ATLANTIS PRESS
Advances in Computer Science Research, volume 62
2nd Joint International Information Technology, Mechanical and Electronic Engineering Conference (JIMEC 2017)

A Novel Processing Model For Scds In ETL

***Li Sun*[1]**
School of computer science and technology, Donghua University, shanghai , china
sli@dhu.edu.cn

***Jiaoyan Zhang*[2]**
School of computer science and technology, Donghua University, shanghai , china
zhangjy0723@163.com

***Jiyun Li*[3]**
School of computer science and technology, Donghua University, shanghai , china
jyli@dhu.edu.cn

Keywords: ETL, MapReduce, map-only

Abstract

ETL(Extract-Transform-Load) which populates data from various data source systems to data warehouses (DWs) is an important part of building data warehouse. Nowadays, as the data growing rapidly, it is a big challenge for ETL to process such huge data quickly. MapReduce is a programming model for large-scale data-intensive processing. It is composed of two functions, map and reduce, this promotes the implementation of many tasks in parallel.
However, this model has its disadvantages. For example, it is not so efficiency when the mappers produce lots of data, which will take a lot of network cost to move the Intermediate data to reducers.
In this paper, we present a new method called map-only. With this method, we do the reduce in the local and do not need to transfer the data to the reducers through the network. The result shows that the method we present performs very well, which improves the speed of processing data for both Type-1 and Type-2 SCDs. For example, when the size of increasing data is 5GB, with the map-only method, it takes only 20 minutes to process the Type-2 SCDs while it costs 28 minutes to process the same data.

1 Introduction

In a data warehouse [1] project, ETL(Extract-Transform-Load) is used to collect data from relevant source systems, then transform them into proper format and load the final data into DW(data warehouse). Generally, more than 70% of the efforts for building a data warehouse is put in ETL. As the continuously increasing amounts of data, traditional ETL flows face great challenges to process such hundreds of gigabytes data. Many enterprises want to spend less time to load the large amount of business data into DW through ETL process so that leaders can make better decision.
With the development of cloud computing, MapReduce [3], a parallel computing programming model for large-scale data set has been popular. A MapReduce program usually includes map and reduce functions, which are used for processing key/value pairs. Many enterprises begin to deploy their data analysis systems on clusters of cheaper machines for the cost saving and the high-efficiency of distributed ETL. However, MapReduce is not good at processing some specific constructs in ETL. Eg. Star schema [2] consisting of a fact table and many dimension tables is widely used in DW. To tackle the problem of changing dimension during DW building, the slowly changing dimension(SCD) mechanism is a commonly adopted. This makes it low efficiency for ETL flows. In order to implement complex ETL-specific activities on MapReduce, many researchers present all kinds of novel methods. Hung-chih Yang et al [4] proposed a new model called Map-Reduce-Merge, it added a merge phase to merge data after the map and reduce modules. The merge task will take the intermediate outputs and joins them. Although this method improves the efficiency of data processing to a certain extent, it is hard to restore when something is wrong with job scheduling and node communication. Xiufeng Liu et al[5] proposed CloudETL, a scalable dimensional ETL framework for Hive. This ETL framework parallelizes ETL execution and loads data into Hive based on Hadoop. However, when the number of nodes of processing data becomes larger, it takes a lot of network cost and load imbalance often occurs.

In this paper, we present a new method called map-only. With this method, we do the reduce in the local and do not need to transfer the data to the reducers through the network.
The rest of this paper is structured as follows. In Section 2, we will show the background of this paper. In Section 3, we detail the method Map-only. In the section 4, we do the experiment and compare the results with and without using the Map-only method. In the Section 5, we summarize the paper.

Running Example: To illustrate the method more clearly, we use the following example (show in fig.1). The example considers a star schema about tests of web pages, which includes a fact table and three dimension tables. The measure errors in fact table testresultfact shows the total errors are found on the given web page in a specified day. The dimension table pagedim is obvious a Type-2 SCDs that records the change of the attribute values such as the size of the given page with time going. The attribute version expresses the version number while the validfrom and validto represent the valid data of the given version. The dimension tables testdim and datedim represent tests and dates of the current pages, respectively. Here we just discuss the processing of pagedim.

