Distributed Query Plan Generation using Ant Colony Optimization

T.V. Vijay 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

Amit Kumar, School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India

ABSTRACT

Query processing is a critical performance evaluation parameter and has received a considerable amount of attention especially in the context of distributed database systems. The aim of distributed query processing is to effectively and efficiently process the query. This entails laying down an optimal distributed query processing strategy that generates efficient query plans. Since in distributed database systems, the data is distributed and replicated at multiple sites, the number of query plans increases exponentially with increase in the number of relations accessed by the query along with increase in the number of sites containing these relations. Thus, from amongst these query plans, there is a need to generate optimal query plans involving lesser number of sites which, in turn, would entail lower site-to-site communication cost leading to faster query response times. In this paper, an attempt has been made to generate such query plans for a distributed query using Ant Colony Optimization (ACO). This ACO based distributed query plan generation (DQPG) algorithm, when compared with the GA based DQPG algorithm, is able to generate comparatively better quality Top-K query plans for a given distributed query.

Keywords: Ant Colony Optimization, Distributed Databases, Query Plans, Query Processing, Swarm Intelligence

1. INTRODUCTION

A distributed database is defined as a collection of data belonging to logically interrelated databases spread over different sites of a computer network (Hakimzadeh, 2005; Özsu & Valduriez, 1991; Özsu & Valduriez, 2004). A distributed database management system is defined as a software system that facilitates the management of such distributed databases with the aim of providing transparency in such distribution to the users (Özsu & Valduriez, 2004). It provides

DOI: 10.4018/ijamc.2015010101
high level support for developing complex applications. Unlike centralized database systems, where the only resource that needs to be shielded from the user is the data, in distributed database management systems the communication network also needs to be managed. The user is independent of the network operational details. This kind of transparency is referred to as network transparency or distribution transparency. This distribution transparency enables the users to pose queries without having knowledge of the location of the data. Another important issue of distributed databases is the replication of data across the database nodes in the network (Özsu & Valduriez, 1997). Data is replicated in order to improve the performance, reliability and availability of the system. The data residing at a particular site is also stored at a site where it is more frequently accessed. As a result, it would increase the locality of reference. This replication should be transparent such that it appears, to a user, that there is a single copy of data, though in reality multiple copies of the same are distributed in database nodes spread across the network. Data fragmentation is also desirable, where data is divided into fragments and each such fragment is stored at different database nodes, in the network (Ceri & Pelagatti, 1985; Özsu & Valduriez, 2004; Özsu & Valduriez, 1997). Fragmentation increases the performance, availability and reliability of the system.

The aim of distributed query processing is to provide answers to user queries in an effective and efficient manner. In distributed databases, queries are usually non-procedural in nature, where the user specifies what is required without specifying how the answer to it should be retrieved. This procedure is actually devised by the query processor in a distributed database management system (Özsu & Valduriez, 2004) and thus relieves the user from tediously processing the query. Query processing is a critical performance issue and has received considerable amount of attention in the context of both centralized as well as distributed database systems (Ceri & Pelagatti, 1985; Kossmann, 2000; Özsu & Valduriez, 2004). It becomes more complex and performance critical in the case of distributed database systems, as a large number of issues like data fragmentation, data replication and distantly located data have an impact on query processing. Distributed query may involve relations, which have been fragmented and/or replicated, leading to inclusion of costs due to communication overheads. If relations at many sites are required for answering the user query, query processing may be time consuming, i.e. query response time would be high due to communication between the involved sites.

Several distributed query processing strategies exist that have attempted to reduce this query processing time by devising query plans to minimize resource consumption. A good measure of assessing resource consumption is the total cost that will be incurred in processing the query. Another good measure is the response time (Epstein et al., 1978) of the query, which is the time elapsed for executing the query. In a distributed system, the minimization of the total cost includes the CPU, I/O and communication costs (Özsu & Valduriez, 1991). The CPU cost is incurred when performing operations on data in the main memory. The I/O cost is the time required for disk accesses. The communication cost is the time needed for exchanging data between sites participating in the execution of the query (Özsu & Valduriez, 2004). The task of reducing this query processing cost, involving the site-to-site communication costs, lies with the query optimizer. A query optimizer is a software module that comprises of three components namely a search space, a cost model, and a search strategy (Özsu & Valduriez, 2004; Úzsu & Valduriez, 1991). In this paper, the distributed query plan generation (DQPG) problem, given in Vijay Kumar et al. (2011), has been addressed. This DQPG problem is discussed next.

### 1.1. The DQPG Problem

In a distributed environment, query processing can become more efficient when fewer numbers of sites are involved. It implies that
Related Content

A Meta-Heuristic Model for Data Classification Using Target Optimization
[www.irma-international.org/article/a-meta-heuristic-model-for-data-classification-using-target-optimization/182235](http://www.irma-international.org/article/a-meta-heuristic-model-for-data-classification-using-target-optimization/182235)

Using Evolutionary Based Approaches to Estimate Software Development Effort
[www.irma-international.org/chapter/using-evolutionary-based-approaches-estimate/44368](http://www.irma-international.org/chapter/using-evolutionary-based-approaches-estimate/44368)

An Industry Internet of Things Framework for Epilepsy Detection, Monitoring, and Control

Solving Vehicle Routing Problem With Multi-Phases Simulated Annealing Algorithm
Modified Sub-Population Based Heat Transfer Search Algorithm for Structural Optimization
www.irma-international.org/article/modified-sub-population-based-heat-transfer-search-algorithm-for-structural-optimization/182234