STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 1, 2005

REDESIGN BASED OPTIMIZATION FOR DISTRIBUTED DATABASES

HOREA-ADRIAN GREBLA, ANCA GOG

Abstract. The execution process of the queries in distributed databases require accurate estimations and predictions for performance characteristics. The problems of data allocation and query optimization done by means of mobile agents and evolutionary algorithms are considered. These problems still present a challenge because of the dynamic changes in number of components and architectural complexity of nowadays system topologies. The distributed system is modeled as a graph structure on which is defined a dynamic cost vector. The cost vector remains consistent, relevant, by use of mobile agents performing cost statistics and vector updates. An evolutionary technique for the re-design phase is proposed. Experimental results prove the efficiency of the proposed technique.

Keywords: Distributed Databases, Data Fragmentation, Data Allocation, Evolutionary Computation, Mobile agents

1. Introduction

Distributed databases (DDBs) have become necessity as networks expand and organizations perform geographically distributed operations. International companies store their data at different sites of a computer network, possibly in a variety of forms, ranging from flat files, to hierarchical, relational or object-oriented databases. The network itself consists of variety of transmission media, network topologies or network speeds. Design approaches for distributed databases have to consider various factors that can affect performance: CPU time, data transmission time, disk I/O operation time. Such distributed system architecture reveals some data management challenges. The system needs to be highly scalable with no critical failure points. In accordance to nowadays computing needs, the latency must not affect the performance of real-time applications. The aim is to provide uniform access to physically distributed data, no mater what the distance between the access location and places data resides. A possible approach is to represent the DDB as a graph and to perform system's management automatically by means

2000 *Mathematics Subject Classification.* 62E99, 68T99.

1998 *CR Categories and Descriptors.* C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks – *Distributed systems* I.2.8 [**Artificial Intelligence**]: Problem solving, Control methods, and Search – *Heuristic methods*;

of mobile agents. An evolutionary algorithm is proposed to solve the problem of re-fragmentation and re-allocation of data.

2. Distributed Database Design Issues

Distributed database management system [8] has to ensure local applications for each computational component as well as global applications on more computational machines; it also has to provide a high-level query language with distributed query power, for distributed applications development. Must be ensured transparency levels that confer the image of a unique database. To improve the performance of global queries, data can be partitioned and spread over the system's components. A distributed database system supports data fragmentation if a relation stored within can be divided in pieces called fragments. These fragments can be stored on different sites residing on the same or different machines. The aim is to store the fragments closer to where they are more frequently used in order to achieve best performance. The partitions can be created horizontal, vertical or mixed (the combination of horizontal and vertical fragmentation).

Let $R[A_1,A_2,\ldots,A_n]$ be a relation where A_i, $i = 1,\ldots,n$ are attributes. A horizontal fragment can be obtained by applying a restriction: $R_i = \sigma_{cond_i}(R)$, where $cond_i$ is the guard condition. So we can rebuild the original relation by union as follows:

$$R = R_1 \cup R_2 \cup \ldots \cup R_k.$$

A vertical fragment is obtained by a projection operation:

$$R_i = \prod_{\{A_{x1},A_{x2},\ldots,A_{xp}\}} (R),$$

where A_{xi}, $i = 1,\ldots,p$ are attributes. The initial relation can be reconstructed by join of the fragments:

$$R = R_1 \bigotimes R_2 \bigotimes \ldots \bigotimes R_l.$$

A DDB system can be represented [5] as a graph where the sites are given by (V), the set of vertices, and the edges (E) given by the direct connections between sites. Each edge has associated a cost, but this cost will be examined later in this paper. For exemplification we consider in Figure 1 a distributed system and obtain in Figure 2 the corresponding graph representation.

