Region Clustering Based Multiple Query Optimization

Pei Xuan Fei[1, a *], Cai Chen[2,b] and Yi Liang[3,c]

[1] Beijing University of Technology, Beijing, China

[2] Beijing University of Technology, Beijing, China

[3] Beijing University of Technology, Beijing, China

[a]feipeixuan@163.com, [b]chencai@bjut.edu.cn, [c]yliang@bjut.edu.cn

Keywords: multiple queries optimization, query sharing, region clustering

Abstract. In order to solve the problem of low efficiency in dealing with large numbers of queries in Hive, we propose a multi-queries region clustering (MQRC) method to improve the overall performance. This method makes effective use of the region overlap between queries and the region aggregation of query sets. By clustering the query set, we divide some queries with high similarity into a group, and share a materialized view within the group, which can effectively reduce the overhead of the query set. The experimental results show that, the method proposed in this paper is superior to the traditional method. It can shorten the time of queries.

Introduction

With the rapid development of Internet and Internet of things, the Internet will produce a lot of data every day and need to be processed. The emergence of other distributed data processing systems such as MapReduce[1] and Spark[2] solves the problem of massive data processing. For massive data, a large number of queries are usually needed to carry out statistical analysis of the data. Users make statistical analysis of data records by writing SQL query statements. These SQL statements run in batch processing, and the number of these statements is huge. Hive is a set of data warehouse analysis system based on Hadoop. It provides SQL query mode to analyze data stored in HDFS file system, and can transform SQL statement into MapReduce job.

The emergence of Hive, a high-level language, simplifies user programming and improves work efficiency. However, all of the optimization work of Hive is carried out around a single query plan, and there is no effective optimization for multiple queries. In fact, there are a large number of common regions between multiple queries, which means that queries have the overlapped query regions. At the same time, there are regional aggregation in multiple queries, and most of the queries are concentrated in some hot query regions.

In this paper, a multi-queries region clustering (MQRC) method is proposed. By partitioning query space and clustering the query sets, the complete query region is divided into multiple small query regions, which reduces search space and improves overall computation efficiency.

Related Work

Multi query optimization[3] is a hot issue in the research field of database. The problem describes how to efficiently use the common task to optimize multiple query statements to produce optimal execution results. By given a set of queries, each query has a set of alternative execution

plans and each plan has a set of tasks. MQO aims to choose an appropriate plan for each query and minimize the total execution time by performing common tasks only once.

The study of MQO problems can be traced back to the 1980s,Sellis[4] was the first systematically presents the formula of the definition of the problem and proposed a heuristic algorithm to solve the MQO problem, but also proved mathematically that the MQO problem is a NP-Hard problem.

In view of the problem of multi query optimization, there are some research achievements[5], which are divided into the following ways:

1) Single query optimization technology

Single query optimization technology considers a single query at a time and tries to generate the most efficient solution by examining and finding cheaper alternative plans.However, in the MQO problem, a suboptimal plan may be preferred for a query if it results in more common tasks with other queries in the set and thus yields global plan featuring a lower cost for the query set.

2) Materialized view technology

Materialized view technology caches intermediate results for some queries to be used by other queries. In the introduction of the framework m2r2[6], it is mentioned that the usual optimization query method is to avoid repeated computation by caching query results. Restore[7]also achieves job sharing and avoids redundant computation caching the results of a complete job or part of the job in advance.

3) Multi-Q

Multi-Q[8] is another way, this paper mentioned that most of the current research works are focus on improving the performance of query processing based on the view of systematics while without considering the characteristics of queries themselves, such as the query similarity, which will cause large numbers of redundant computation, effect query execution efficiency. Multi-Q present a multi-queries reuse dependence graph construction method to depict the dependence between the multiple queries and put forward a Multi-Q processing algorithm based on MRDG to achieve the query results reuse.

Native Materialized View Method

After analysis, it is found that most of the execution time of a query is spent on reading data from the HDFS file system, and reading data is a common task. Native materialized view method can scan sharing. By performing a preprocessing job, all the records involved in multiple queries are written into a materialized view and all the queries only read data from this materialized view. Because the data size of the materialized view is smaller than the data size of the data set, the native materialized view method can reduce the search space of the query, reducing the execution cost of the query set.