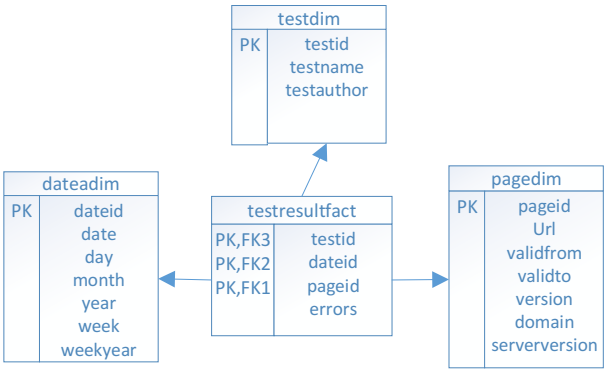

fig.1 Star schema of the running example

2 Background

In this section, we will introduce Mapreduce, Hadoop Distributed File System(HDFS) and Extract-Transform-Load (ETL).

2.1 MapReduce

MapReduce, first proposed by Google, is a programming model used for processing and generating large data sets. This model includes two basic functions: map and reduce functions:
Map :(k1, v1) ->[(k2, v2)]
Reduce: (k2, [v2]) -> [v3]
The map function is used to build data set, it is the preparation of data. It has two input variables: key k1 and value v2. Then it transforms them into a list of intermediate key/value pairs based on user-defined logic. These key/value pairs will be partitioned according to k2. The value lists which have the same key are in the same group. The reduce function completes calculations such as maximum and average, etc. It also has two input variables intermediate key k2 and the list of intermediate value [v2], then reduce function aggregates the list of value[v2], output the final list of [v3]. fig.1 shows the MapReduce Architecture.

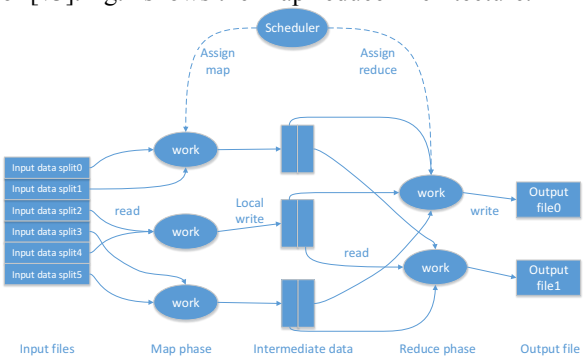

fig.2 MapReduce Architecture

In fig.2, we can see that an input data set is usually divided into many data splits, then map deal with them in parallel and output the intermediate data. After shuffling and sorting, the intermediate data are sent to the reduce. After processing in the reduce, final data are output to the HDFS.

2.2 Hadoop Distributed File System(HDFS)

HDFS influenced by GFS is an open-source file system. It is used for handling lager amount of data with the optimized methods. HDFS has the characteristics of high fault tolerance and it also provides high throughput to access the application's data, so it is suitable for those applications with very large data set. Moreover, it is designed to deploy on the cheap hardware, which makes it widely used.
HDFS is made up of one namenode and many datanodes. The HDFS namenode manages the directory of the file system, deals with the read and write requests as well as configurates the copy strategy. Besides, namenode also manages the namespace which records the information of user's operation (such like create, update, delete) to the file. The HDFS datanodes are mainly used for operating data blocks. Files are splitted and stored on datanodes in the form of blocks of fixed size (configurable and defaults to 64MB).In order to recover data quickly and automatically in the case of hardware and software failures, HDFS has data replication and the default factor is 3. Notice that files can be written only once and no updating is allowed to the existing files.
When a user want to read the file, request must be sent to the namenode. After namenode queries the metadata tables, it return the metadata of the file to the user and cut off the connection with the user. Then the user can directly access to the related data and get the whole file. If a user wants to write the file, request also should be sent to the namenode and namenode write the file name into namespace. The file will be splitted into many blocks according to the size of the file. Then Namenode will query the metadata tables to acquire the free blocks and return the information to the user. Finally, the user connects with the Datanodes and writes the file into the blocks with write permissions.

2.3 Extract-Transform-Load (ETL)

ETL is playing an important role in Data Warehouse. It is quite complicated and time-consuming, which is due to the plethora of ETL-specific technological process. Thus, lots of scholars dedicated to study the ETL process for improving the efficiency of the data warehouse. The ETL process ETL is mainly composed of three parts: extraction, transformation, loading.
a. Extraction
Extraction is a process that identifies data from various sources and extracts the data efficiently. It goes through files or databases to select and find suitable data using kinds of criteria.
b. Transformation
Transformation is a process that helps change data from different sources into the format which used in the data mart or data warehouse. In other words, it cleanses and integrates data from diversiform schema to the defined schema, which makes it meaningful to the next process.
c. Loading
Loading is a process that moves the transformed data from operational system into data warehouse. For dimension table and fact table, there are different strategies.

