Extended Adaptive Join Operator with Bind-Bloom Join for Federated SPARQL Queries

Damla Oguz, Institute of Research in Computer Science of Toulouse (IRIT), Paul Sabatier University, Toulouse, France & Department of Computer Engineering, Izmir Institute of Technology, Izmir, Turkey & Department of Computer Engineering, Ege University, Izmir, Turkey
Shaoyi Yin, Institute of Research in Computer Science of Toulouse (IRIT), Paul Sabatier University, Toulouse, France
Belgin Ergenç, Department of Computer Engineering, Izmir Institute of Technology, Izmir, Turkey
Abdelkader Hamouurlain, Institute of Research in Computer Science of Toulouse (IRIT), Paul Sabatier University, Toulouse, France
Oguz Dikenelli, Department of Computer Engineering, Ege University, Izmir, Turkey

ABSTRACT

The goal of query optimization in query federation over linked data is to minimize the response time and the completion time. Communication time has the highest impact on them both. Static query optimization can end up with inefficient execution plans due to unpredictable data arrival rates and missing statistics. This study is an extension of adaptive join operator which always begins with symmetric hash join to minimize the response time, and can change the join method to bind join to minimize the completion time. The authors extend adaptive join operator with bind-bloom join to further reduce the communication time and, consequently, to minimize the completion time. They compare the new operator with symmetric hash join, bind join, bind-bloom join, and adaptive join operator with respect to the response time and the completion time. Performance evaluation shows that the extended operator provides optimal response time and further reduces the completion time. Moreover, it has the adaptation ability to different data arrival rates.

KEYWORDS

Adaptive Query Optimization, Bloom Filter, Distributed Query Processing, Join Methods, Linked Data, Query Federation

1. INTRODUCTION

As the increase in the number of data sources on linked data, a distributed data space on the web is generated. This huge global data space can be automatically queried by using two approaches called link traversal (Hartig, Bizer, & Freytag, 2009) and query federation (Görlitz & Staab, 2011a). The first approach is based on discovering potentially relevant data by following the links between them. In other words, it finds the related data sources during the query execution. The second approach, query federation, divides the query into subqueries and distributes them to the SPARQL endpoints of the relevant data sources. The intermediate results from the data sources are aggregated and the final

DOI: 10.4018/IJDWM.2017070103
results are generated. Although both approaches have the advantage of providing up-to-date results, link traversal cannot guarantee finding all results because the relevant data sources change according to the starting point. For this reason, we focus on the query federation approach.

The objective of engines in query federation is to minimize both the response time and the completion time. Response time is the time to generate the first result tuple, whereas completion time is the time to provide all result tuples. Response time and completion time include communication time, I/O time and CPU time. Since the communication time dominates other costs, the main objective of the federated query engines can be stated as to minimize the communication cost. Static query optimization (Selinger, Astrahan, Chamberlin, Lorie, & Price, 1979) is not adequate for federated queries, because they are executed over the SPARQL endpoints of the selected distributed data sources on the web, and the data arrival rates are unexpected. Moreover, most of the statistics about the data sources are missing or unreliable. These constraints show that adaptive query optimization (Deshpande, Ives, & Raman, 2007) is a necessity for query federation over linked data.

Adaptive query optimization has been studied in detail in relational databases (Babu & Bizarro, 2005; Deshpande et al., 2007; Morvan & Hameurlain, 2009; Gounaris, Tsamoura, & Manolopoulos, 2013). However, it is a new research area for linked data. There are only two engines which consider adaptive query optimization for federated queries over SPARQL endpoints: ANAPSID (Acosta, Vidal, Lampo, Castillo, & Ruckhaus, 2011) and ADERIS (Lynden, Kojima, Matono, & Tanimura, 2010, 2011). The first one proposes a non-blocking join method based on symmetric hash join (Wilschut & Apers, 1991) and Xjoin (Urhan & Franklin, 2000), while the second one uses a cost model for dynamically changing the join order. Other than these, AVALANCHE (Basca & Bernstein, 2010, 2014) collects statistical information about relevant data sources and then generates its execution plan to provide the first k tuples. In addition, there are several studies which concentrate on join ordering for SPARQL queries by using different techniques such as evolutionary algorithms (Oren, Guéret, & Schlobach, 2008; Hogenboom, Milea, Frasincar, & Kaymak, 2009) and ant colony (Hogenboom, Frasincar, & Kaymak, 2013; Kalayci, Kalayci, & Birant, 2015). To the best of our knowledge, adaptive join operator (Öguz, Yin, Hameurlain, Ergenc, & Dikenelli, 2016) is the first study which aims to reduce both the response time and the completion time for query federation over SPARQL endpoints.

As mentioned above, the communication cost is the dominant cost in distributed environments. Bloom filter (Bloom, 1970), which is a space efficient data structure, is widely used in relational databases (Mackert & Lohman, 1986; Mullin, 1990; Michael, Nejdl, Papapetrou, & Siberski, 2007; Ives & Taylor, 2008). It is utilized in different linked data tasks such as identity reasoning (Williams, 2008) and data source selection (Hose & Schenkel, 2012). Bloom filter is also employed to reduce the communication cost in two studies of linked data (Basca & Bernstein, 2014; Groppe, Heinrich, & Werner, 2015).

In this paper, we present an extended version of our previous work (Öguz et al., 2016) in which adaptive join operator is proposed. The new contributions of this paper are as follows: i) We improve our previous proposal with bind-bloom join (Basca & Bernstein, 2014; Groppe et al., 2015) for both single join queries and multi-join queries by including bind-bloom join to the candidate join methods. ii) We present a detailed performance evaluation study which shows the advantage of our new proposal. iii) We extend our related work with new studies and comparison of adaptive query optimization methods in query federation. Our operator uses symmetric hash join in the beginning to minimize the response time, and can change the join method to bind join or bind-bloom join. Bind-bloom join, shortly can be defined as a kind of bind join enhanced with bloom filter in order to minimize the communication time. It is explained in detail in the following section. Performance evaluation shows that the extended operator has both the advantage of optimal response time and the adaptation ability to different data arrival rates in order to minimize the completion time. Moreover, it provides faster completion time than our previous operator in all test cases.
Related Content

Multiple Decisional Query Optimization in Big Data Warehouse
[www.igi-global.com/article/multiple-decisional-query-optimization-in-big-data-warehouse/208691?camid=4v1a](www.igi-global.com/article/multiple-decisional-query-optimization-in-big-data-warehouse/208691?camid=4v1a)

iTrade: A Mobile Data-Driven Stock Trading System with Concept Drift Adaptation
[www.igi-global.com/article/itrade/122516?camid=4v1a](www.igi-global.com/article/itrade/122516?camid=4v1a)
Cluster-Based Input Selection for Transparent Fuzzy Modeling
[www.igi-global.com/article/cluster-based-input-selection-transparent/1771?camid=4v1a](www.igi-global.com/article/cluster-based-input-selection-transparent/1771?camid=4v1a)

Big Data Applications in Healthcare
[www.igi-global.com/chapter/big-data-applications-in-healthcare/150214?camid=4v1a](www.igi-global.com/chapter/big-data-applications-in-healthcare/150214?camid=4v1a)