Although the native materialized view technology can achieve scan sharing, it avoids reading the input data repeatedly, thereby reducing the execution cost of the query set. However, the native materialized view method does not respond well to the large number of queries. When the number of queries is large, the data size of the materialized view will be very large. Therefore, the search space of the query cannot be effectively reduced.

Multi-queries Region Clustering Method

In order to further improve the processing efficiency of multi-queries, a multiple query optimization method based on regional clustering is proposed in this paper. This method makes effective use of the area overlap between queries and the region aggregation of query sets. The basic idea is to cluster the query set and group the queries with higher similarity into one group. Each group produces a materialized view, dividing the large query space into multiple small query spaces, making the query from load data in a smaller query space. Compared with the native materialized view method, this paper divides similar multiple queries into one group. By sharing the materialized view in the group, the execution cost of the query set can be reduced.

Query Space Partition

Through the analysis of the where clause of the query, the relevant columns of the where clause are obtained, and each column is divided according to the given granularity. Finally, the entire large query space is divided into a series of small query units. According to the WHERE statement restriction of the query, the region covered by the query can be approximated by a set of query units.

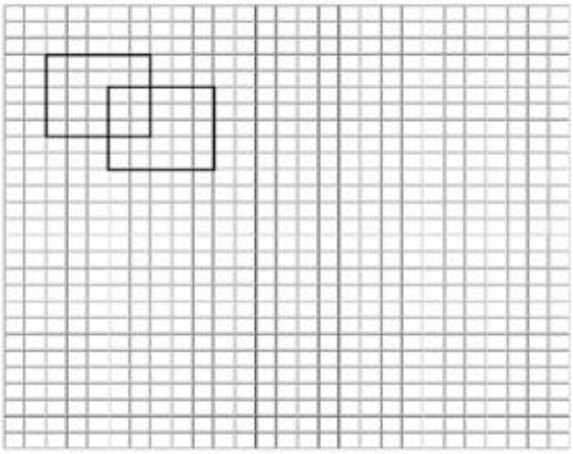

Fig.1 Two queries and query space

Query Similarity Definition

The query similarity is used to describe the degree of similarity of two queries, which can be represented by the following formula:

$$Sim(q1,q2)=\frac{Sum_{unit}(Set_{unit}(q1)\cap Set_{unit}(q2))}{Sum_{unit}(Set_{unit}(q1)\cup Set_{unit}(q2))} \quad (1)$$

If Sim(q1,q2) =0 is not similar, if it is 1, it is similar and it is the maximum match.

The average query similarity is used to describe the degree of similarity between a query and multiple other queries, which can be expressed by the following formula:

$$Sim_{avg}(q)=\sum_{i=1}^{n}(Sim(q,q_i))/n \quad (2)$$

The average query similarity of a query is high, indicating that the query is similar to other multiple queries. The average similarity of the calculated query is prepared for the later query clustering.

Query Clustering based on Greedy Algorithm

The core step of the MQRC is to build m queries into n query groups, and n≤m. The main purpose of this step is to cluster multiple queries with a high degree of similarity into one group. In the existing literature, there are a large number of techniques and methods for clustering. In general, these clustering techniques use some method to measure the distance between clustering objects, and these objects are points. However, the clustering object in this paper is a query, so it is not possible to cluster the query set using these clustering algorithms. For this reason, this paper proposes a clustering method based on greedy strategy to complete query clustering.

The core idea of this algorithm is to calculate the similarity between queries and classify the queries with higher similarity into one group. The specific steps of the algorithm are as follows:

Step 1: Calculate the average similarity of each query in turn, and sort them according to the average similarity from big to small.

Step 2: Initialize a query group and use the query with the highest average similarity in the candidate query set as the initial object.

Step 3: Calculate the average similarity of each query and query group in the candidate query set in turn, and select from them the query with the highest average similarity.

Step 4: If the average similarity of the query with the highest degree of similarity is greater than SIM_THRESHOLD, add this query to the query group and delete this query from the candidate query set, then return to Step 3. Otherwise, it means that the query is not similar to the grouping of this query and then return to Step 2. In this paper we set the threshold SIM_THRESHOLD to 0.6.