3 Map-only

It is not efficient for moving a large amount of data from mapper to reducer. Here we present a method called map-only to process dimension data. This method makes good use of data locality in HDFS.

This method has two steps, the first step is to create data warehouse and the second step is to process incremental load.

a.creates data warehouse

At first, there is no data in the Data Warehouse, so we need to create data warehouse. When we input the data files, we partition the records according to the key (here we use the url as the key).The records have the same key are in the same partition. Assuming that the number of the partitions is n as see in the left of fig.3. Then we do map and reduce in the local partitions and put the final results into data warehouse.

b. incremental load

After the first step, the data are stored in the data warehouse as n partitions, each partition has its own key and stores the records that have the same key. The Data Warehouse is shown as fig.3. Now we consider the incremental load, we load the data which has exists in the warehouse into Hadoop, and put it into datanodes. Each datanodes contain a partition. Then we load the incremental data and use the same partition function as before to partition it, the key of the record, which is the same as the previous partition, is placed together. Then do the map-only job in the local. After the map, load all the partition to data warehouse.

We will use fig.4 to explain the method. We assume there is no data in the data warehouse at first. The input data is partitioned into n partitions (p1, p2, … pn) and partition1 is placed in node1, partition2 is placed in node2 and so on. In the local datanodes, we do map and reduce. Then we load the incremental data into a mapper to partition the data according to the key and results in k files (r1, r2, …, rk). Compared with the data in the datanodes, the files have the same key are put in the same datanodes. For example, if the key of r1 is same as the key in the node1 then r1 is sent to node1. After k files(r1, r2, …, rn) are in datanodes, then do the map-only job in the local. When we finish the job, we load all the partitions into data warehouse according to their keys.

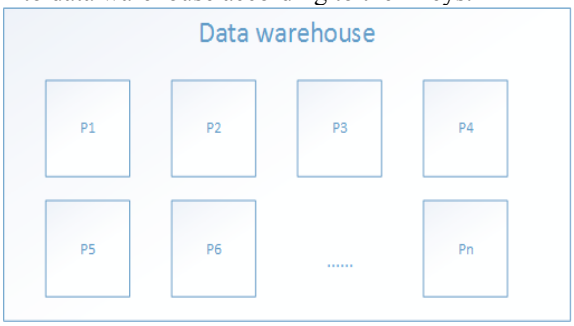

fig.3 Data warehouse

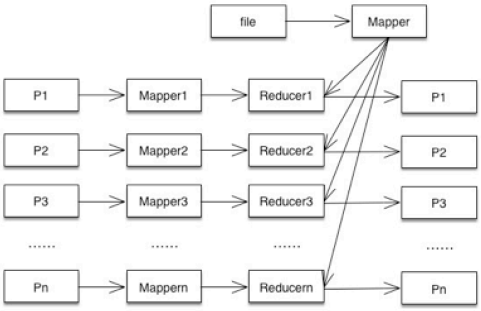

fig.4 map-only

In order to make the above more concrete, we will use fig.5 to explain the map-only method. As show in fig.5, we assume that there is already some data in the data warehouse. The map input includes the increasing data and the existing dimension data. A record from the incremental data has the attributes url, size, moddate. The existing data has the Type-2 SCDs attributes version, validfrom and validto. In the mapper, we transform the incremental data into dimension values by adding the SCD attributes. For the existing data, nothing is done. Then we do local reduce, the map output is shuffled to the reducers. The key-value pairs are grouped by url and the unique numbers id are sent to the key attribute. The value of SCD attributes are updated. The valueto is updated to the starting valid of the following version (see validfrom and validto of different version for the url=www.p1.com), the version number is also updated. All the records are finally written to HDFS.