The system must preserve distributed data independence [9], such that any change of physical location of data must not disturb application functionality. A good management of DDBs implies a considerable effort in the design phase of the system and also implies a redesign phase for performance tuning. One of the design phase component that raise problems represent data fragmentation and allocation. The biggest improvement in system's response can be achieved by fragmentation and reallocation in the design refinement phase. The use of mobile agents can bring great performance value to the system because a software agent [10] can act autonomously on behalf of the administrator. The elements of the system do not

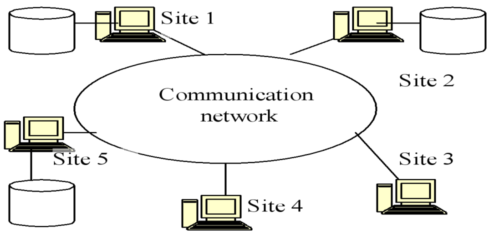

FIGURE 1. Distributed database system

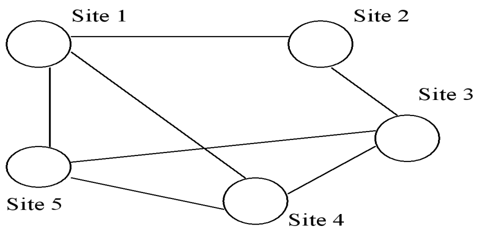

FIGURE 2. Corresponding graph of the system from Figure 1

have to be connected all the time, agents can travel in the network and execute at different hosts by taking their state and implementation with them. Agents can be intelligent, take decisions and react to environment changes to perform their actions, and most important, they can cooperate to fulfill their common goal.

3. Overview and Architecture of the System

In what follows, a distributed database system architecture where design relies on the graph representation and system management improvement by use of agents is proposed. An agent based architecture with distributed access and concurrent queries in heterogeneous database system is described. The considered architecture provides high scalability and performance optimization. The main improvement is the manner of cost definition between sites:
• First, we define the initial cost assigned by the system designer to an edge; this cost is estimated based on network transfer rate, data access time and computing power on a site. We call this ***initial estimated cost***.
• At some given times we can obtain more accurate cost in the system; we define this cost the ***up-to-date computed cost***.

Translator agents perform the translation of local names to global names and provide a common language for distributed queries assuring local database management system independence.

Retriever agents collect data from corresponding fragments by communicating with Translators. In fact they build the query in the agent common language and ask the translators for results.

Optimizer can be unique for the entire system or can be cloned; it contains the query optimizer. One role of the *Optimizer* is to build *up-to-date computed cost* from statistics for the sites with respect to the amount of data accessed on that site. The proposed architecture is depicted in Figure 3.

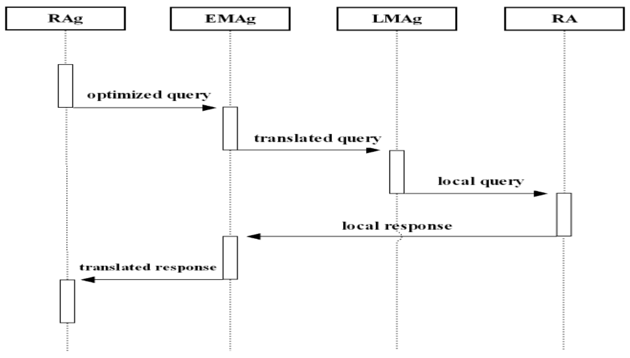

FIGURE 3. Proposed architecture

4. EVOLUTIONARY FRAGMENTATION AND ALLOCATION ALGORITHM IN DISTRIBUTED DATABASES

The problem of database fragmentation and data allocation is modeled as a graph. We have to distribute m tuples to n nodes of the graph. The costs of the edges between the vertexes of the graph are given. Also, statistics referring to the frequency of the requested tuples in the graph are given (computed by agents). The tuples' distribution can be reduced to an optimization problem which goal is to minimize the costs generated by the queries in the graph. An evolutionary algorithm is proposed to solve this NP-Complete problem [6]. The proposed algorithm is called Evolutionary Fragmentation and Allocation Algorithm in Distributed Databases (EFA algorithm).

A fixed size population is used in the proposed algorithm. The m tuples that have to be distributed to nodes will be denoted by $t_1, t_2, \ldots, t_m$. There are no restrictions regarding the minimum or the maximum number of tuples contained by a node. A potential solution of the problem (a chromosome) is a string of constant length $\{x_1, x_2, \ldots, x_m\}$, where the gene x_i, $x_i \in \{1, 2, \ldots, n\}$, indicates to which node the tuple i belongs.