Algorithm 1 greedyClustering

Input: The list of query

Output: The list of group

Procedure greedyCluster(List queryList)

```
1:  init groups
2:  candidateSet=sortBySimilarity(queryList)
3:  while(!candidateSet.isEmpty())
4:      init group
5:      initialObject= candidateSet.getFirst()
6:      group.add(initialObject)
7:      candidateSet.remove(initialObject)
8:      while
9:          object=getMaxObject(cluster,candidateSet)
10:         if(object.value > SIM_THRESHOLD)
```

```
11:             group.add(object)
12:             candidateSet.remove(object)
13:         end if
14:         else
15:             groups.add(group);
16:             break;
17:         end else
18:     end while
19: end while
20: return groups
```

Query Processing

Before executing the query, perform a preprocessed MapReduce job to generate multiple materialized views. This paper uses the MultipleOutputs module provided by MapReduce to implement multiple output. Execute a preprocessed MapReduce job and output multiple files to the HDFS file system. Each file is a materialized view and each materialized view corresponds to a query group. All queries in a group share a materialized view.

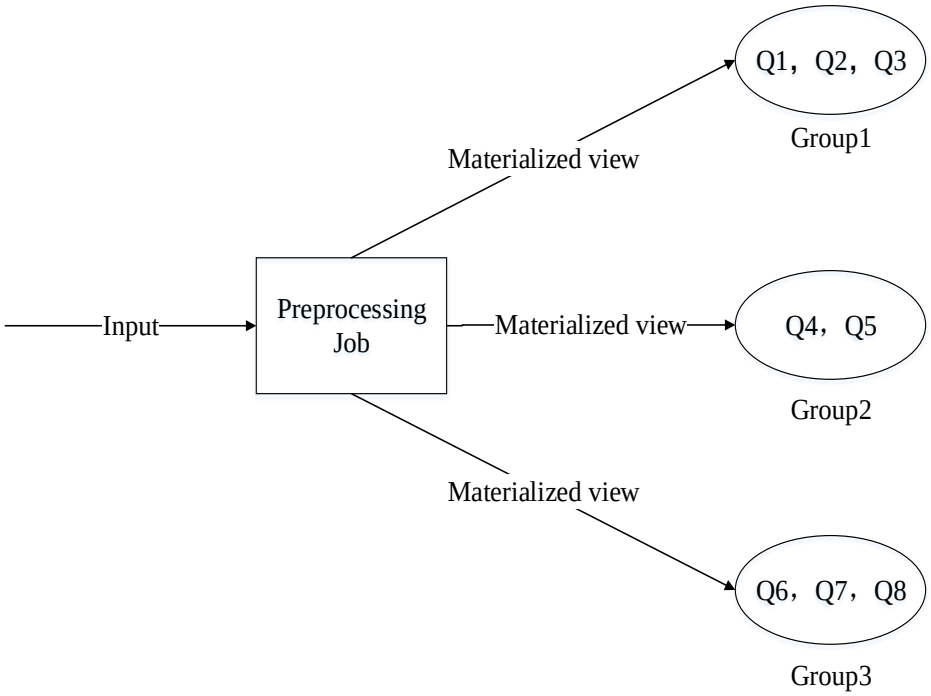

Fig.2 Region clustering based multiple query optimization

Experiment

Environment Introduction

Experiments in this paper uses a three nodes Hadoop cluster, in where there are 2 data nodes, 1 name node. Each node of the cluster has Ubuntu 14.04 installed on it, Hadoop version is 2.7 for this experiment, JDK version is 1.8 and Hive version is 2.1.1. And each node contains 2 core CPU, 4GB memory and 120 GB disks.

Method and Result

The experimental data are generated by the benchmark test set TPC-H. TPC-H is widely used in the industry to test the comprehensive performance of complex decision support systems. In this experiment, we generate a data set of 20GB through the TPC-H tool, import it into the hive in the parquet format and generate queries based on the following query template:

```
select l_returnflag, l_linestatus,sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge
    from lineitem
    where l_extendedprice between x and x+y and l_quantity between x and x+y
    group by l_returnflag, l_linestatus order by l_returnflag, l_linestatus
```

Where the parameters x and y are some random values generated by the query generator. We use a query generator to generate 100 queries randomly and submit these queries. Moreover, we use the processing time of the query set as the metric.

