DATA ALLOCATION IN DISTRIBUTED DATABASE SYSTEMS PERFORMED BY MOBILE INTELLIGENT AGENTS

by
Horea Grebla, Grigor Moldovan,
Sergiu Adrian Darabant, Alina Câmpan

Abstract: As the European Union extends its boundaries the major companies have extended their presence on different markets resulting sales expansion and marketing specialization. Moreover, globalization brings a bigger impact on vital business's data because of the applications that have been developed on platforms having specific aspects by means of database technology and infrastructure. In fact, even if you are a big company, it is not always profitable to bring an expensive solution to a new member company in a pour country, but to develop and extend the existing solution at smaller rates, but with less computational flexibility.
As a result, the companies have to implement the same or more applications on different database systems, but have a global view on the entire system. A major cost in executing queries in a distributed database system is the data transfer cost. This cost is a result of transferring multiple database segments (fragments) accessed by a query to the original place of the query.
Due to the complexity of the modern database management systems the execution process of the queries implies accurate estimations and predictions for performance characteristics. This seems to be very hard at early design stages.
In this paper we propose a solution for data allocation and optimization in distributed database systems done by means of mobile agents having attached learning capacities completed by collaboration and coordination functionalities.
To exemplify our model we construct a fragment dependency graph model on which we express the dependencies among the fragments accessed within a distributed query. This model is than used to formulate data allocation problems and to propose a solution implying mobile agents and agent learning and cooperation.
As conclusions we underline some of the advantages our solution can bring to this research field and the future work we intend to perform.
Keywords: distributed database system, data allocation, distributed database design, distributed query processing, graph model, mobile agents, agent learning, agents cooperation.

1. Introduction

A distributed database [4] is a Data Collection which satisfies the following assumptions: resides on more than one machine with computational power; machines are connected by a communication network; it benefits of a

distributed database management system which allows users to feel they work on the entire database and gives users the opportunity to declare what they want not how they want. The practical experience has demonstrated that there are powerful reasons for a distributed system to be feasible it has to be relational. So we suppose we deal only with this type of DBMSs.

Distributed database management system [4] has to ensure local applications for each computational station as well as global applications on more computational machines; to develop applications it has to provide a high level query language with distributed query building means. Transparency levels must confer the image of a unique database.

A distributed database system supports data fragmentation if a relation stored within can be divided in pieces called fragments for physical storage purposes. Data fragments can be stored at different sites, on the same or different machines, where they are more frequently used thus having a lower network traffic and increased performance. Data fragmentation can be done horizontal and vertical.

Let $\Re[A_1, A_2, ..., A_n]$ be a relation where $A_i, i = \overline{1,n}$ are attributes.

A horizontal fragment can be obtained by applying a restriction: $\Re_i = \sigma_{cond_i}(\Re)$, where $cond_i$ is the guard condition. So we can rebuild the original relation by union as follows: $\Re = \Re_1 \cup \Re_2 \cup ... \cup \Re_k$.

A vertical fragment is obtained by a projection operation: $\Re_i = \prod_{\{A_{x1}, A_{x2}, ..., A_{xp}\}}(\Re)$ where $Ax_i, i = \overline{1,p}$ are attributes. The initial relation can be reconstructed by join of the fragments: $\Re = \Re_1 \otimes \Re_2 \otimes ... \otimes \Re_l$

1.1 The data model

In the following we present our data model [5] for better understanding the theory by a practical example:

Customer(Code, City)- 10.000 tuples on computer C1
Product(Code, Color) – 100.000 tuples on computer C2
Invoice(Customer,Product,Quantity) - 1.000.000 tuples on computer C1
Let the query be:
CUSTOMER Code FROM City Ct1 WHICH buys PRODUCT OF COLOR Cl1

In SQL language is written as:

```
SELECT CUSTOMER.Code FROM CUSTOMER, PRODUCT, INVOICE
WHERE CUSTOMER.City='Cit1' AND PRODUCT.Color='Col1'
      AND CUSTOMER.Code=INVOICE.Customer AND
PRODUCT.Code=INVOICE.Product
```

Let the statistical data distribution be:

- PRODUCT of Color Col1 = 10
- INVOICE of CUSTOMER FROM City Cit1 = 100.000
- transfer rate on network lines = 10.000 bits/second
- transfer protocol delay = 1 second/message
- Total Transfer Time = Number of Messages + Number of bits / 10.000

Taking into account various strategies we obtain the following responses for query:

Scenario 1: moving PRODUCT on C1
query processing on C1
Total time = 16.7 minutes

Scenario 2: intermediate query processing on C2 for Color matching
intermediate query processing on C2 for Product matching on INVOICE
final query processing on C2 with Customer matching on C1
Total time = 20 seconds

Scenario 3: intermediate query processing on C2 for Color matching moving result on C1
intermediate query processing on C1 for INVOICE matching
final query processing on C1 for City matching
Total time = 1 second

For better response of the system we horizontally fragment PRODUCT table completely and disjoint such that products of color *Col1* are transferred on C1 as a fragment.

