Distributed Query Plan Generation using Cuckoo Search Algorithm

Monika Yadav, School of Computer and Systems Science, Jawaharlal Nehru University, New Delhi, India
T. V. Vijay Kumar, School of Computer and Systems Science, Jawaharlal Nehru University, New Delhi, India

ABSTRACT

Query processing in distributed databases involves data transmission amongst sites capable of providing answers to a distributed query. For this, a distributed query processing strategy, which generates efficient query processing plans for a given distributed query, needs to be devised. Since in distributed databases, the data is fragmented and replicated at multiple sites, the number of query plans increases exponentially with increase in the number of sites capable of providing answers to a distributed query. As a result, generating efficient query processing plans, from amongst all possible query plans, becomes a complex problem. This distributed query plan generation (DQPG) problem has been addressed using the Cuckoo Search Algorithm (CSA) in this paper. Accordingly, a CSA based DQPG algorithm (DQPG\textsubscript{CSA}) that aims to generate Top-K query plans having minimum cost of processing a distributed query has been proposed. Experimental based comparison of DQPG\textsubscript{CSA} with the existing GA based DQPG algorithm shows that the former is able to generate Top-K query plans that have a comparatively lower query processing cost. This, in turn, reduces the query response time resulting in efficient decision making.

KEYWORDS

Algorithms, Cuckoo Search Algorithm (CSA), Distributed Query Plan Generation (DQPG), Distributed Query Processing, Swarm Intelligence

1. INTRODUCTION

Data is the most essential part of every application in computer science (Haigh, 2006). All the processing and computational tasks are done over data. The large amount of data so generated, needs to be transformed into information. This would require data to be stored in such a manner whereupon it can be extracted and used later. Initially, database management systems provided centralized control and were used to cater to the various requests from users at different sites. These systems though simple, as data was stored and handled at a central site, suffered from limitation like central point failure. Also, a huge amount of communication cost was incurred while accessing the central site from the remote sites, from where queries were posed. These limitations were overcome by distributing the data across multiple sites. This resulted in the concept of distributed databases. A distributed database is a collection of various databases distributed over different sites connected through a communication network (Ceri and Pelagatti, 1984; Ozsu and Valduriez, 2011). The system, which makes this distribution transparent to the user, is referred to as a distributed database management systems (DDBMS) (Ceri and Pelagatti, 1984; Ozsu and Valduriez, 2011, Hoffer et al., 2009). DDBMS is required to provide a single unified view of the data distributed across multiple disparate data sources in order to simplify the access to this data by the users. This posed challenges
for data integration, which requires the combining of data from different sources. As data was replicated in different data sources, complex customized programs were required to be designed for data retrieval and manipulation. However, while considering the computational power of a system, the size of data becomes important. Bigger the size of data, greater would be the time and space complexities of the program managing it. Hence, the access mechanism became a bottleneck for such systems. Query processing is an important factor determining the performance of DDBMS (Yu and Chang, 1984). Processing a query in a distributed environment is different from processing it in a centralized system. Various issues need to be addressed, as the relations required to process a query are distributed across multiple disparate sites. Query processing in distributed databases involves data transmission amongst sites that can provide answers to a distributed query. For this, a distributed query processing (DQP) strategy needs to be devised that generates efficient query processing plans for a given distributed query. The main objective of a DQP is to transform a high level distributed query, which is generally a relational calculus query, into a low level query, which is generally a relational algebra query, along with the communication operators. This transformation must take care of the correctness (in terms of semantics) and efficiency (in terms of the time and space complexity). As there are multiple copies of relations spread across different sites, there can be many possible transformations, generating semantically equivalent low level queries for a high level query. So, the main objective is to select a query that require minimal resource consumption for data communication and local processing. Thus, resource consumption becomes a measure to determine the total cost to process a query in distributed systems (Kossmann, 2000, Yu and Chang, 1984). The two important resources, the local processing cost and the communication cost, must be wisely used to reduce the total cost of processing a distributed query. Local processing cost includes the CPU cost and the I/O cost. Communication cost depends on the communication network and the amount of data to be transferred between sites. Since the communication cost is the dominant cost, a DQP strategy primarily needs to generate query plans that reduce this cost (Ozsu and Valduriez, 2011). Further in distributed databases, the data is fragmented and replicated at multiple sites and the number of query plans increases exponentially with an increase in the number of sites that can provide the answer to a distributed query. As a result, generating an efficient query processing plan, from amongst all possible query plans, becomes a complex task. This problem of generating efficient distributed query plans, referred to as the Distributed Query Plan Generation (DQPG) problem (Vijay Kumar et al., 2010; 2011), has been addressed in this paper. The DQPG problem is discussed next.

1.1. The DQPG Problem

The DQPG problem given in (Vijay Kumar et al., 2010; 2011) focuses on generating ‘close’ query plans for a distributed query. The degree of closeness depends on the number of sites required for processing the query along with the higher concentration of relations in these participating sites. Query plans involving fewer sites and having higher concentrations of relations within the participating sites are considered ‘close’ and are preferred over other query plans, as they incur lesser communication costs, which happens to be the major cost in processing a distributed query. As an example, consider the distributed SQL query given in Figure 1. The relation–site matrix, for the six relations accessed by this distributed query, is shown in Figure 2.

A ‘1’ in the Relation–Site matrix signifies the presence of a relation at the corresponding site and its absence is denoted by ‘0’. Some of the possible legal query plans are shown in Figure 3.

Figure 1. Distributed SQL query

Select A1, A2, A3, A4, A5
From R1, R2, R3, R4, R5, R6
Energy Management in Cloud Computing
www.igi-global.com/chapter/energy-management-in-cloud-computing/146742?camid=4v1a