Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 37 (2014) 447 – 450

Procedia Computer Science

International Workshop on Intelligent Techniques in Distributed Systems (ITDS-2014)

Distributed Database Design: A Case Study

Umut Tosun*

Baskent University Department of Computer Engineering, Engineering Faculty Baglica Campus, Ankara 06530, Turkey

Abstract

Data Allocation is an important problem in Distributed Database Design. Generally, evolutionary algorithms are used to determine the assignments of fragments to sites. Data Allocation Algorithms should handle replication, query frequencies, quality of service (QoS), cite capacities, table update costs, selection and projection costs. Most of the algorithms in the literature attack one or few components of the problem. In this paper, we present a case study considering all of these features. The proposed model uses Integer Linear Programming for the formulation of the problem.

Peer-review under responsibility of the Program Chairs of EUSPN-2014 and ICTH 2014.
Keywords: Distributed Databases, Replication, Data Allocation

1. Introduction

Query response times, quality of service (QoS), consistency and integrity of data are very important in Distributed Database Management System (DDBMS) applications. In a DDBMS, tables and fragments are distributed on different sites. Each query is executed from a site. The total cost consists of the cost the of query plan execution and the cost of table/fragment accesses through the network. The data allocation problem is NP-complete. Therefore, evolutionary algorithms are generally used to find a minimal cost solution to the problem. Data allocation algorithms try to minimize the table/fragment access cost of the queries. They find an optimal allocation of tables/fragments to sites. They also consider parameters like redundant data, table update costs, and site capacities. There are several factors to be considered when designing a DDBMS. The queries deployed may have shared tasks and same queries may originate from different sites. Site capacities, processing elements, storage and query response times are to be handled at the same time. Therefore, the problem shows the nature of a multi objective optimization problem. We designed a model with Integer Linear Programming (ILP) which has the capability of issuing each of these factors as constraints. Network topology, replication, table/fragment update costs, originating sites, site capacities, query frequencies can all be defined as constraints in this formulation.

*Umut Tosun. Tel.: +0-090-312-2466661/2099 ; fax: +0-090-312-2466660.
E-mail address: utosun@baskent.edu.tr

Peer-review under responsibility of the Program Chairs of EUSPN-2014 and ICTH 2014.
doi:10.1016/j.procs.2014.08.067

2. Related Work

Genetic algorithms,[1,2,3] simulated annealing and mean field annealing[3], and ant colony heuristics[4] are some of the approaches in the literature for the solution of the data allocation problem. All of these methods omit one or more features of the problem. Corcoran[1] and Frieder[2] do not consider site capacities and redundant data. The genetic algorithm, simulated annealing and mean field annealing solutions proposed by Ahmad[3] consider only non redundant data. The ant colony approach proposed by Adl[4] models the problem as a quadratic assignment problem. However, update costs and replication costs were not handled in this work. Several algorithms were proposed with integer linear programming[5,6,7]. These algorithms are generally simple and do not cover a realistic query plan or network topology. These formulations attack only small portions of the problem. They consider a specific part of the problem such as allocation of fragments to sites horizontally/vertically or non-redundant allocation of data. Cornell and Yu[5] proposed a method to assign relations and join operations to sites. Their algorithm tries to minimize the communication costs and aims the utilization of resources while assigning fragments to sites and executing join tasks at the same time. The algorithm lacks to visualize the problem as a combination of query optimization, network utilization and data allocation. The suggested approach tries to solve these problems separately. Furthermore, the proposed integer linear programming formulation is complicated. Ailamaki and Papadomanolakis[8] also used ILP for showing an efficient and realistic bound for index selection. They claimed that the suggested approach finds tightly bounded solutions.

3. Distributed Database Design and Integer Linear Programming

The algorithm first calculates all the distances among the sites by Dijkstra's shortest path algorithm[9]. All input queries are assumed to be left deep. Base tables are considered as the leaves of the query trees. Sample query trees which are used in our case study are shown in Figure 1. Base tables are represented with capital letter T and the tables are named as T_0 through T_n where n is the number of tables. Similarly, the joins are named as t_0 through t_n. Selectivity factor is the ratio of the data to be transferred after the join operation. Base tables can also be truncated by a selectivity factor.

Our network model consists of different link communication speeds as shown in Figure 2. There are three sites S_0, S_1 and S_2. The sites have capacities C_0 = 18MB, C_1 = 15MB and C_2 = 10MB. Links have communication speeds of 100KBps, 200KBps and 500KBps. The queries to be executed are shown in Figure 1. There are three sites and four base tables. In order to represent the assignments of base tables to sites, we use the formalization in Table 1. The total number of variables is 12 for site-table assignments given for this example problem instance. There are 30 constraints and 8 of them are for the constraints stating whether replicas are allowed for each relation by giving the replica count as a constraint (e.g. 1 means no replication for the corresponding table). Next, 4 constraints are given to make sure that the total storage requirements for tables assigned to particular sites do not exceed the storage capacities of each site.

