# Building quotient cube with MapReduce in hadoop

Juan Zhang
Department of Computer Science and Technology,
East China Normal University,
Shanghai 200241, China
violet1987_love@126.com

***Abstract*—In order to solve the problem that how to improve the efficiency of query and calculation in massive data, a method of building quotient cubes in Hadoop plateform which combined the advantage of the quotient cube and MapReduce model is proposed in this paper. At first, all cubes will be established and their aggregate value will be calculated in the Mapping stage. All the key/value pair formed in Mapping stage will be passed to Reducing stage. Equivalence partitioning will be carried out In this stage, and the minimum aggregation cube of each equivalence partitioning will be the key with its aggregate value. According to the minimum aggregation cubes, we can get the quotient cubes. In order to improve the speed of parallel computing and reduce network traffic, equivalence class division will be executed locally after the Map stage, it is named as combiner stage. In this paper, MapReduce model is used to improve the efficiency of building quotient cube because of its ability of parallel computing in a large amount of data. In addition, the experiment proved that, under certain circumstances, increasing the number of Mapper/Reducer task can reduce the building time effectively, and improve the construction efficiency.**

***Keywords- Data warehousing, Quotient cube, Hadoop plateform, MapReduce model; cloud computing Datas***

## I. INTRODUCTION

Since data warehouse has been proposed in 1990s, many scholars are committed to the study of the data cube compression and have got amount of research achievements. Quotient cube is one of the most important research achievements. Quotient cube is one of the most important research achievements which can keep the semantic of data cubes well. It can support the operations of Roll-up and Drill-down, so that it is easy to calculate the new cubes.

In recent years, with the rapid development of network applications, more and more sites can be generated daily data to TB level, such as microblogging, e-commerce sites and so on. It has become difficult that improving common data warehouse system to make them be enable to effectively store and analyze the data. The emergence of cloud computing systems can solve these problems effectively. Since 2004, Big Table, GFS(Google File System) and MapReduce model had been proposed by Google step by step. Based on these research achievements in Google, cloud computing platforms have been developed rapidly. Among these platforms, Hadoop is one of the most widely used plateform not only in academia but also in commercial.

In this paper, we combed the advantages of the quotient cube and Hadoop platform, and finished building quotient cube with the MapReduce model in Hadoop. MapReduce model is one of the important parts of Hadoop, it support parallel computing to speed up the processing of the data. In the Map function ,we get all cubes which can be computed by Roll-up or Drill-down; in the Reduce fuction , we merged equivalence class and abtain the minimum aggregated cubes. The collection of all the minimum aggregated cube will be quotient cube. We proposed and implemented the algorithm of building quotient cube, it is the foundation for our subsequent work in future.

## II. RELATED WORK

Date cube is an important concept and research direction in OLAP(Online Analytical Precessing). In recent years, the researches on the data cube are mainly in two aspects: first, how to compress and store cube. Due to the massive data, storing all the data cubes needs a lot of space and resources. As a result, the effective method on compression and storage becomes an important research topic. Second, how to choose and materialize the data cubes. In order to help customers gain effective data, many scholars pay attention on finding the better method of how to choose and materialize datacubes.

Researchers have proposed a variety of algorithms on Cube compression and Storage, such as iceberg cube, quotient cube and so on. In Iceberg cube, the cubes which do not meet the requirements of pre-defined constraints will be deleted and not stored. By this means, the original cubes will be compressed. Quotient cube is based on equivalence partitioning. It keeps the semantics of the original cubes, so that it can get the results of the query on any cube easily. At the same time, some scholars have proposed the concept of the iceberg quotient cube. This cube is a combination of quotient cube and iceberg cube.

## III. RELATED CONCEPTS

### *A. Dimension and Dimension Hierarchy*

Dimension: Dimension is a particular point of view that users will concern about the data. For example, considering a company's sales, looking from the region, it can be divided into the sales in Shanghai, Guangzhou, Beijing, etc.; from the time point of view, it can be divided into sales in 2010, 2011, 2012 and so on; analysising from the perspective of products categories, it can be divided into Chocolate sales, Cookie sales, beverage sales, etc. Among them, Region, Time, Product Variety are data Dimensions.

