A Context-Based Performance Enhancement Algorithm for Columnar Storage in MapReduce with Hive

Yashvardhan Sharma, Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani, India
Saurabh Verma, Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani, India
Sumit Kumar, Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani, India
Shivam U., Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani, India

ABSTRACT

To achieve high reliability and scalability, most large-scale data warehouse systems have adopted the cluster-based architecture. In this context, MapReduce has emerged as a promising architecture for large scale data warehousing and data analytics on commodity clusters. The MapReduce framework offers several lucrative features such as high fault-tolerance, scalability and use of a variety of hardware from low to high range. But these benefits have resulted in substantial performance compromise. In this paper, we propose the design of a novel cluster-based data warehouse system, Daenyrys for data processing on Hadoop – an open source implementation of the MapReduce framework under the umbrella of Apache. Daenyrys is a data management system which has the capability to take decision about the optimum partitioning scheme for the Hadoop’s distributed file system (DFS). The optimum partitioning scheme improves the performance of the complete framework. The choice of the optimum partitioning is query-context dependent. In Daenyrys, the columns are formed into optimized groups to provide the basis for the partitioning of tables vertically. Daenyrys has an algorithm that monitors the context of current queries and based on the observations, it re-partitions the DFS for better performance and resource utilization. In the proposed system, Hive, a MapReduce-based SQL-like query engine is supported above the DFS.

Keywords: Cluster-Based Architecture, Column Storage, Hadoop’s Distributed File System (HDFS), Hive, MapReduce

DOI: 10.4018/ijcac.2013100104
1. INTRODUCTION

In the era where data is voluminous and continuously being added, processing analytical queries on terabytes of data in a scalable, reliable and efficient way has become one of the most critical issue for data warehousing systems. To deal with data at such levels, special-purpose computations are needed for extracting valuable patterns, related or new trends and contextual knowledge. Given the amount of data, most of these computations need to be executed in parallel distributed fashion. Furthermore, these computations involve complex analysis based on complex data exploration algorithms which require multiple data-sets to be processed simultaneously. (Lähteenmäki & Peitsa, 2012)

Cluster-based data warehouse systems are an implementation of parallel distributed execution for data analysis. The primary concern in cluster-based data warehouse systems is that they work efficiently to speed up the computations rather than collapsing under their own weight. The weight here refers to the extra overheads of distributed data storage techniques and synchronization, fault-tolerance algorithms and network load introduced because of distributed setup of computation nodes. Execution dependencies among queries introduce a strictly sequential ordering of jobs which further increases existing overheads. This may lead to more performance degradation. (Chen et al., 2012; Dittrich et al., 2012)

This paper proposes an intelligent context-based performance enhancement algorithm for columnar storage of data in DFS in MapReduce with Hive (Dittrich et al., 2012; Thusoo et al., 2009). The algorithm monitors the context of executing queries over a period of time and creates a partitioning scheme for efficient retrieval of data from the Hadoop Distributed File System (HDFS) tables. The partitioning scheme consists of columns of tables ranked according to the effectiveness of partitions on them. (Abadi et al., 2008; Shvachko et al., 2010; Stonebraker et al., 2005; Xu et al., 2010)

2. BACKGROUND

To carry out data-intensive analysis in a scalable, fault-tolerant and efficient manner for a distributed environment, Google introduced a distributed and parallel programming framework called MapReduce (Condie et al., 2010; Dean & Ghemawat, 2010). The MapReduce framework is highly desirable as it allows a programmer to specify the analytical job and address the issue of translating the job into sub tasks on multiple machines which are completely automated. Under the umbrella of Apache, an open source implementation of the MapReduce framework, referred to as Hadoop, is freely available to both commercial and academic users. Given its easy access, Hadoop has become a popular choice to process big data produced by the web applications and business industry. Furthermore, due to the success of Hadoop and MapReduce, there is a significant interest in the traditional data warehousing industry to explore the integration of the MapReduce paradigm for large-scale analytical processing of relational data. The two major efforts to provide a declarative interface on top of Hadoop run-time environment are the Pig from Yahoo! and the Hive from Facebook. (Dittrich et al., 2012; Kaldewey et al., 2012; Pavlo et al., 2009; Songting et al., 2010; Thusoo et al., 2009)

Hive is an open-source data warehousing solution built on top of Hadoop. Hive supports queries expressed in a SQL-like declarative language - HiveQL, which are compiled into map-reduce jobs executed on Hadoop. In addition, HiveQL supports custom map-reduce scripts to be plugged into queries. The language includes a type system with support for tables containing primitive types, collections like arrays and maps, and nested compositions of the same. The underlying IO libraries can be extended to query data in custom formats. Hive also includes a system catalog, Hive-Metastore, containing schemas and statistics, which is useful in data exploration and query optimization. In Facebook, the Hive warehouse contains several
From User Requirements to Querying of Fuzzy Summaries
www.igi-global.com/article/from-user-requirements-to-querying-of-fuzzy-summaries/111522?camid=4v1a