We also suppose that we have C3 on the network that represents the management's terminal and C4 that represents the chief's accountant terminal, so we can suppose that they initiate from time to time intensive queries for reports needed.

In our model we are interested to allocate data for optimal response of the system taking into account some factors that may influence the system's behavior.

2. The graph model

Our distributed system can be represented as a graph [3]. We consider the graph $G = (V, E)$ associated with our model, where the sites of the system are given by (V) , the set of vertices and the direct connections between sites represent the edges (E). We also associate to each edge a cost, the meaning of this cost will be explained later.

Our allocation problem can be formulated in terms of graphs theory by finding the graph load balance such as to obtain minimal values for paths from one vertices to another.

After the model is defined and the costs will be explained we can say that the allocation problem is solved more accurately by applying matrix algorithms for min values in graphs.

The graphical representation of our initial model is presented in the figure 1:

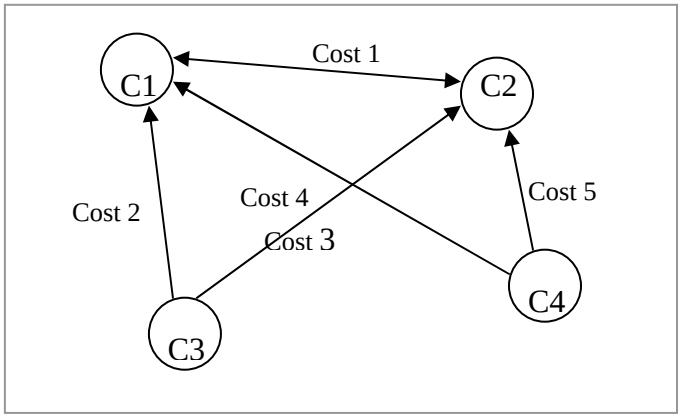

Fig. 1 – the graph model

Queries can be initiated from all the nodes but data is stored only on C1 and C2, thus we obtain the orientation of edges. The initial cost are given by the response time for the query initiated from the starting node to the end node of an edge.

3. Software agents

There are some domains in which we need computer applications to be simple, the system to do what we “tell” him to do, but there are an increasing number of situations when we require systems to decide for themselves what to do to accomplish the tasks. Such computer systems are known as agents [1]. Agents that act in rapidly changing, unpredictable or open environments and where their action can fail are known as intelligent agents.

The agent (fig. 2) takes sensory input from the environment, and produces as output actions that affect it. The interaction is usually an ongoing, non-terminating one.

The key problem facing an agent is that of deciding which of its actions it should perform in order to best satisfy its design objectives. Agent architectures [1] are software architectures for decision-making systems that are embedded in an environment. The complexity of the decision-making process can be affected by a number of different environmental properties: accessibility, determinism, episodic, static or dynamic, discrete or continuous.

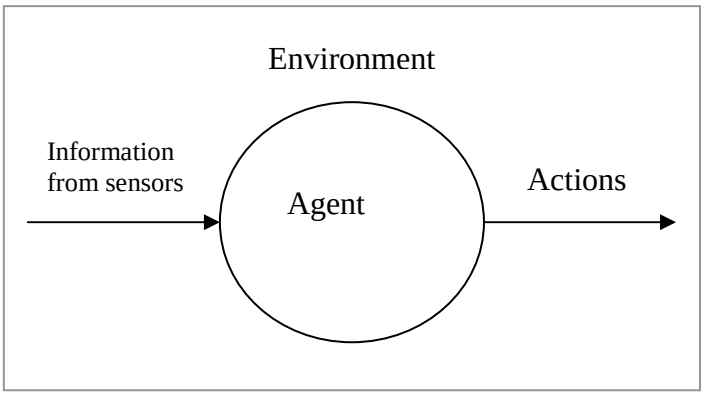

Fig. 2 - An intelligent agent

An intelligent agent is one that is capable of flexible autonomous action in order to meet its design objectives, where flexibility means:

- reactivity: intelligent agents are able to perceive their environment , and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives;

- pro-activeness : intelligent agents are able to exhibit goal-directed behavior by taking the initiative in order to satisfy their design objectives;
- social ability: intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

An expert system [1] is one that is capable of solving problems or giving advice in some knowledge -rich domain. We can differentiate an agent from an expert system by the fact that an expert system acts as a consultant and giving advice in choosing the solution while an intelligent agent can act to change the environment.