Dimension Hierarchy: Different levels of one Dimension. A dimension hierarchy determines the category names users can used on this Dimension. These category names are also formed one-way dependence. For example, "Region" can be divided into multiple levels: Country, Privince, City, Street. There will be an Dependency: All ← Country←Privince←City←Street. Child nodes are always more refined than its parent. According to the dimension hierarchy, users can obtain the required data by means of Roll-up or Drill-down.

## *B. Data Cube*

A formal description of Data Cube is given as following:

The data cube can be represented as a six tuple form: Cube=( Dom, D, Mdom, M, f , aggr ).

*a)* $Dom = dom_1 \times dom_2 \times \ldots \times dom_n, n > 0$ *,it is called the dimension of the domain, so* $dom_i (1 \le i \le n)$ *are all domains;*

*b)* $D = \{d_1, \cdots, d_n\}$ *, it is known as the identities set on dimension, so* $d_i (1 \le i \le n)$ *is the identity of* $dom_i (1 \le i \le n)$ .

*c)* $MDom = mdom_1 \times mdom_2 \times mdom_n, m > 0$ *, it is referred to domains of the measures, so* $mdom_i (1 \le i \le n)$ *are all domains;*

*d)* $M = \{m_1, \cdots, m_n\}$ *, and* $m_i (1 \le i \le n)$ *is the identity of* $mdom_i (1 \le i \le n)$ *;*

*e)* $f : Dom \rightarrow MDom$ *, it is the partially mapped from Dom to MDom which is called cube base;*

*f) Aggr is aggregate functions on MDom.*

## *C. quotient cube*

Quotient cube is a compression method on all cubes and it can keep the semantics between cubes. First of all, cubes will be divided into some equivalence classes by dimensions, and on this basis, select the minimum aggregation cube in each equivalence class and. All the minimum aggregation cubes are gathered to be quotient cube.

Example 1. Given the data in the following table ( in this case, the aggregation function is SUM)

TABLE I. THE BASIC DATA

| VendingID | Product | Season | Sales |
|---|---|---|---|
| S1 | P1 | spring(s) | 5 |
| S2 | P2 | autumn(a) | 10 |
| S2 | P1 | winter(w) | 15 |
| S3 | P2 | winter(w) | 10 |
| S1 | P2 | autumn(a) | 5 |
| S3 | P1 | spring(s) | 8 |

TABLE II. EQUIVALENCE CLASSES AND THE MINIMUM AGGREGATION CUBE IN EACH

| ID | the minimum aggregation cube | equivalence classes |
|---|---|---|
| 1 | (S1,P1,s:5) | (S1,P1,s) (S1,P1,*) (S1,*,s) |
| 2 | (S2,P2,a:10) | (S2,P2,a) (S2,P2,*) (S2,*,a) |
| 3 | (S2,P1,w:15) | (S2,P1,w) (S2,P1,*) (S2,*,w) (*,P1,w) |
| 4 | (S3,P2,w:10) | (S3,P2,w) (S3,P2,*) (S3,*,w) (*,P2,w) |
| 5 | (S1,P2,a:5) | (S1,P2,a) (S1,P2,*) (S1,*,a) |
| 6 | (S3,P1,s:8) | (S3,P1,s) (S3,P1,*) (S3,*,s) |
| 7 | (S3,*,*:18) | (S3,*,*) |
| 8 | (S1,*,*:10) | (S1,*,*) |
| 9 | (*,P1,s:13) | (*,P1,s) (*,*,s) |
| 10 | (*,P2,a:15) | (*,P2,a)(*,*,a) |
| 11 | (*,*,w:25) | (*,*,w) |
| 12 | (*,P2,*:25) | (*,P2,*) |
| 13 | (*,P1,*:28) | (*,P1,*) |
| 14 | (S2,*,*:25) | (S2,*,*) |
| 15 | (*,*,*:53) | (*,*,*) |

## *D. MapReduce in Hadoop*

At OSDI(Operating Systems Design and Implementation ) meeting in 2004, MapReduce programming model was been proposed by Google, and it has been applied to the implementation of some of Google's module. The MapReduce model greatly reduces the difficulty of parallel progressing of data in a large cluster. As an open source cloud computing platform, MapReduce is also implemented in Hadoop.

