International Journal of Data Analytics
Volume 1 • Issue 1 • January-June 2020

A High-Level Interactive Query Language for Big Data Analytics Based on a Functional Model

Symphorien Monsia, LTSIRS, Tunis, Tunisia
Sami Faiz, LTSIRS, Tunis, Tunisia

ABSTRACT

Information technologies such as the internet, and social networks, produce vast amounts of data exponentially (known as Big Data) and use conventional information systems. Big Data is characterized by volume, a high rate of generation, and variety. Systems integration and data querying systems must be adapted to cope with the emergence of Big Data. The authors' interest is with the impact Big Data has on the decision-making environment, most particularly, the data querying phase. Their contribution is the development of a parallel and distributed platform, named high level query language for big data analytics (HLQL-BDA), created to query vast amounts of data in a computer cluster based on the MapReduce paradigm. The query language in HLQL-BDA is implemented by means of interactive query language based on a functional model. The researchers' experiment shows the scalability of HLQL-BDA when they increase the number of nodes and the size of data.

KEYWORDS

Big Data Analytics, Big Data, Functional Model, High-Level Query Languages, Hive, Jaql, MapReduce Paradigm, Parallel and Distributed Platform, Pig, SQL on Hadoop

INTRODUCTION

It is currently the era of massive production of data commonly referred to as 'Big Data'. This is an Anglophone expression used to designate datasets that become so large that they become difficult to work with conventional database management systems. These massive data come from several sources among them the Web, sensor networks, connected peripherals and online social networks (such as Twitter, Facebook, Google+, Foursquare, Linkedin, etc.) that gather billions of users. This infinite growth of data poses very difficult scientific challenges for traditional database techniques: on the one hand big data creates a challenge for their integration within existing information systems and on the other hand big data creates a challenge for the high-level query languages needed to make it possible to query these massive data.

In parallel with the emergence of big data, new paradigms such as cloud computing (Sosinsky, 2010), the MapReduce framework (Dean and Ghemawat, 2004) and non-relational data models (Han et al., 2011) have emerged to try to address the challenges posed by big data. In this article, the authors present a parallel and distributed platform called HLQL-BDA (High Level Query Language for Big Data Analytics), designed to query big data in a cluster of computers based on the MapReduce paradigm. The query language in HLQL-BDA has been implemented by means of an interactive query language based on a functional model (Spyratos, 2006).

DOI: 10.4018/IJDA.2020010102

The rest of the article is organized as follows. In section 2 the researchers present a study of the existing literature describing several of the major massive data query platforms. In section 3 they describe the theoretical research work on which their approach is based. In section 4 the authors present the basic and fundamental principles of their proposed HLQL-BDA approach. In section 5 they present the results of the experiments carried out on their HLQL-BDA platform which enabled them to validate their work. Finally, in section 6 the researchers present their conclusions and next steps to follow.

LITERATURE REVIEW

In this section, the authors present several of the major and most commonly used massive data query platforms as described in the existing literature.

MapReduce Framework

MapReduce (Dean and Ghemawat, 2004) is a parallel programming model initiated in 2004 by Google. According to this programming paradigm, two basic functions must be implemented by the programmer himself. The first is the 'map' function which takes as input a key/value pair and outputs a list of key/value pairs. The second is the 'reduce' function which takes as input a key as well as a set of values relative to this key and produces a list of values as output (see Figure 1). MapReduce can use any distributed data storage system (like GFS i.e. Google File System (Ghemawat et al., 2003)) as its underlying file system for reading inputs as well as writing outputs. The efficiency of this model against big data's processing on a large number of interconnected machines has proved its worth (for example, hundreds of MapReduce programs have been implemented and more than one thousand MapReduce jobs are run daily on Google clusters). However, it has some gaps. One of the major disadvantages of the MapReduce framework is its inability to execute iterative algorithms because it is not designed to handle repetitive tasks (Singh and Reddy, 2015). As a result, a multitude of other systems (i.e. MapReduce extensions) have emerged in recent years, many of which overcome the problem of performing repetitive tasks amongst which the researchers can cite Haloop (Bu et al., 2010), Twister (Ekanayake et al., 2010), iMapReduce (Zhang et al., 2012) and CGL MapReduce (Ekanayake et al., 2008 ; Palit and Reddy, 2011). Another major disadvantage of this framework in which the authors are particularly interested in is its level of abstraction which is too low and therefore very difficult to learn, use, reuse and maintain (Chen et al., 2014; Lee et al., 2012; Olston et al., 2008; Thusoo et al., 2009).

To simplify the development of MapReduce programs to users in an Apache Hadoop environment (one of the open source implementations of Google MapReduce), the main idea in the scientific literature is to increase the abstraction level of the MapReduce framework by constructing high-level query languages (HLQLs) on it to overcome its complexity (Carbone et al., 2015). In the next subsection, the researchers examine three popular implementations of SQL on Apache Hadoop: Pig (Olston et al., 2008), Hive (Thusoo et al., 2009) and Jaql (Beyer et al., 2011). The authors have chosen to study these three HLQLs because they are the most popular and the most used.

High-Level Query Languages (HLQLs)

Pig (Olston et al., 2008) was developed by Yahoo. Its query language is called Pig Latin, a data flow language. Pig can handle complex data structures. Unlike SQL, Pig does not require the data to have a schema and is therefore well suited for processing unstructured data. As traditional SQL, Pig Latin is relationally complete (i.e. it allows to express all calculation queries and relational algebra). The extensibility of Pig Latin is ensured by the UDFs (i.e. User Defined Functions). Hive (Thusoo et al., 2009) was developed by Facebook. Its query language is an SQL dialect called HiveQL, a declarative language. Unlike Pig Latin, HiveQL is not a data flow language but rather allows to express queries to the SQL. Also, unlike Pig, Hive is preferable for processing structured data and a data schema