Distributed problem solving is characteristic for getting agents to work together well to solve problems that require collective effort. Due to an inherent distribution of resources such as knowledge, capability, information, and expertise among the agents, an agent in a distributed problem-solving system is unable to accomplish its own tasks alone, or at least can accomplish its tasks better when cooperating with others.

Our data allocation problem can be modeled by means of agents because there are already enounced in literature search algorithms for agents. We can categorize this problem as a path-finding problem.

A path-finding problem [2] consists of the following components: a set of nodes *N*, each representing a state, and a set of directed links *L*, each representing an operator available to a problem solving agent. We assume that we know the initial state. Also, there exists a set of nodes *G*, each of which represents a goal state. For each link, the weight of the link is defined, which represents the cost of applying the operator. We call the weight of the link between two nodes the distance between the nodes. We call the nodes that have directed links from node *i* neighbors of node *i*. Moreover we know that in path-finding problems the optimality principle holds. The optimality principle can be formulated as: a path is optimal if and only if every segment of it is optimal. So, if there exists a shortest path from the start node to a goal node, and there exists an intermediate node *x* on the path, the segment from the start node to node *x* is the optimal path from the start node to node *x*. Similarly, the segment from node *x* to the goal state also represents the shortest path from node *x* to the goal state.

The acquisition of new knowledge and motor and cognitive skills and the incorporation of the acquired knowledge and skills in future system activities, provided that this acquisition and incorporation [1] is conducted by the system itself and leads to an improvement in its performance is called learning. There are several types of learning: mutual learning, cooperative learning, collaborative learning, co-learning, team learning, social learning, shared learning, pluralistic learning, and organizational learning. These types can be mixed in a distributed environment to obtain a system that best satisfies the purpose it was constructed for.

4. Solution description

As we mentioned before we model our system as a directed graph, in which by means of intelligent mobile agents we have to find the load balance as to obtain minimal paths from each node a query can be formulated to every other node involved in that query.

There are solutions for data allocation in classical system models. But we want to underline the fact that intelligent agents can react to the environment and change it to better fulfill their goal (see fig. 2). So we can combine a graph model with a path-finding problem for agents as well as with more types of learning: unsupervised and decentralized learning plus centralized or collaborative learning.

The objective of our agent system is to find a path from the initial configuration to the goal configuration, meaning that we need to fragment the data and to obtain a load balance in the graph such that we can obtain reasonable response times from each node where we can initiate a query.

The cost of an edge can now have associated a meaning. We are interested to obtain the best response of the system in the majority of the cases, so we have to perform data allocation having this in mind. For the beginning we can approximate, estimate, the accesses from one site to data from other sites knowing the response time per data unit. So we can build our graph associating the cost of data unit transfer time (as mentioned for fig. 2). When performing data allocation we multiply the physical dimension of the fragment with the cost of unit transfer and thus obtain the distance matrix in our graph. On this matrix we can apply several matrix algorithms for minimal values; some examples [2] are Floyd-Hu and Floyd-Warshall-Hu.

Floyd-Hu algorithm:

We denote $l_{ij}^{*} = \{\min\{l(\mu) \mid \mu \textit{ is path from } i \textit{ to } j\}, i \in X,\ j \in \hat{\Gamma} i \textit{ ; else } \infty$

Consider the matrix $V = (v_{ij})$ of the values of the edges of the graph (as in Bellman-Kalaba) and we define the matrices $V^K, k = \overline{1, n+1}$ such as $V^1 = V$, and by knowing the elements of V^k we compute the elements of V^{k+1} by $v_{ij}^{k+1} = \min\{v_{ij}^k, v_{ik}^k + v_{kj}^k\}$

