Optimized Access Strategies for a Distributed Database Design

Rajinder Singh tovirk@yahoo.com
Assoc. Professor
Faculty of Computer Science & Engineering
Guru Nanak Dev University, Amritsar-143001
Punjab, India.

Gurvinder Singh gsbawa71@yahoo.com
Assoc. Professor
Faculty of Computer Science & Engineering
Guru Nanak Dev University, Amritsar-143001
Punjab, India.

Varinder Pannu viki_virk@yahoo.com
Computer Engineer
Faculty of Computer Science & Engineering
Govt. Polytechnic, Amritsar-143001
Punjab, India.

Abstract

Distributed Database Query Optimization is achieved thru many complex sub operations on the Relations, Network Sites, Local Processing Facilities and the Database System itself. Many of these sub problems are NP-Hard itself, which makes Distributed Database Query Optimization a very complex and hard process. One of these NP Hard components is optimal allocation of various sub-queries to different sites of data distribution. Most of prevalent solutions take help of Exhaustive Enumeration Techniques, along with use of innovative heuristics. In this Paper we have proposed a stochastic model simulating a Distributed Database environment, and shown benefits of using innovative Genetic Algorithms (GA) for optimizing the sequence of sub-query operations allocation over the Network Sites. Also, the effect of varying Genetic Parameters on Solution's quality is analyzed.

Keywords: Distributed Query Optimization, Database Statistics, Query Execution Plan, Genetic Algorithms, Operation Allocation.

1. INTRODUCTION

Query Optimization process involves finding a near optimal query execution plan which represents the overall execution strategy for the query. The efficiency of distributed database system is significantly dependent on the extent of optimality of this execution plan. According to Ozsu and Valduriez[1],this process of generating a good query execution strategy involves three phases. First is to find a *search space* which is a set of alternative execution plans for query. Second is to build a *cost model* which can compare costs of different execution plans. Finally in third step we explore a *search strategy* to find the best possible execution plan using cost model.
Before putting any queries to a Distributed Database, one needs to design it according to the needs of an organization. Analysts have to plan data/Fragment allocation according to the nature and frequencies of the various queries at different sites. Different Sequence of operations or sub-operations needed to generate results of a query is very large e.g. as near to n! for relational join of n relations. Decisions have to be taken dynamically for allocation of intermediate relation fragments generated during the query. A Transaction Profile is build to provide this information for various queries to the database. This paper gives a Genetic Algorithm for this step of Execution Plan. It assumes that a transaction profile provides the necessary details of fragment allocation and cost profiles for various pair of sites. It gives a cost model to predict cost of various allocation

plans for intermediate relations and sub operations generated during process of Query. Finally this GA finds the best possible execution strategy in terms of finding sequence and sites for various sub operations, called Operation Allocation Problem[2].
A GA(Genetic Algorithm) has several advantages over other approaches, as it has been successfully applied to a vast set of real world applications, not only that it gives a robust solution but also a good set of alternative possible solutions to choose from [3] and is inherently parallel to achieve good response time.

2. PREVIOUS RESEARCH WORK

Distributed database systems design and query optimization has been and will remain an active area of research for a lot times to come, due to complex and intractable nature of the problem[4,5,6,7,8,9,10].Most of the work has concentrated on two aspects: Data Allocation(The plan of allocating Fragments to various sites) and Operation Allocation(How to generate a sequence of subqueries on various sites). Apers and P.M have discussed in detail the data allocation problem and their fragmentation in [11]. An integrated solution to problems of Data Fragmentation, allocation, replication in Distributed Databases, has been proposed in Tamhankar & Ram[12]. Zehai Zhou[10] propose using heuristics and genetic algorithms for large scale database query optimization.The NP Hard problem is reduced to a join ordering problem similar to a variant of a Travelling Salesman Problem.Several heuristics and a GA is proposed for solving the join order problem.

Simulation experiments for comparison of Branch & Bound, Simulated Annealing, Greedy approaches for operation allocation problem have been describes in detail by Martin & Lam[13].
Frieder and Baru [14] propose dynamic site selection strategies for distributed database design on a microcomputer. March & Rho in [2] have proposed an excellent cost model for reducing local I/O costs, CPU Costs and Communication Costs in operation allocation strategy. Johansonn & Noumann in [15] extended their work bu considering parallel processing and Load Balancing in Data and Operation Allocation.