In MapReduce, calculation is divided into two stages: Map stage and Reduce Stage. The processing method is defined for each sun-block of data in Map stage; Reduce phase defines the reduction algorithm of the intermediate results which are gained in Map stage. In order to reduce network traffic, you can add a Combine stage which is executed after the Map stage. In essence, it is a Reduce process executed in the local. In MapReduce model, many tasks are done automatically such as task scheduling, etc, so it can reduce the programmers' jobs as far as possible.

## IV. ALGORITHMS ON QUOTIENT CUBE BUILDING

In this paper, it is assumed that the input data are basic cube data in constructing the quotient cube.

Quotient cube building is divided into two parts: (1)Calculation all cubes which can be gained by Roll-up operation from each basic cube, and it will be finished in Map stage; (2) Make all records from Map stage to be divided into suitable equivalence classes and collect all of the minimum gathered cubes in each equivalence class to be quotient cube.

## *A. Algorithm 1: Map (key, value, context)*

The task in Map stage is to read each row from the input data. According to the records, all dimensions and aggregation values will be gathered to form the outkey. All cubes which can obtain by Drill-down or Roll-up will be serialized to be byte stream that will be the outvalue

```
Map (key, value, context)
Input: key, byte offset; Value, records of the basic data in the table; Context: the stored information.
//key is not used in this algorithm
Output: outkey, the combination of all dimension attributes and gathered value; Outvalue, all cubes that can be obtained by Roll-up.
//outkey and outvalue are formed by special format. For example, (S1, P1, s:5), outkey will be "S1-P1-s-5", Outvalue will be "S1-P1-s| S1-*-s| S1-P1-*| *-P1-s| S1-*-* |*-*-s |*-P1-*|*-*-*"
Method:
Steps 1: Get all dimension attributes and metric value from value, then connect them with "-". It will be outkey.
Steps 2: Let D = {a, b, c,…, }; call GenerateAllCubes(D);
//a, b, c are dimension attributes; GenerateAllCubes(D) is a function to get all cubes from dimensions
Steps 3: Combine the results of GenerateAllCubes(D) with "|", it will be outvalue.
Steps 4: Call MapReduce API, context. Write (outkey, outvalue)
Function: GenerateAllCubes(D);
Steps 1: String s, put all dimension attributes into s;
Steps 2: For each dimension attributes x in s ( for i = 0 to s.length)
    create new cubes according to cubes which has been created before;
```

## B. *Algorithm 2: Reduce (key, value, context)*

In Reduce stage, all cubes will be divided into the equivalence classes and the CountStore will be calculated by the aggregate values and function. Quotient cube will be got by these equivalence classes.

In MapReduce, a Map node can be called multiple times to handle multiple records, then it will write the results in the local disk. If Reduce function can be execyted in local firstly, and then call for Reduce node through the network, it can reduce network traffic. Combiner stage can complete this operation.

```
Reduce (key, values, context)
Input: key, outkey which is formed in Map stage; value, outvalue, which is formed in Map stage; context, the stored information such as gathered method.
Output: outkey, quotient cube and aggregated value.
Method:
Step 1: Read from context to get the aggregate function;
Step 2: Make the key value to be a minimum aggregation cube, and construct an equivalence class for each;
Step 3: Split values by "|", and then traversing all split results, if it does not appear, check that it can be Roll-up by the cube in key; else create an new equivalence class and calculate the value with the aggregate function, at the same time, delete the cube from where it was existed originally;
//For example: (S1-P1-s: 5, "S1-P1-s| S1-*-s| S1-P1-*| *-P1-s| S1-*-* |*-*-s |*-P1-*|*-*-*"), the first equivalence class will be itself. Then the value will be split to be (S1-P1-s), (S1-*-s), (S1-P1-*), (*-P1-s), (S1-*-*), (*-*-s), (*-P1-*), (*-*-*). If the cell is (S2-P2-*), check that it can be Roll-up by the cube (S2-P2-a); if the cell is (*-*-*), there will be a new equivalence class created and its key is (*-*-*:15), and at the same time, (*-*-*) in (S1-P1-s: 5) will be deleted.
Step 4: Make the aggregate function to be outkey; gather the minimum aggregated cube of each equivalence class to be outvalue.
Step 5: Call MapReduce API: context. Write (outkey, outvalue)
```

