Chapter 12
Efficient Storage and Parallel Query of Massive XML Data in Hadoop

Wei Yan
Liaoning University, China

ABSTRACT

In order to solve the problem of storage and query for massive XML data, a method of efficient storage and parallel query for a massive volume of XML data with Hadoop is proposed. This method can store massive XML data in Hadoop and the massive XML data is divided into many XML data blocks and loaded on HDFS. The parallel query method of massive XML data is proposed, which uses parallel XPath queries based on multiple predicate selection, and the results of parallel query can satisfy the requirement of query given by the user. In this chapter, the map logic algorithm and the reduce logic algorithm based on parallel XPath queries based using MapReduce programming model are proposed, and the parallel query processing of massive XML data is realized. In addition, the method of MapReduce query optimization based on multiple predicate selection is proposed to reduce the data transfer volume of the system and improve the performance of the system. Finally, the effectiveness of the proposed method is verified by experiment.

INTRODUCTION

In recent years, with the rapid growth of data on the Internet, XML has become the de-facto standard for data representation, data storage and data exchange on the World Wide Web (Bray *et al*. 1997). As a semi-structured data format, XML has many advantages such as simplicity, scalability and cross-platform. Many data have been produced and transformed into XML data format. With the rapid development of data generation and collection technology, the volume of XML data has become enormous and also grows very quickly. Traditional query processing model of XML data is difficult to deal with the problem, which is query processing for massive XML data. In this way, how to store massive XML data effectively and how to query massive XML data has become a hot issue in the current academic community.

DOI: 10.4018/978-1-5225-8446-9.ch012

In the real world, scientific data and log messages are often kept in the form of XML, and the scale of data is becoming very large. For example, UniprotKB provides the data set of global protein resource (Bairoch *et al*. 2005). It is the most comprehensive resource for protein information, providing protein sequences and functional information. For the time being, the size of an XML dataset in the UniprotKB dataset exceeds 260GB, and new XML elements and attributes are continually added to this dataset. Wikipedia provides a knowledge base as the XML data format with a size of over 40 GB. At present, the query processing techniques of massive XML data have attracted attention of the academic community. This chapter adopts the MapReduce programming framework to design the query system of massive XML data on the Hadoop platform, and proposes a parallel query method for processing massive XML data.

Currently, the widespread use of XML data led to an increasing interest in searching and query XML data. XML provides a natural model for tree-structured heterogeneous sources, and that XPath and XQuery are the most commonly used XML query languages (Eisenberg *et al*. 2013). The XPath query language plays an important role in XML query processing: it is widely used in almost every XML technology, starting from query languages such as XQuery and XSLT, to access control languages such as XACML, to JavaScript engine of popular web browsers. The traditional mechanism of XML query processing usually represents user's query request in the form of the predicate in XPath query statement and returns the query result that matches query expression of the predicate exactly. With the continuous growth of XML data volume, the time-consuming and efficiency of storing and querying massive XML data exceeds the ability of traditional XML query processing techniques. How to store and query massive XML data is an important issue in cloud computing environments.

Because XML data has the characteristics of structure and content, the MapReduce programming model is proposed to process the XML file through a pipeline, which is a series of processing steps that receive XML structured data (Zinn *et al*. 2010). The XML data are then updated by the black box function, resulting in the output of the modifying XML structures. Emoto *et al*. (2012) proposes an effective algorithm to deal with the XPath query of tree structure in parallel with the MapReduce programming model. Linear acceleration is realized for tree reduction computations of large-scale XML data, which implements various tree computations such as XPath queries. The above literatures used MapReduce programming model to process massive XML data. However, they did not propose an effective method for partitioning XML data blocks, which were distributed in the XML parallel query system and increased the workload of the system. Under the MapReduce programming model, Bidoit *et al*. (2013) showed a prototype system for querying and updating data on large XML documents, which can statically and dynamically partition the inputted XML documents, so to distribute the computing load among the machines of MapReduce clusters. Choi *et al*. (2012) showed a prototype system HadoopXML, which simultaneously processes many twig pattern queries for a massive volume of XML data in parallel on the Hadoop platform using MapReduce programming model. Specifically, HadoopXML provides an efficient way to process a single large XML file in parallel, and processes multiple twig pattern queries simultaneously with a shared input scan.

However, the existing method of querying massive XML data based on MapReduce programming model does not propose effective storing strategies and clear parallel querying algorithm. For this reason, this chapter proposes a storing and querying method of massive XML data on the Hadoop platform using MapReduce framework. First of all, this chapter proposes the system architecture of processing massive XML data on the Hadoop platform, in which massive XML data are split into data blocks and store in HDFS. The system represents the user's query in the form of the query predicate, which uses the parallel XPath query to process the query predicate. To further send numerous query predicate to