3. Objective Function & Cost Model

Database Statistics

The main factor affecting the performance of an execution strategy is the size of the intermediate fragments produced during the execution of the sub operations of the query. As many intermediate relations or fragments will need to move over various sites, we need to estimate the size of them to determine transmission costs. This estimation is based on statistical information about base relation and formulas to predict the cardinalities of the results of operations [1].

The set of operations (sub-queries) generated in response to a query can be represented by an operator tree. Nodes of operator tree represent various operations and lines represent cost (based on size of fragment) of operation sequence. A site's Local CPU and I/O costs are proportional to the size in bytes (blocks) of data processed and communication costs depend on communication coefficients between a pair of sites and bytes of blocks moved.

The main assumptions are that Transaction Profiles are known a-priori, providing the details of frequencies of transaction at various sites, base relation sizes and allocation plan at various sites, communication coefficients giving cost of communication amongst various pair of sites and local I/O and CPU coefficients. Also query execution order is given, we emphasize on finding sub-query allocation and cost associated to it with respect to allocation to various sites.
Projection, Selection and Joins account for most of the sub-queries in a Database Query and for simplicity purposes, only these operations have been considered.

3.1 Objective Function Formulation

We start by simulating a design of distributed database by taking a set 'S' of data distribution sites. Set 'R' of relations/fragments stored on those sites. A Set 'Q' as set of transactions.
Let a query transaction ***'q'*** for retrieval, be broken into a set of ***'j'*** sub queries on the 'R' set of relations.

3.1.1 Decision Variables :-

(i) Data Allocation Variable $\boldsymbol{A_{rs}}$

$\boldsymbol{A_{rs}} = 1$ *(if site 'S' holds copy of relation/fragment 'r')*

$\boldsymbol{A_{rs}} = 0$ *(otherwise i.e. fragment 'r' copy is not available at set S)*

(ii) Variables for site selection for sub query execution:

$\boldsymbol{S_{ys}^{q}}$ **:** *(Represents sequence of sub query execution at various sites in the life time of query)*

$\boldsymbol{S_{ys}^{q} = 1}$ *(if subquery 'y' of Query 'q' is done at site s)*

$\boldsymbol{S_{ys}^{q} = 0}$ *(otherwise)*

(iii) For Join operations a notation is proposed to handle left previous operation operation of a join operation (LPO) & right previous operation of a join(RPO) as following:

$\boldsymbol{S_{yv[p]S}} = 1$ *(for [p] = 1 for left previous operation of a Join)*

$\boldsymbol{S_{yv[p]S}} = 1$ *(for [p] = 2 for right previous operation of a Join)*

$\boldsymbol{S_{yv[p]S}} = 0$ *otherwise*

(iv) $\boldsymbol{f^{q}_{ry}}$ represents the query tree in such a way that sub query '*y*' of query '*q*' references the intermediate relation/fragment *r*.

$\boldsymbol{f^{q}_{ry} = 1}$ *(if the base relation 'r' or intermediate fragment 'r' is used by sub query 'y' of 'q' query)*

$\boldsymbol{f^{q}_{ry} = 0}$ *otherwise*

(v) For use of intermediate Relations by Join Operation

$\boldsymbol{f^{q}_{ryv[p]} = 1}$ *(for lpo of join 'y')*

$\boldsymbol{f^{q}_{ryv[p]} = 1}$ *(for rpo of join 'y')*

$\boldsymbol{f^{q}_{ryv[p]} = 0}$ *otherwise*

By making use of above decision variables operation allocation problem formulation is represented as,

Given a input data file highlighting data allocation scheme matrix, given by $\boldsymbol{S^{q}_{ys}}$ Data Allocation Scheme Matrix: A base relation ***y*** stored at site ***s***.
i.e.

Given a Transaction Profile,a Data Allocation Sceme represented by variable $\boldsymbol{A_{rs}}$

And given $\boldsymbol{I^q_{ry}}$ *intermediate relations/fragments is used by sub query* ***y*** *of query* ***q***

We have an objective Function to calculate as to find $\boldsymbol{S^q_{ys}}$

3.1.2) Cost model

Given a set of fragments
R = $\{r_1,r_2,\ldots,r_n\}$

& a network of sites.
S = $\{s_1,s_2,\ldots,s_m\}$

& a set of sub queries
Q = $\{q_1,q_2,\ldots,q_q\}$

Sub Query Allocation problem involves finding the "optimal" possible distribution of R to S.
Ozsu gives a model for Total cost as Total Cost Function having two components: query processing and storage cost as

$$TOC = \sum QPC_i + \sum_{\forall s \in S} \sum_{\forall fj \in F} STC_{jk}$$

Where QPC_i is query processing cost of application q_i and STC_{jk} is the cost of storing fragment F_j at site S_k.

