area0 is transferred to area0, the Host1 and Host2 are added to the blacklist, and then sqoop is used to import the data sheet table_area0 into area0, which in turn introduces the other sub tables into the corresponding storage area.

Step 5 of the experiment data query test. In turn, 1000 random data queries, each query request through the pre-processing module processing, data query strategy is as follows pseudo-code:

```
for each area in areas{
  flag←false;
  n←k(x, y);
  if n > N/2 then
        flag←true;
        query(area);
  else
        continue;
  end if
  if flag != true then
        query(areas);
  end if
  }
```

As shown in Figure 6, the abscissa indicates the type of query and the ordinate indicates the time required for the query. And then 1000 queries are carried out, and the key column query is compared with the full table query. When 1000 queries satisfy 3 key columns, the query time is shortened by 13.86% compared with the full table query time. When 1000 queries satisfy 2 key columns, the time is shortened by 14.02%. When the query satisfies 1 key column Query conditions, the query time is not much change. The experimental results are in agreement with the conclusion. The three key columns and the two key columns are larger than N/2 (rounded down), so only one storage area is queried; one key column does not satisfy the condition, so query all areas.

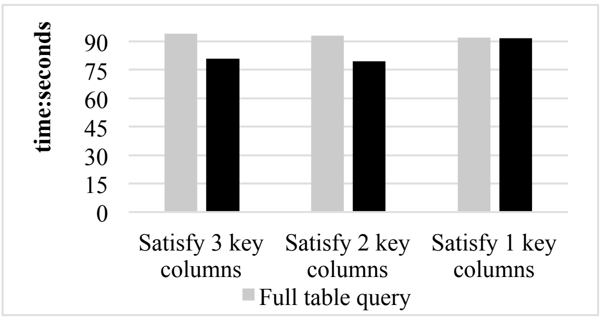

Figure 6. The Four groups of query test results

As shown in Figure 7, the abscissa represents the ratio of three or two key columns in the query request, and the ordinate represents the time required for the query. The experiments were carried out 1000 queries in turn, and the queries consisted of three key columns, two key columns, and one key column. The higher the proportion of three key columns or two key columns in a hybrid query, the shorter the query time. In the experiment, seven experiments were conducted. When the proportion is 30%, the query time is 92.5 seconds; when the ratio is 40%, the query time is 90.8 seconds; when the proportion is 50%, the query time is 88.3 seconds; when the proportion is 60%, the query time is 86.1 Seconds; when the ratio is 70%, the query time is 84.3 seconds, when the ratio is 90%, the query time is 80.9 seconds. The experimental results are in agreement with the conclusion. The higher the ratio of N/2 key columns in the query request, the shorter the query time, and the less the number of tasks and the number of working nodes, the higher the throughput of the system.

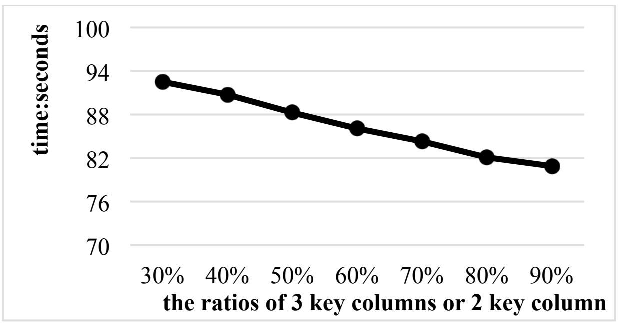

Figure 7. The Mixed query test results

In summary, the data storage technology based on the key column pre-processing can effectively reduce the invalid query and improve the system throughput, but when the query request contains less than or equal to N/2 (rounded down) key column, the query results are the same as the original system.

Conclusion

In this paper, we present a distributed structured data storage and query technology for commodity selection applications. In the data storage stage, the data is partitioned and stored by key column pre-processing. In the data query phase, when a query request containing a key column is processed, the query task can be generated only for the storage region containing the result information, reducing the number of full table queries and the number of query nodes. The technology is mainly applied to the application of commodity screening category, In the case of key column requirements unchanged, can significantly improve system throughput, it has obvious effect on reducing the latency and high workload of the existing e-commerce system in the high concurrent query.

Acknowledgements

Funding for this research was provided by the Program for Beijing Top Young Talents Cultivation Project(CIT&TCD201504057). We like to thank anonymous reviewers for their valuable comments.

References

1. Aysan Rasooli, Douglas G. Down. High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion. IEEE **2012**:1284-1291 (2012)
2. Yazd S A, Venkatesan S, Mittal N. Proceedings of the IEEE Symposium on Reliable Distributed Systems, **34(1)**:457-462 (2012)
3. Lu J, Guting R H. Proceedings of the IEEE International Conference on Parallel & Distributed Systems, **90(1)**:738-743 (2012)
4. Lu H, Hai-Shan C, Ting-Ting H. International Conference on NETWORKING & Distributed Computing, **2012**:59-63 (2012)
5. Kala Karun A, Chitharanjan K. Information & Communication Technologies, **2013**:132-137 (2013)
6. Farag Azzedin. International Conference on Collaboration Technologies and Systems. **2013**:155-161 (2013)
7. Kalavri V, Brundza V, Vlassov V. IEEE International Conference on Cloud Computing Technology and Science. IEEE Computer Society, **2013**:250-257 (2013)
8. Kurazumi S, Tsumura T, Saito S, et al. 2012 Third International Conference on Networking and Computing. IEEE Computer Society, **2012**:288-292 (2012)

9. He Y, Lee R, Yin H, et al. Data Engineering (ICDE), 2011 IEEE 27th International Conference on. IEEE, **2011**:1199-1208 (2011)
10. Liu SW, Kong LM, Ren KJ, et al. Chinese Journal of Computers, **34(11)**:2121−2130 (2011).
11. Zheng P. Chinese Journal of Computers, **33(8)**:1472-1480 (2010).
12. Lin W W. Journal of South China University of Technology, **40(1)**:152-158 (2012).
13. Zhou S. Nonferrous Metals Processing **41(6)**:53-55 (2012)
14. Eltabakh M Y, Tian Y, Zcan F, et al. Proceedings of the Vldb Endowment, **4(9)**:575-585 (2011)
15. Agarwal S, Dunagan J, Jain N, et al. Usenix Nsdi, **2010**:17-32 (2010)
16. Xie J, Yin S, Ruan X, et al. IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2010, **2010**:1-9 (2010)