The potential solutions are evaluated by means of a real-valued fitness function $F, F \colon X \to \mathbb{R}$, where X denotes the space of potential solutions. The fitness of a chromosome takes into account the costs of the edges between nodes and the statistics regarding the frequency of the requested tuples in the graph:

$$F(x) = \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij} c_{iN_j},$$

where f_{ij}, $i \in \{1, 2, \ldots, n\}$, $j \in \{1, 2, \ldots, m\}$, represents the frequency of the requests of the tuple j from the node i of the graph. Also, c_{iN_j}, $i \in \{1, 2, \ldots, n\}$, $j \in \{1, 2, \ldots, m\}$, represents the cost of the edges between the node i and the node that contains the tuple j, denoted by N_j. The fitness function F is to be minimized.

Rank-based selection for recombination mechanism [4], *two points crossover* and weak mutation operator [1] are considered for the proposed algorithm [3]. The best from parent and offspring enters the new generation [2].

The algorithm ends after a certain number of generations that did not improve the best solution of the generation [7]. The best solution obtained during the search process is considered to be the solution of the problem.

5. Experimental results

A graph having five nodes is considered ($n = 5$). Let us denote the five nodes by N_1, N_2, N_3, N_4, N_5. The associated costs for the edges between the given nodes are depicted in Figure 4. *Remark:* We are interested only in the direct cost

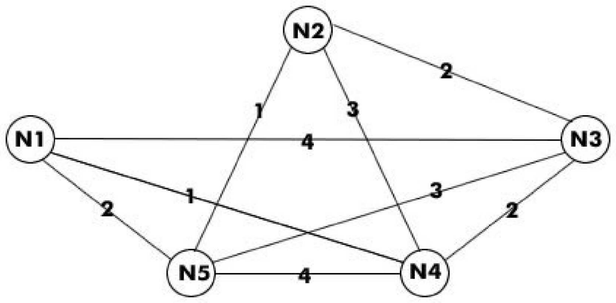

FIGURE 4. Proposed architecture

between two nodes, and that is why there are nodes without edges between them, even if there could be a path between the two nodes by using intermediate nodes.

A dataset of 1.200.000 tuples is given. The given tuples are denoted by $t_1, t_2, \ldots$, t_m, where m represents the number of tuples. The existing dataset fragmentation and distribution of tuples in nodes are depicted in the Table 1.

Table 1. Dataset fragmentation and distribution of tuples in nodes.

Node	**Dataset fragmentation**	**Number of tuples/node**
N_1	$t_1 - t_{100.000}$	100.000
N_2	$t_{100.001} - t_{380.000}$	280.000
N_3	$t_{380.001} - t_{540.000}$	160.000
N_4	$t_{540.001} - t_{720.000}$	180.000
N_5	$t_{720.001} - t_{1.200.000}$	480.000

The statistics regarding the frequency of requests of the tuples from each node are depicted in Tables 2 - 6.

Table 2. The frequency of requests of the tuples from the node N_1

Tuples	**Frequency**
$t_{40.001} - t_{90.000}$	4
$t_{420.001} - t_{560.000}$	10
$t_{610.001} - t_{730.000}$	12
$t_{980.001} - t_{1.100.000}$	5

Table 3. The frequency of requests of the tuples from the node N_2

Tuples	**Frequency**
$t_{250.001} - t_{330.000}$	2
$t_{560.001} - t_{680.000}$	14
$t_{1.100.001} - t_{1.200.000}$	7

Table 4. The frequency of requests of the tuples from the node N_3

Tuples	**Frequency**
$t_1 - t_{100.000}$	3
$t_{250.001} - t_{290.000}$	10
$t_{880.001} - t_{970.000}$	9
$t_{990.001} - t_{1.000.000}$	16

The tuples that do not appear in the tables containing the frequency of requests are never requested. They will remain inside the nodes that contain them before applying the EFA algorithm. The proposed EFA algorithm was applied for data described above. The chosen values for the algorithm parameters are written in Table 7.

TABLE 5. The frequency of requests of the tuples from the node N_4

Tuples	**Frequency**
$t_{100.001} - t_{170.000}$	3
$t_{220.001} - t_{330.000}$	7
$t_{450.001} - t_{560.000}$	13
$t_{680.001} - t_{770.000}$	8

TABLE 6. The frequency of requests of the tuples from the node N_5

Tuples	**Frequency**
$t_{200.001} - t_{260.000}$	12
$t_{700.001} - t_{830.000}$	6
$t_{920.001} - t_{980.000}$	1

TABLE 7. The EFA algorithm parameters

Population size	**Number of generations that did not improve the current solution**	**Probability of recombination**	**Probability of mutation**
200	50	0.7	0.1

After applying EFA algorithm, the way the tuples are redistributed to the nodes of the graph, by taking into account the frequency of the requests of the tuples, is described in Table 8.

