Storage and Query Processing Architectures for RDF Data

Tanvi Chawla
https://orcid.org/0000-0001-9168-1793
Department of Computer Science and Engineering, Malaviya National Institute of Technology Jaipur, India

INTRODUCTION

Resource Description Framework (RDF) also known as the Semantic Web data model represents data in the form of triples (subject, predicate, object). The RDF data can also be represented in form of a graph where subjects and objects depict the vertices of this graph. A RDF triple is formed by an edge connecting these vertices and the predicate (or property) depicting the edge labels (Peng et al., 2016). RDF is a W3C proposed standard that is used to model objects and represent Semantic Web data. Some of the characteristics of RDF data that are similar to Big data are Velocity, Volume, Veracity and Variety (Özsu, 2016). As the domain of RDF data usage is broadening with its use now not just being limited to Semantic Web, it is becoming quite difficult to handle such large scale RDF data. RDF model is used to represent resources on the web and develop detailed descriptions (also called metadata) for these resources. SPARQL is the W3C proposed query language for querying RDF data. The databases specifically used for storing and querying RDF data are known as triplestores or RDF stores. One of the most popular RDF stores is RDF-3x (Neumann & Weikum, 2008). These triplestores unlike relational databases are optimized to store only RDF data and not any other type of data. The SPARQL query language can be used for querying RDF data from these stores (Banane & Belangour, 2019).

Some of the main challenges in large scale RDF management include storage and query processing. The application of existing SPARQL query optimization techniques on large RDF data is also a huge concern. Many SPARQL queries contain complex joins need to be efficiently executed on this huge RDF data. Some of the popular RDF engines focusing on query performance include RDF-3x (Neumann & Weikum, 2008), Virtuoso (Erling & Mikhailov, 2010), Hexastore (Weiss et al., 2008), TripleBit (Yuan et al., 2013) etc. These engines support RDF storage and SPARQL querying (Yuan et al., 2014). These RDF engines are centralized and thus, store RDF data on a single node. These systems are insufficient for large scale RDF storage and handling complex queries on such huge amount of data. As a result distributed RDF management systems came into existing for improving query performance on large RDF data. These systems partition RDF data among nodes in a cluster and execute SPARQL queries on partitioned RDF data in a distributed manner. One of the main issues faced by distributed RDF systems is the cost of partitioning large RDF data (Harbi et al., 2015). Also, these systems face some bottlenecks while loading or query processing such large RDF data (Cheng & Kotoulas, 2015). Virtuoso Cluster Edition (Erling & Mikhailov, 2010), Clustered TDB (Owens et al., 2009), 4store (Harris et al., 2009) are some of the examples of distributed RDF systems that are built on a specialized computer cluster. The disadvantage with these specialized cluster distributed systems is that they require a dedicated infrastructure. But this limitation can be overcome with distributed RDF systems that use cloud-based solutions.

DOI: 10.4018/978-1-7998-9220-5.ch019

Copyright © 2023, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

The cloud-based RDF systems can be used for multiple purposes and not just for RDF data management. Thus, the existing cloud-based clusters installed with Hadoop MapReduce or Spark framework can be directly used for managing huge RDF data (Schätzle et al., 2011). Some of the advantages of using cloud-based solutions for RDF data management are increased availability and cost reduction. Also, the cloud computing infrastructure can easily store such huge RDF data. The underlying Hadoop Distributed File System (HDFS) of Hadoop can be used for RDF storage and MapReduce or Spark framework can be used for processing SPARQL queries over this stored RDF data in HDFS. The advantage with these cloud-based distributed RDF systems is that the data nodes in a cluster can employ any other storage system (such as a triplestore) instead of the basic file system (i.e. HDFS) for RDF storage (Liu, 2010). Some of the examples of cloud-based distributed RDF systems are SHARD (Rohloff & Schantz, 2010), PigSPARQL (Schätzle et al., 2011), S2RDF (Schätzle et al., 2016), SPARQLGX (Graux et al., 2016) etc. One of the cloud-based distributed RDF systems which does not use HDFS for RDF storage but uses MapReduce for query processing is Huang et al. (Huang et al., 2011). A centralized RDF store (or triple store) such as RDF-3x is installed on each data node in the cluster of these systems. In this chapter, the authors discuss both the types of distributed RDF systems (i) which do not use cloud computing technologies (specialized RDF frameworks) and, (ii) which use cloud computing solutions (cloud-based RDF frameworks).