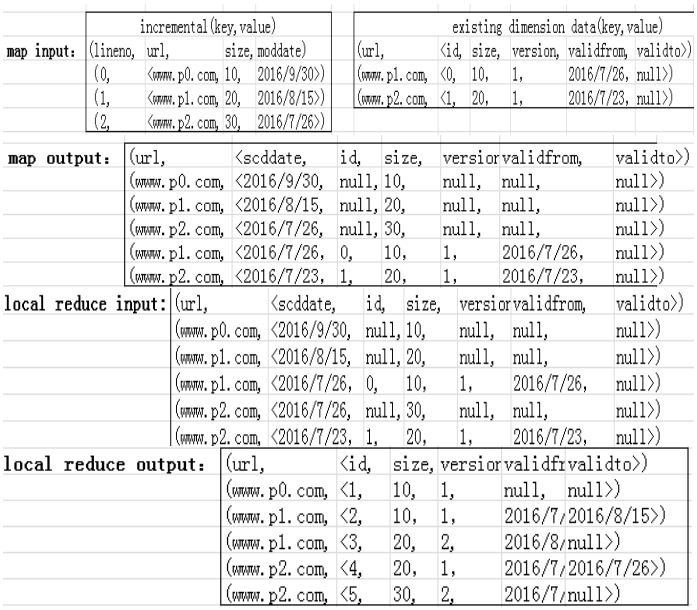

fig.5 MapReduce input and output when processing the Type-2 SCD pagedim

4 Experiments

In this section, we evaluate the performance of the map-only method. Our experiment is performed on a local cluster of 4 machines: One master and three slaves. We use four computers which have CentOS6.5（64bit）system and in the same local area network. Master used to run Hadoop cluster

of three processes, which are the NameNode, SecondaryNameNode and the ResourceManager. Besides, Master is also responsible for managing the HDFS and resource scheduling. Three slaves are used to run the DataNode and NodeManager.

Here we compare the initialization and the increment of Type-2 SCDs with the map-only and without the map-only method. As we see the fig.6 and fig.7, it is obvious that using the map-only method cost less time. As the size of data increasing, the time for using map-only method is nearly 50% less than that of not using the map-only method. Besides, we also compare the initialization and the increment of Type-1 SCDs. See in the fig.8 and fig.9, we reach a conclusion that with the map-only method, it is much more efficient.

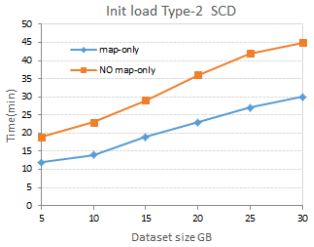

fig.6 Init load Type-2 SCD

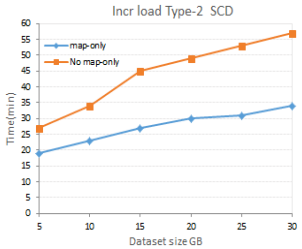

fig.7 Incr load Type-2 SCD

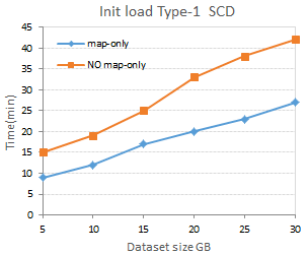

fig.8 Init load Type-1 SCD

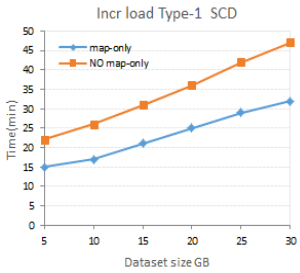

fig.9 Incr load Type-1 SCD

5 Conclusion

With the growing amount of data, it becomes a big challenging for data warehousing technologies to process the data. This paper presented the Map-only method, unlike the traditional map and reduce, this method does the reduce in the local. We do not need to transform the data from mappers to reducers through the network. Because of this, it takes less time to process the data of Type-1 and Type-2 SCDs especially with the increasing size of data. The result of the experiments shows that when the size of data is small, the difference between using Map-only and not using Map-only method is very little. Once the size of data gets larger, the difference becomes bigger. So it is obvious that the Map-only method is very efficient.

Acknowledgements

I would like to express my gratitude to all the people who helped me for writing this paper. I will show my deepest gratitude to Professor Sun li and Li jiyun. They gave me many help during my study in Donghua University. Without their help, I won't finish this paper so quickly.

Besides, I would like to express my heartfelt gratitude to my classmates, they encouraged me and helped me so I never gave up even I met great trouble.

References

[1] Inmon W H. Building the Data Warehouse[M]. John Wiley & Sons, Inc. 1996.

[2] R. Kimball and M. Ross. "The Data Warehouse Toolkit".John Wiley and Son, 1996.

[3] Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: Proc.of OSDI, pp. 137–150 (2004)

[4] Yang H C, Dasdan A, Hsiao R L, et al. Map-reduce-merge: simplified relational data processing on large clusters[C]// ACM SIGMOD International Conference on Management of Data, Beijing, China, June. 2007:1029-1040.

[5] Liu X, Thomsen C, Pedersen T B. CloudETL: scalable dimensional ETL for hive[C]// Proceedings of the 18th International Database Engineering & Applications Symposium. ACM, 2014:195-206.

[6] Srikanth K. Data Warehousing Concept Using ETL Process for SCD Type-2[J]. American Journal of Engineering Research, 2013, 2(4).