Distributed Query Plan Generation using Particle Swarm Optimization

T.V. Vijay Kumar, School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India
Amit Kumar, School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India
Rahul Singh, School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India

ABSTRACT

A large number of queries are posed on databases spread across the globe. In order to process these queries efficiently, optimal query processing strategies that generate efficient query processing plans are being devised. In distributed relational database systems, due to replication of relations at multiple sites, the relations required to answer a query may necessitate accessing of data from multiple sites. This leads to an exponential increase in the number of possible alternative query plans for processing a query. Though it is not computationally feasible to explore all possible query plans in such a large search space, the query plan that provides the most cost-effective option for query processing is considered necessary and should be generated for a given query. In this paper, an attempt has been made to generate such optimal query plans using Set based Comprehensive Learning Particle Swarm Optimization (S-CLPSO). Experimental comparisons of this algorithm with the GA based distributed query plan generation algorithm shows that for higher number of relations, the S-CLPSO based algorithm is able to generate comparatively better quality Top-K query plans.

Keywords: Distributed Databases, Particle Swarm Optimization, Query Plans, Query Processing, Set Based Comprehensive Learning Particle Swarm Optimization (S-CLPSO)

1. INTRODUCTION

A distributed database encompasses coherent data, spread across various sites of a computer network (Ceri & Pelagatti, 1984). A Distributed Database Management System (DDBMS) deals with managing such distributed databases. DDBMS presents a simple and unified interface to users by providing them with access to the disparate databases, as if they were not distributed (Ozsu & Valduriez, 1991). The performance of a DDBMS is determined by its ability to process queries in an effective and efficient manner (Rho & March, 1997). The query

DOI: 10.4018/ijsir.2013070104
The query processing problem is much more complicated in distributed environments, as there are various parameters affecting their performance (Alom et al., 2009). The required information for processing a distributed query is usually available at different sites. The query processing, thus, would involve transmission of data between these sites. These data transmissions, along with local data processing, constitute a distributed query processing (DQP) strategy for a query.

In distributed query processing (Liu & Yu, 1993), the distributed query is parsed before arriving at an effective query processing strategy for it. This strategy comprises of effective and efficient query processing plans that would decompose the distributed queries into local sub-queries to be executed at their respective sites. Also, the order and the site at which the results of the sub-queries are integrated is also part of this plan. The final integrated result is provided as the answer to the distributed query. Thus the DQP strategy aims to generate query processing plans that reduce the amount of data transfer between sites and thereby reduces the distributed query response time (Ozsu & Valduriez, 1991; Bodorik & Riordon, 1988; Liu et al., 1996). This paper focuses on generating query processing plans for distributed relational queries.

The major costs incurred in DQP are CPU, I/O and the site-to-site communication cost. Among these, the site-to-site communication cost is the dominant cost. This cost can be reduced if fewer sites are involved in processing a distributed query. In order to process a distributed query, the data required may have to be obtained from several sites distributed over a computer network. Furthermore, as the number of sites containing the relations accessed by the query increase, the number of possible valid query plans also increases. So it becomes imperative to arrive at a query processing plan that entails an optimal cost for query processing. However, the number of such possible query plans increases exponentially with increase in the number of relations in the query and also with increase in the number of sites containing them (Ioannidis & Kang, 1990). Thus, a large search space comprising all possible query plans needs to be explored in order to compute the optimal query plans. This exhaustive search is not computationally feasible (Ioannidis & Kang, 1990). Further, this being a combinatorial optimization problem (Jarke & Koch, 1984), it can be addressed by techniques based on heuristics like greedy, evolutionary, and randomized (Ioannidis & Kang, 1990; Bennett et al., 1991; Dong & Liang, 2007; Gregory, 1998; Ioannidis, 1987; Stillger & Spiliopoulou, 1996; Swami & Gupta, 1998). However, efficiency of these techniques is affected by the unconventional behavior, in specific instances, of the problem (Ioannidis & Kang, 1991). In Vijay Kumar et al. (2011), an approach that generates “close” query plans with respect to the number of sites involved and the concentration of relations in the sites for a distributed relational query is given. As per (Vijay Kumar et al., 2011), query processing over lesser number of sites would be more efficient and thus query plans involving fewer sites need to be generated. Such query plans, referred to as “close” query plans, are generated using the genetic algorithm (GA) in Vijay Kumar et al. (2011).

In Vijay Kumar et al. (2011), two heuristics were used to generate query plans for a given query. The first heuristic is based on the number of sites required for processing a user query. As per this heuristic, lesser number of sites would entail lower site-to-site communication costs, which happens to be the major cost incurred in distributed query processing. Thus, query plans involving fewer sites need to be generated before the ones involving higher number of sites. The second heuristic is concerned with the concentration of relations accessed by the query in the individual participating sites. As per this heuristic, query plans having sites with higher concentration of relations are more desirable, as the costly join operation would be performed between relations at a single site. These two heuristics are used to define the following cost function, referred to as the Query Proximity Cost \( QPC \), which computes the cost of proximity of data relevant for answering a user query:
Swarm-Inspired Routing Algorithms for Unstructured P2P Networks
[www.igi-global.com/article/swarm-inspired-routing-algorithms-for-unstructured-p2p-networks/210069?camid=4v1a](www.igi-global.com/article/swarm-inspired-routing-algorithms-for-unstructured-p2p-networks/210069?camid=4v1a)