We choose Ozsu's model of query cost as function of sum of local processing costs and transmission costs .We simplify it further by ignoring update costs and ignoring concurrency control costs as we are giving model for retrieval transactions(queries) only. Further concurrent retrievals don't impose any more integrity control costs.
Ozsu's formulation gives

$QPC_i = PC_i + TC_i$ *(PC: Processing Cost, TC :Transmission Cost)*

&

$PC_i = AC_i + IE_i + CC_i$ *(AC: Access Costs, IE : Integrity Enforcement Costs, CC : Concurrent Update control costs)*

In our model we discard the sum of two costs components ($IE_i + CC_i$), because as discussed in Para above, we present a simple model of retrieval queries only.

Therefore Access Costs may be represented as

$$AC_i = \sum_{\forall s \in S} \sum_{\forall fj \in F} (u_{ij} * UR_{ij} + r_{ij} * RR_{ij}) * x_{jk} * LPC_k$$

The summation gives total number of accesses for all the fragments referenced by $\boldsymbol{q_i}$. Now $\boldsymbol{x_{jk}}$ selects only those cost volume entries for sites where fragments are stored actually.

3.1.3) Local Processing Costs

For Simple selection & projections

$LOPC^q_y = \sum_s S^q_{ys}(I\ PO_s \sum_r I^q_{ry}M^q_{ry} + \ CPC_s \sum_r I^q_{ry}M^q_{ry})$ (1)

Where M^q_{ry} = No. of memory blocks of relations 'r' accessed by sub-query y of q.

IPO_s = *Input Output Cost Coefficient of site s in msec per 8k bytes*

CPC_s = *CPU Cost coefficient of site s.*

So equation (1) represents local processing costs of transforming input relation from disk to memory and CPU time for processing a Selection or Projection at sites ***s***.

Ozsu's model ignores join's local processing cost details. For that we have extended this model to add local join costs details as following.

Local processing costs for a join

$LOPC^q_y = \sum_s S^q_{ys} IPO_s \sum_p \sum_r p_v I^q_{ryv[p]} M_q$ 2(a)

$+$

$\sum_s S^q_{ys}(IPO_t \prod_r I^q_{ry} M^q_{ry} + \ CPC_s \prod_r I^q_{ry} M^q_{ry})$ 2(b)

Where p_v is Selectivity Factor & is referred as the ratio of possible different values of a field to the domain of that field.($0 <= p_v <= 1$)

$M_{ryv[p]}$ is the size of intermediate relation

where v[p] represents	p=1	for left previous operation of a join &
	p=2	for right previous operation of a join.

Equation 2(a) represents

Input Output costs in storing intermediate results of previous operations to the site of current join operation.

Equation 2(b) represents

CPU & I/O costs for performing current join operations at site 's'.

3.1.4) Communication Costs:

An involved in case of join operations only as we have assured that selections & projections of retrievals an to be done only at sites which hold a copy of that base relations. Join may be performed at any of possible sites.

$COMM^q_y = \sum_p \sum_s \sum_t S^q_{yv[m]s} * \ S^q_{yt} C_{st} (\sum^q_r I_{ryv[p]} \ M^q_{ryv[p]})$

Where

C_{st} (*is the communication cost coefficient taken from input data matrix)*

$C_{st} = 0$ if $(s = t)$ *(i.e. previous operations and join operation on same site)*

If final operation is not done at the query destination site then a Communication component is added separately for sending the final query result that site.

4. THE GENETIC ALGORITHM (GA_OA)

GA_OA (Genetic Algorithm for Operation Allocation) starts by generating an initial pool of solutions by random generation of operation sequence at given number of sites. An improvement in this previous prevalent approach in [9][10] has been done in this GA is to use transaction profile statistics to generate it.

Each chromosome is evaluated according to objective function and assigned a Fitness value accordingly. Next populations are generated using principles of GA as in [3] i.e. applying SELECTION,CROSSOVER & MUTATION, The fitter a member is more chance it gets to enter the mating pool to generate next population.

Crossover is used so that off-spring shares features of both parents and possibly improves over them. Mutation operator is applied with very small probability like.02 ,so as some important features of parent population of chromosomes are not lost .Elitism is also applied, which ensure that best chromosome of previous population enters next population by 1oo percent probability.