6. Conclusions and future work

An evolutionary algorithm called EFA was proposed for the redesign phase, meaning re-fragmentation and re-allocation, in our distributed system. The considered problem is a NP-Complete one. EFA was successfully applied and experimental results have proved the efficiency of the proposed algorithm.

As future work, the method can be improved by computing the costs weighted with factors like local interest for fragments (recommend replication or not), real-time response importance (some applications do not need real-time response), data access frequency (balance sheet data may be consulted once in a month).

The weight of the factors in the cost computation can be changed in time, also changes in network topology or transmission media can influence the response time. The statistics are useful for rebalancing the system by re-computing the costs to obtain best response time for all queries on any site.

TABLE 8. Reallocation of tuples in nodes after applying EFA.

Node	**Dataset refragmentation**	**Number of tuples/node**
N_1	$t_{680.001} - t_{770.000}$, $t_{1.000.001} - t_{1.100.000}$	190.000
N_2	$t_{170.001} - t_{200.000}$, $t_{250.001} - t_{260.000}$ $t_{330.001} - t_{380.000}$, $t_{560.001} - t_{610.000}$ $t_{770.001} - t_{830.000}$, $t_{970.001} - t_{980.000}$ $t_{1.100.001} - t_{1.200.000}$	310.000
N_3	$t_1 - t_{40.000}$, $t_{90.000} - t_{100.000}$ $t_{260.001} - t_{330.000}$, $t_{380.001} - t_{420.000}$ $t_{880.001} - t_{970.000}$, $t_{990.001} - t_{1.000.000}$	260.000
N_4	$t_{40.001} - t_{90.000}$, $t_{100.001} - t_{170.000}$ $t_{420.001} - t_{560.000}$, $t_{980.001} - t_{990.000}$	270.000
N_5	$t_{200.001} - t_{250.000}$, $t_{610.001} - t_{680.000}$ $t_{830.001} - t_{880.000}$	170.000

REFERENCES

[1] Bäck, T., Fogel, D.B., Michalewicz, Z. (Editors), *Handbook of Evolutionary Computation*, Institute of Physics Publishing, Bristol and Oxford University Press, New York, 1997.

[2] Bäck, T., *Optimal mutation rates in genetic search*, Proceedings of the 5th International Conference On Genetic Algorithms, Ed. S. Forrest, Morgan Kaufmann, San Mateo, CA, 2-8, 1993.

[3] Dumitrescu, D., Lazzerini, B., Jain, L.C, Dumitrescu, A., *Evolutionary Computation*, CRC Press, Boca Raton, FL., 2000.

[4] Goldberg, D.E., Deb, K., *A comparative analysis of selection schemes used in genetic algorithms*, Foundations of Genetic Algorithms G.J.E. Rawlins (Ed.), Morgan Kaufmann, San Mateo, CA, 69-93, 1991.

[5] Moldovan, G., *Reorganization of a Distributed Database*, Babes-Bolyai University, Seminar of Models, Structures and Information Processing, Preprint nr. 5, p. 3-10, 1984.

[6] Levin, K. D., Morgan, H. L., *Optimizing distributed databases-A framework for research*, Proceedings of AFZPS NCC, vol. 44. AFIPS Press, pp. 473-478, 1975.

[7] Mitchell, M., *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.

[8] Oszu, M. T., Valduriez, P., *Principles of Distributed Database Systems*, Prentice Hall, Englewood Cliffs, NJ, 1999.

[9] Piattini, M. and Diaz, O., *Advanced Database Technology and Design*, Artech House, Inc. 685 Canton Street Norwood, MA 02062, 2000.

[10] Weiss, G., *Multiagent System, A Modern Approach to Distributed Artificial Intelligence*, MIT Press , USA, 2000.

BABES-BOLYAI UNIVERSITY OF CLUJ-NAPOCA, FACULTY OF MATHEMATICS AND COMPUTER SCIENCE, COMPUTER SCIENCE DEPARTMENT

E-mail address: `horea,anca@cs.ubbcluj.ro`