For comparison purposed, we implement three query processing methods that are: (1) Hive handles each query independently. (2) Native materialized view method. (3) Multi query region clustering (MQRC) method is proposed in this paper.

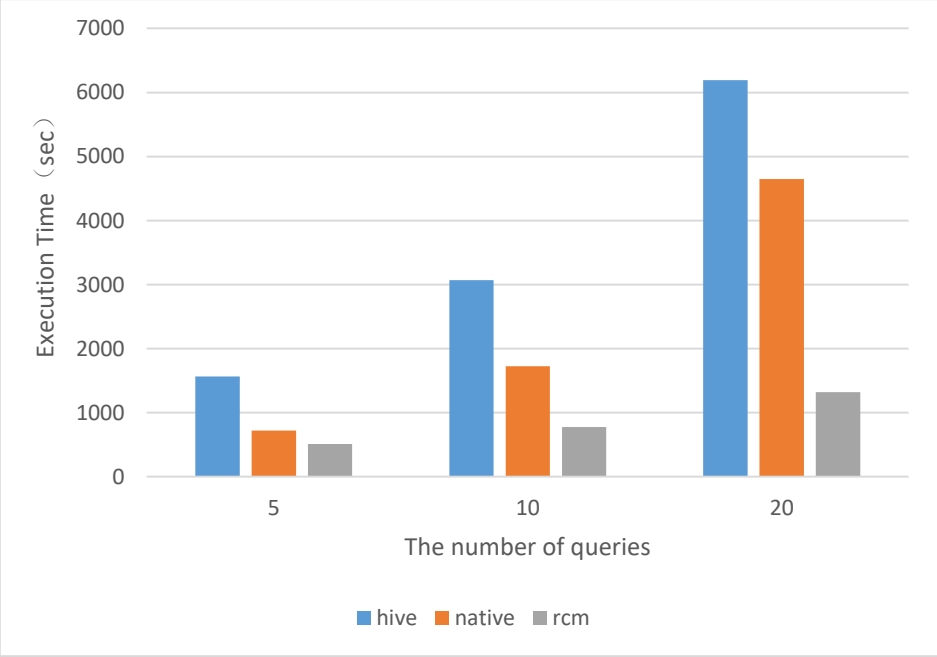

Fig.3 The comparison of performance

From the experimental results, it is easy to see that the MQRC method is faster than the native method. This is because the MQRC method effectively uses the regional overlap between queries and the regional aggregation of the query set. By clustering the query set, some queries with high similarity are divided into a group, and materialized views are shared within the group.

Conclusions

In this paper, we propose a multi-queries region clustering (MQRC) method to support multiple queries processing. By clustering the query set, we divide some queries with high similarity into a group, and share a materialized view within the group, which can effectively reduce the overhead of the query set. The experimental result verify that our method is much better than traditional methods.

References

[1] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[C]//Conference on Symposium on Opearting Systems Design & Implementation. 2008: 10–10.
[2] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, et al. Spark: cluster computing with working sets[C]//Usenix Conference on Hot Topics in Cloud Computing. 2010: 10–10.
[3] SELLIS T, GHOSH S. On the multiple-query optimization problem[J]. Knowledge & Data Engineering IEEE Transactions on, 2002, 2(2): 262–266.
[4] SELLIS T K. Multiple-query optimization[J]. Acm Transactions on Database Systems, 1988, 13(1): 23–52.
[5] SAHAL R, KHAFAGY M, OMARA F. Comparative Study of Multi-query Optimization Techniques using Shared Predicate-based for Big Data[J]. International Journal of Grid and Distributed Computing, 2016, 9: 229–240.
[6] KALAVRI V, SHANG H, VLASSOV V. m2r2: A Framework for Results Materialization and Reuse in High-Level Dataflow Systems for Big Data[C]//IEEE International Conference on Computational Science and Engineering. 2013: 894–901.
[7] ELGHANDOUR I, ABOULNAGA A. ReStore: Reusing Results of MapReduce Jobs[J]. Pig .in Proc.acm Sigmod, 2012, 5(6): 701–704.
[8] DING D, DONG F, LUO J. Multi-Q: Multiple Queries Optimization Based on MapReduce in Cloud[C]//Second International Conference on Advanced Cloud and Big Data. 2014: 100–107.