There are 29 equations used for specifying the nodes that perform each operation of given queries and the objective function includes the communication cost for each possible selected path. There are 2 queries used in our examples. Query 1 executes 100 times from originating site S_0. Query 2 executes 20 times from site S_2. In our examples we consider at most 2 replicas for all of the tables. Update ratios' parameters are selected as 0.1, 0.05, 0.5 and 0.1 for tables T_0-T_3. Update costs are calculated by multiplying the ratio by table size. The objective function of the optimization problem is to minimize the sum of costs of transmitting base tables and intermediate results used by queries to sites S_0, S_1 and S_2 while executing the queries. Table size for T_0 = 10MB and T_1 = 8MB which gives 10MB $\times$ 0.5 = 5MB for T_0, and 8MB $\times$ 1 = 8MB for T_1 where 0.5 and 1 are the table selectivity values. When performing the join operation, the resulting intermediate relation t_0 is calculated as 5MB $\times$ 8MB $\times$ 0.3 = 12MB where 0.3 is the join selectivity. Similar to Query 1, Query 2 has two tables T_2 = 6MB and T_3 = 5MB. This results 6MB $\times$ 0.4 = 2.4MB and 5MB $\times$ 0.8 = 4MB where 0.4 and 0.8 are the table selectivity values. When performing the join operation, size of the resulting intermediate relation t_1 is calculated as 2.4MB $\times$ 4MB $\times$ 0.1 = 0.96MB where 0.1 is the join selectivity.

There are a total of 48 variables used to represent the optimization problem as a Linear Programming model. 12 of the values represent table to site assignments as shown in Table 1 whereas 36 of the variables represent the communication costs and update costs. There are 8 equations corresponding to replication formulation. Equation 1

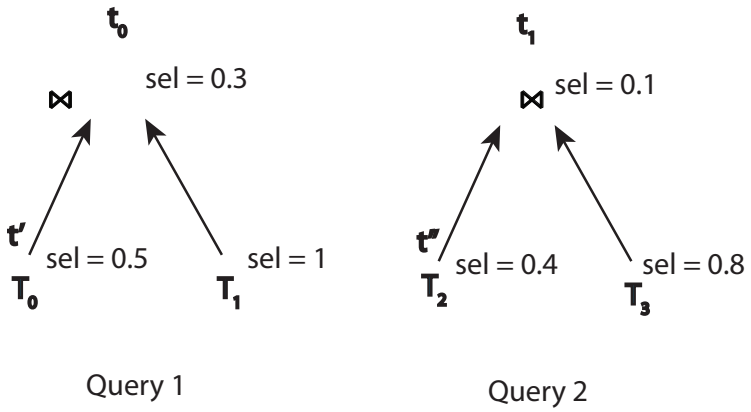

Fig. 1. Query trees used in our examples.

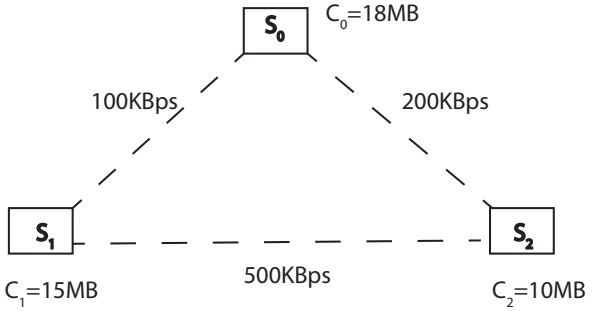

Fig. 2. Site Capacities: 18MB, 15MB, 10MB-Links: 100KB, 200KB, 500KB

Table 1. Table to Site assignment variables used in our model.

Table \ Site	S_0	S_1	S_2
T_0	x_1	x_2	x_3
T_1	x_4	x_5	x_6
T_2	x_7	x_8	x_9
T_3	x_{10}	x_{11}	x_{12}

through Equation 4 represent the minimum number of tables to be inserted to sites. We used these constraints since linear programming aims to set variables to 0 otherwise. Equation 5 through Equation 8 represent the maximum number of tables to be inserted to sites. Generally the system tries to use replicas when update costs are zero. Each query tries to exploit the tables it uses on its originating site.

$$x_1 + x_2 + x_3 >= 1 \quad (1) \qquad x_4 + x_5 + x_6 >= 1 \quad (2)$$
$$x_7 + x_8 + x_9 >= 1 \quad (3) \qquad x_{10} + x_{11} + x_{12} >= 1 \quad (4)$$
$$x_1 + x_2 + x_3 <= 2 \quad (5) \qquad x_4 + x_5 + x_6 <= 2 \quad (6)$$
$$x_7 + x_8 + x_9 <= 2 \quad (7) \qquad x_{10} + x_{11} + x_{12} <= 2 \quad (8)$$

Site capacities are represented with Equation 9 to Equation 11. The communication costs are the most important part of the system. The most important issue with the communication cost variables is that the variable selection for the parts of the query should be consistent. Query 2 originating from S_2 is expressed by equations Equation 12 to Equation 20.

