Multiple Decisional Query Optimization in Big Data Warehouse

Ratsimbazafy Rado, University of Lyon, Bron, France
Omar Boussaid, University of Lyon, Bron, France

ABSTRACT

Data warehousing (DW) area has always motivated a plethora of hard optimization problem that cannot be solved in polynomial time. Those optimization problems are more complex and interesting when it comes to multiple OLAP queries. In this article, the authors explore the potential of distributed environment for an established data warehouse, database-related optimization problem, the problem of Multiple Query Optimization (MQO). In traditional DW materializing views is an optimization technic to solve such problem by storing pre-computed join or frequently asked queries. In this era of big data this kind of view materialization is not suitable due to the data size. In this article, the authors tackle the problem of MQO on distributed DW by using a multiple, small, shared and easy to maintain shared data. The evaluation shows that, compared to available default execution engine, the authors’ approach consumes on average 20% less memory in the Map-scan task and it is 12% faster regarding the execution time of interactive and reporting queries from TPC-DS.

KEYWORDS

Big Data Warehousing, Data Sharing, Distributed Environment, Multiple Query Optimization, OLAP, View Materialization

INTRODUCTION

The business intelligence system has evolved since the growing interest in big data, and therefore data warehouses and OLAP (Online Analytical Processing) (Chaudhuri et al. (2011)). New models of data warehousing (DW) have appeared (Figure 1). We can cite, as examples: (1) DW based on relational database management systems (RDBMS), for example Teradata¹, Greenplum²; and (2) DW based on MapReduce paradigm (Dean and Ghemawat, 2004) like Hadoop/Hive (Thusoo et al., 2010a).

Inside a big data warehouse system (Hadoop/Hive) there is still a physical independence between the data warehouse structure and data storage, as we have seen in RDBMS. The main difference relies on data storage. In Hadoop/Hive, data is stored following Hadoop Distributed File System (HDFS) method (block files, data replication, etc.). Nonetheless, there is a convergence between Hadoop/Hive and traditional RDBMS in how to retrieve information from the file system with SQL (Structured Query Language). We only specify what information we need, and the system will compute it to obtain the data from the storage unit. In Hadoop, a MapReduce framework is used to retrieve all data from HDFS and compute them to give the query result.

DOI: 10.4018/IJDWM.2018070102

Copyright © 2018, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
Since the first version of Hadoop/Hive, it has evolved to become more convenient to use for companies and researchers, but there are still several challenges to tackle, in the big data warehouse and OLAP (Cuzzocrea et al., 2013). One of those challenges is the end-user performance in query execution, particularly in case of multiple-query or intensive-query execution. With the constantly increasing amount of data, we are also facing a growing number of users. In this paper, our primary concern is the number of concurrent queries from users. The problem is known, in RDBMS, as multiple query optimization, or MQO (Sellis, 1988).

In this work, we are interested in the query execution in DW that uses MapReduce framework, particularly on multiple-query execution with MapReduce. As far as we know, in MapReduce framework, all queries are processed in parallel with Map and Reduce tasks and distributed on a cluster of computers. To execute a single query, MapReduce evaluates the query and generates the list of Map and Reduce tasks (Figure 3). During multiple-query execution MapReduce evaluates all queries one at a time, producing an execution latency and an enormous resource consumption, especially due to multiple read/write operations on HDFS distributed file system.

More research has occurred in the multiple-query processing in a distributed environment. Researchers have conducted diversified studies in: (1) system and resources tuning (Herodotou (2011), Herodotou (2012), Marcus (2016)), (2) job scheduling (Zaharia (2009), Isard (2009)) and (3) tasks and/or data sharing (Wang, 2013; Nykiel, 2010; Elgandour, 2012). Those works have pointed out that changing the cluster configuration, ordering the job execution or reusing intermediate results can improve in a significant way MapReduce efficiency during multiple-query or job execution.

The aim of the present work is to improve the processing time of massive online analysis, in distributed data warehouses. As a starting point, we used the assumptions from traditional RDBMS and we assume that inside a workload of queries, a subset of queries may share data. Since MapReduce evaluates all queries on at a time, our first concern was to find all shared data before the query transformation into a MapReduce job; secondly, we needed to materialize those shared data to avoid multiple HDFS scan tasks in order to reduce read/write operations. In our approach, we were inspired by the optimization techniques in RDBMS, such as the Multiple Query Optimization (MQO), the view materialization and the data sharing (Sellis, 1988; Bello et al., 1998; Gupta and Mumick, 1999; Goldstein and Larson, 2001). We performed several query workloads from TPC-DS benchmark to evaluate our approach. Our first experiments have been the subject of a comparison with the standard system provided by Hive and showed promising results regarding storage occupation and query execution time.

Section 2 overviews the data storage, the querying model on Hadoop environment and surveys the works related to our proposal. Section 3 outlines and explains our method for data sharing and materialization. Section 4 describes the implementation of our approach. Section 5 points out...
Mobile Phone Customer Type Discrimination via Stochastic Gradient Boosting
www.igi-global.com/article/mobile-phone-customer-type-discrimination/1783?camid=4v1a

Text Mining to Define a Validated Model of Hospital Rankings
www.igi-global.com/chapter/text-mining-define-validated-model/10186?camid=4v1a