In Combiner function, the operation is the same as Reduce function. The only difference is that the output of the Reduce function is preserved in the final output file, while in combiner function it is written in the intermediate file, and then will be transmitted to reduce node.

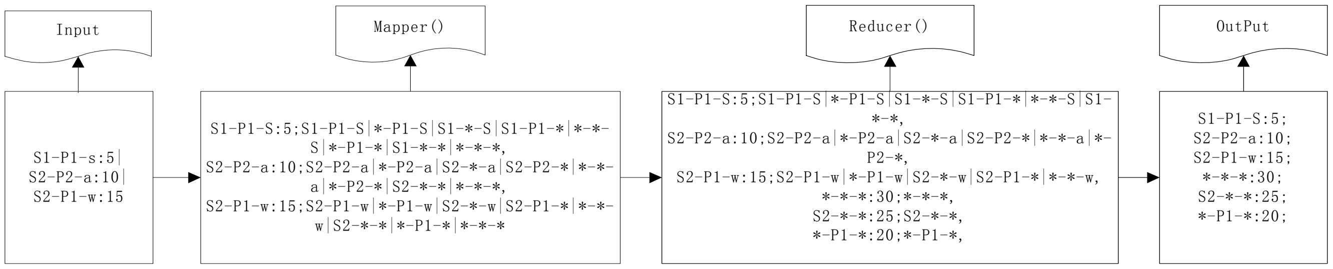
Input
Mapper()
Reducer()
OutPut
S1-P1-s:5|
S2-P2-a:10|
S2-P1-w:15
S1-P1-S:5;S1-P1-S|*-P1-S|S1-*-S|S1-P1-*|*-*-S|*-P1-*|S1-*-*|*-*-*,
S2-P2-a:10;S2-P2-a|*-P2-a|S2-*-a|S2-P2-*|*-*-a|*-P2-*|S2-*-*|*-*-*,
S2-P1-w:15;S2-P1-w|*-P1-w|S2-*-w|S2-P1-*|*-*-w|S2-*-*|*-P1-*|*-*-*
S1-P1-S:5;S1-P1-S|*-P1-S|S1-*-S|S1-P1-*|*-*-S|S1-*-*,
S2-P2-a:10;S2-P2-a|*-P2-a|S2-*-a|S2-P2-*|*-*-a|*-P2-*,
S2-P1-w:15;S2-P1-w|*-P1-w|S2-*-w|S2-P1-*|*-*-w,
*-*-*:30;*-*-*,
S2-*-*:25;S2-*-*,
*-P1-*:20;*-P1-*,
S1-P1-S:5;
S2-P2-a:10;
S2-P1-w:15;
*-*-*:30;
S2-*-*:25;
*-P1-*:20;

Figure 1. Algorithm schematic diagram with Map() and Reduce()

## V. EXPERIMENT RESULTS

### A. *Experiment Design*

In my experiment, the data is the trading records from 2008 to 2011 of an company in Shanghai, it contains transaction date, product ID, vending machine ID, total sales and so on. We make the data to be our required as the basic data in Table1.

The data cube is divided into three dimensions: VendingID, ProductID, and Season. The measure value is Stores and the aggregated method is SUM.

I design three kinds of experiment . The first one, I compared the running time of different numbers of Map at the same number of Reduce and data; the second, I compared the running time in single machine and in MapReduce which only had one Map and one Reduce task; the third, I compared the running time with different number of nodes, map tasks and reduce tasks.