$$10x_1 + 8x_4 + 6x_7 + 5x_{10} <= 18 \quad (9) \qquad 10x_2 + 8x_5 + 6x_8 + 5x_{11} <= 15 \quad (10)$$
$$10x_3 + 8x_6 + 6x_9 + 5x_{12} <= 10 \quad (11) \qquad -x_{12} + x_{42} + x_{45} + x_{48} = 0 \quad (12)$$
$$-x_{11} + x_{41} + x_{44} + x_{47} = 0 \quad (13) \qquad -x_{10} + x_{42} + x_{23} + x_{46} = 0 \quad (14)$$
$$-x_{33} - x_{36} - x_{39} + x_{46} + x_{47} + x_{48} = 0 \quad (15) \qquad -x_{32} - x_{35} - x_{38} + x_{43} + x_{44} + x_{45} = 0 \quad (16)$$
$$-x_{31} - x_{34} - x_{37} + x_{40} + x_{41} + x_{42} = 0 \quad (17) \qquad -x_9 - x_{39} - x_{38} - x_{37} = 0 \quad (18)$$
$$-x_8 - x_{36} - x_{35} - x_{34} = 0 \quad (19) \qquad -x_7 + x_{33} + x_{32} + x_{31} = 0 \quad (20)$$

The objective function which consists of update costs and communication costs are calculated as follows for Figure 2. The communication links have costs 10 sec. for S_0-S_1, 5 sec. for S_0-S_2 and finally 2 sec. for S_1-S_2 links. These costs are average costs to transfer 1 MB of data between two sites. We know that the update ratios for the respective tables are 0.1, 0.05, 0.5 and 0.1. Finally, the objective function for Figure 2 is Equation 21. After running the plan for our example, tables T_0 and T_1 are located at site S_0 and tables T_2 and T_3 are settled in site S_1.

$$\begin{aligned} &x_1 + x_2 + x_3 + 0.4x_4 + 0.4x_5 + 0.46x_6 + 3x_7 + 3x_8 + 3x_9 + 0.5x_{10} + 0.5x_{11} + 0.5x_{12} + 0x_{13} + 5000x_{14} + \\ &2500x_{15} + 5000x_{16} + 0x_{17} + 1000x_{18} + 2500x_{19} + 1000x_{20} + 0x_{21} + 0x_{22} + 17000x_{23} + 8500x_{24} + 5000x_{25} + \\ &12000x_{26} + 7000x_{27} + 1000x_{28} + 13000x_{29} + 6000x_{30} + 0x_{31} + 480x_{32} + 240x_{33} + 480x_{34} + 0x_{35} + 960x_{36} + \\ &240x_{37} + 96x_{38} + 0x_{39} + 96x_{40} + 518.4x_{41} + 240x_{42} + 576x_{43} + 96x_{44} + 96x_{45} + 336x_{46} + 192x_{47} + 0x_{48} \end{aligned} \quad (21)$$

4. Conclusion

In this paper, an Integer Linear Programming formulation for the data allocation problem in distributed databases is proposed. The proposed model exactly handles issues like site capacities, query frequencies and communication costs. The model does not deal with fragmentation and same queries originating from different sites. Selecting the appropriate network topology, network operation costs and query response times are also the other factors to be handled in a realistic design. We plan to extend the algorithm for shared-task queries and fragment management in the future. Load balancing and concurrent task execution are the other criteria to be handled as a future work.

References

1. A.L. Corcoran, and J. Hale, "A Genetic Algorithm for Fragment Allocation in a Distributed Database System," *In Proc. 1994 Symp. on Applied Computing*, pp. 247-250, 1994
2. O. Frieder, and H. T. Siegelmann, "Multiprocessor Document Allocation: A Genetic Algorithm Approach," *Transactions on Knowledge and Data Engineering*, vol.9, no.4 , 1997, pp.640642
3. I. Ahmad, K. Karlapalem, Y. Kwok, and S. So, "Evolutionary algorithms for allocating data in distributed database systems," *International Journal of Distributed and Parallel Databases*, vol. 11, no. 1, pp. 532, 2002
4. R.K. Adl, and S.M.T.R. Rankoohi, "A new ant colony optimization based algorithm for data allocation problem in distributed databases," *Knowledge and Information Systems*, vol. 20, no. 3, pp. 349-372, 2009.
5. D.W. Cornell and P.S. Yu, "Site assignment for relations and join operations in the distributed transaction processing environment," *In Proc. Fourth Int. Conf. on Data Eng.*, pp. 100-108, 1988.
6. B.Gavish and H. Pirkul, "Computer and database location in distributed computer systems," *IEEE Transactions on Computers*, vol. C-35, no. 7, pp. 583-590, 1986.
7. S. Ram and R.E. Marsten, "A model for database allocation incorporating a concurrency control mechanism," *IEEE Transactions on Knowledge and Data Engineering*, vol. 3, no. 3, pp. 389-395, 1991.
8. S. Papadomanolakis and A. Ailamaki. "An integer linear programming approach to database design," *In Proc. of the 2007 IEEE 23rd Int. Conf. on Data Eng. Workshop*, p.442-449,2007.
9. E. W. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numeriche Mathematik*, 1:269-271, 1959.