The rest of the chapter proceeds as follows: Section 3 provides an overview of the different RDF frameworks. In Section 3.1, the centralized RDF frameworks are discussed in detail along with their architecture. Section 3.2 describes the different types of distributed RDF frameworks categorized into specialized and cloud-based along with their architecture. The authors propose a scalable cloud-based RDF framework and describe its architecture in Section 4. Finally, some future research directions are examined in Section 5 and this chapter is concluded in Section 6.

BACKGROUND

Resource Description Framework i.e. RDF is generally used for representing and organizing resources in knowledge graphs owing to its flexible nature . The RDF data is a collection of triples i.e. (s,p,o) where predicate p, represents a relationship between the subject s and object o. The RDF dataset consists of multiple RDF statements which can be represented as a directed graph where edges denote the predicates while the vertices denote the subjects and the objects. Some of the formats used for serializing RDF data are N-Triples (nt), Notation-3 (ns3), RDF/XML, Turtle (ttl) etc. Simple Protocol and RDF Query Language i.e. SPARQL) is a query language used in the Semantic Web. This language is used to fetch data from the RDF data model. A SPARQL query may contain triple patterns, optional patterns, conditions, attributes etc. and the results of the query may yield multiple values (Chawla et al., 2016). SPARQL can be used for querying RDF data, RDFS, and OWL ontologies. The different types of SPARQL queries are star, chain, snowflake, complex etc. A Starshaped query consists of subject-subject joins. Chain-shaped (also known as linear) queries consist of subject-object joins. The snowflake-shaped queries are a combination of multiple star shapes. A complex query is a combination of other query shapes. RDF FRAMEWORKSThe increase in amount of RDF data has led to development of many scalable and efficient schemes for large scale RDF data management. The focus has shifted towards developing scalable RDF frameworks to address this challenge. In this section, different types of RDF frameworks are discussed. These frameworks can be classified as- centralized and distributed. A taxonomy of these frameworks is illustrated in Figure 1.Centralized RDF FrameworksIn these systems, the storage and

14 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/storage-and-query-processing-architectures-for-rdf-data/317454

Related Content

Tool Condition Monitoring Using Artificial Neural Network Models
Srinivasa P. Paiand Nagabhushana T. N. (2020). *Handbook of Research on Emerging Trends and Applications of Machine Learning (pp. 550-576).*
www.irma-international.org/chapter/tool-condition-monitoring-using-artificial-neural-network-models/247581

Intelligent System for Credit Risk Management in Financial Institutions
Philip Sarfo-Manu, Gifty Siawand Peter Appiahene (2019). *International Journal of Artificial Intelligence and Machine Learning (pp. 57-67).*
www.irma-international.org/article/intelligent-system-for-credit-risk-management-in-financial-institutions/238128

Crop Prediction for Smart Agriculture Using Ensemble of Classifiers
Khushal Kindraand Bhuvaneswari Amma N. G. (2023). *Machine Learning and Deep Learning for Smart Agriculture and Applications (pp. 124-141).*
www.irma-international.org/chapter/crop-prediction-for-smart-agriculture-using-ensemble-of-classifiers/329893

Role of Technology in Improving the Quality of Financial Advisory for Personal Financial Management
Niranjan Kulkarni, Omvir Gautamand Swapnil Pradeep Shah (2023). *Advanced Machine Learning Algorithms for Complex Financial Applications (pp. 55-80).*
www.irma-international.org/chapter/role-of-technology-in-improving-the-quality-of-financial-advisory-for-personal-financial-management/317017

Early Warning System Framework Proposal, Based on Big Data Environment
Goran Klepac, Robert Kopaland Leo Mrsic (2019). *International Journal of Artificial Intelligence and Machine Learning (pp. 35-66).*
www.irma-international.org/article/early-warning-system-framework-proposal-based-on-big-data-environment/233889