Phase 1: initializations

Phase 2: base iteration

for *k=1,n* do

for *i=1,n* do

for *j=1,n* do

$$v_{ij}^{k+1} = \min\{v_{ij}^k, v_{ik}^k + v_{kj}^k\}$$

end for

end for

end for

The impact of a system built with mobile intelligent agents resides on the fact that we can provide more accurate information about the data transfer. Intelligent agents can learn from the system and provide some statistics about the mean value of the query occurrence on a specific site, representing a percentage *x* of the cost. Another unit of the cost, with percentage *y* can be the real time importance for a query to be performed from a site. This amount *y* can be different from one edge to another, but for simplicity we consider it to be the same. It can be determined by an agent who travels from site to site and collaborates with other agents or can be transmitted in a cooperative session from one agent to another. So we can perform an initial parameterization and at given times reparameterization to reassign values to these variables. Furthermore we can build the agent system such that it can react and change the system by reassigning some values by itself to improve the response.

The system can change the value of *x* and *y* or can add other variables to better describe the cost.

The final *cost* associated to an edge is given by 3 elements considered: query occurrence *qo*, real time importance *rti* and the response meantime *rmt* refreshed after each query.

The formula for the cost of an edge becomes:

$$\cos t = qo * x + rti * y + rmt * (100 - x - y)$$

But this cost can have a dynamic nature provided that network traffic can be slowed by other data communication (ex. administrative tasks), changes in the topology of the network (ex. a new employee on the accounting dept.) or the actual real time importance of a query initiated from one node of the system (ex. the monthly balance sheet which needs extensive computation).

The role of some agents is to recalculate the value of each cost and to provide two types of updated structures: the cost matrix, which is general for the entire system and that can be viewed by the system administrator (or others who have access to a global view) and for each node a path vector which has as indexes the nodes that can be reached from the current node and values the cost of the path for that specific node. These structures can be obtained by a single agent who travels from node to node performing cost statistics or by individual agents collaborating with each other to reach their goal. The second choice seems to be more flexible provided [1] that an agent can better do a limited amount of computations using only information available from "the neighborhood " and then take appropriate actions based on the available resources. A good example of an algorithm is Decoupled Real-Time Bidirectional Search [1] where two or more problem solvers make independent decisions to achieve the goal. Furthermore we can adopt the Real-Time Multiagent Search algorithm, dividing the cost problem in multiple subgoals that can change during agent lifetime :

$$TC = \sum_{\forall q_i \in Q} QO_i * x + \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} RMT_{jk} * (100 - x - y) + \sum_{\forall q_i \in Q} RTI_{q_i} * y$$

where *TC* represents the total cost, *Q* the query , *S* the site and *F* the fragment, *x* and *y* represent the percentage of the element involved in the cost structure.

5. Conclusions

As stated in the article, mobile intelligent agents can provide a good solution in data allocation problem for distributed database systems.

Main advantages for the use of the agents are the increased autonomy of the system, the increased flexibility to topological changes and even to load balance changes, the system being able to react to these changes. In plus, the human interaction for system administration is significantly reduced, and the optimizations being done not by flair but by correct algorithms implemented in the logic of the system.

As future work we intend to establish the simplest software model and to implement such a system to have real life measurements for the effectiveness of such algorithms conjugated with intelligent agents.

The agent platform that we intend to use may be chosen from ex IBM Aglet Platform or Voyager Platform. These platforms have the development language based on Java, so we can use JDBC bridge to connect to various databases residing on different operating systems.

References:

1.Weiss, Gherhard. (2000). *Multiagent System, A Modern Approach to Distributed Artificial Intelligence,* MIT Press , USA.)

2.Toadere, Teodor . (2002). *Graphs. Theory, Algorithms and Applications,* Editura Albastra, Cluj-Napoca, ROMANIA)

3.Kasa, Zoltan and Varga, Viorica . (1997). *A Graph Model of Distributed Database System*, Studia, Cluj-Napoca, ROMANIA)

4.Oszu, M. Tamer and Valduriez, P. (1991). *Principles of Distributed Database Systems*, Prentice Hall.New Jersey, USA)

5.Grebla, A. Horea and Moldovan, G. (2004). *Replication Techniques for Distributed Database Design*, ICCC 2004,Baile Felix Spa, ROMANIA)

Authors:

Horea Adrian Grebla – "Babes-Bolyai" University of Cluj Napoca, Romania
Grigor Moldovan – "Babes-Bolyai" University of Cluj Napoca, Romania
Sergiu Adrian Darabant – "Babes-Bolyai" University of Cluj Napoca, Romania
Alina Campan – "Babes-Bolyai" University of Cluj Napoca, Romania