## B. *Experiment results*

In figure 2, it shows the relationship between Node Number and Runtime to build the quotient cubes. In this experiment, the data is about 5.4G. I compared the running time with different number of map stage and keep the number of reduce is 0.95* the number of nodes. According to the results, I found that it can reduce the running time if I change the size of DFS block to increase the number of Map. But it doesn't mean that the more number of map, the less running time it will.

When using one work node in MapReduce, it will spend more runtime to build cubes than in single model. According to Figure 3, we can get this conclusion. On the one hand, one task will be divided into many Mapper tasks depending on the size of the block, so that it will increase the I/O operation; on the other hand, MapReduce model will spend much time on internal sorting.

In Table 3, the results show that the numbers of map and reduce should be assigned reasonably. It will be a subject in future that how to balance the number of nodes, map tasks, reduce tasks and data to improve the efficiency of building quotient cubes.

Generally speaking, in order to improve the efficiency, the numbers of map and reduce should be assigned reasonably.

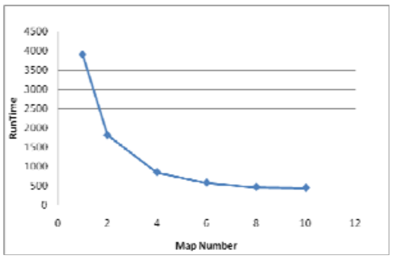

Figure 2. Relationship between Node Number and Runtime

Figure 3. Runtime with different numbers of nodes, map and reduce tasks

| Node Number | Map Number | Reduce Number | Running Time(s) |
|---|---|---|---|
| 1 | 12 | 1 | 2380 |
| 4 | 12 | 1 | 2696 |
| 4 | 12 | 4 | 1688 |
| 4 | 12 | 8 | 1391 |
| 10 | 12 | 12 | 1822 |

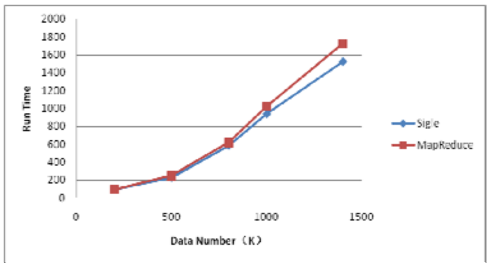

Figure 4. Runtime of MapReduce and single

## VI. SUMMARY

In this paper, we have realized the quotient cube construction with MapReduce model in Hadoop platform. The significance of this study is that data parallel computing is used on building quotient cube, as a result, it can improve the efficiency of building quotient cube on massive amounts of data.

But this is only a foundation, there are many studies needed to be done in future.

(1). In this paper, we assume that the input data for Map function are the basic cube. But in many cases, the data in fact are not the basic cube data. We need to convert the fact data to the required basic effectively. It remains to be researched.

(2). The compression algorithm only memory quotient cube, and how to improve the accuracy and efficiency of queries and calculate other cubes with quotient cubes is another problem needed to be solved.

## REFERENCES

[1] Pedro Furtadoand, Henrique Madeira. Data Cube Compression withQuantiCubes[A]. London, UK:Data Warehousing and Knowledge Discovery[C].2000:162-167.

[2] Fay Chang , Jeffrey Dean , Sanjay Ghemawat ,et al.Bigtable: A Distributed Storage System for Structured Data [J]. ACM Transactions on Computer Systems.2008,26(2):1-26.

[3] Jairam Chandar.Join Algorithms using Map/Reduce[D]. Edinburgh:University of Edinburgh,2010.

[4] Jeffrey Dean,Sanjay Ghemawat.MapReduce: simplified data processing on large clusters[J].Communications of the ACM,2008,51(1):107-113.

[5] Hadoop, http://hadoop.apache.org/.

[6] HBase, http://hbase.apache.org/.

[7] Jiawei Han, Micheline Kamber.Data Mining Concepts and Techologies[M]. Beijing: mechanical industry publishing house, 2007.

[8] Laks V. S. Lakshmanan, Jian Pei, Jiawei Han.Quotient cube: how to summarize the semantics of a data cube[A].VLDB '02 Proceedings of the 28th international conference on Very Large Data Bases[C].2002:778 – 789.