A sequence of integers like 2 1 3 3 4 1 is used to represent a chromosome such that it represents the sub-query allocation plan in the way that, that sub-query 1 is done at site 2, sub-query 2 is done at site1, sub-query 3 is done at site3, sub-query 4 is done at site 3, sub-query 5 is done at site 4, sub-query 6 is done at site 1.

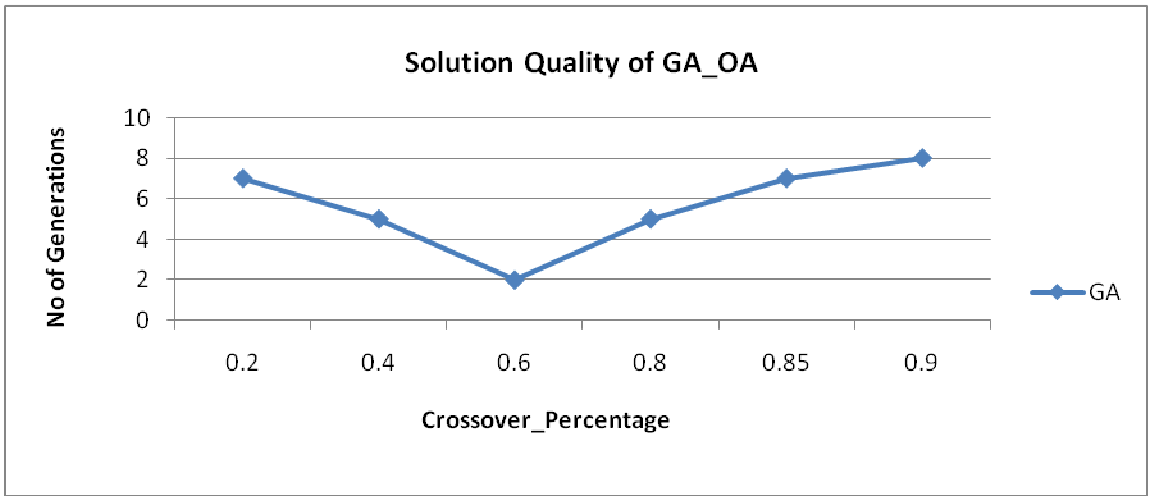
A Structured English representation of GA_OA is outlined below:

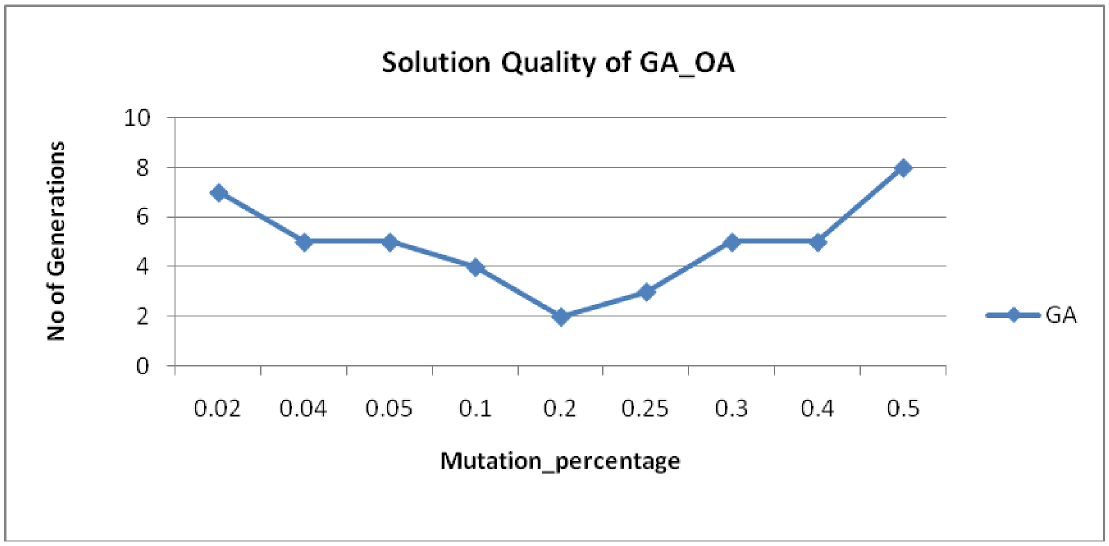
1. *Generate an initial population pool of chromosomes, based on half the members of the pool generated from transaction profiles, data allocation profiles and others selecting randomly over no. of sites .*
2. *Evaluate the fitness of each member based on objective function to reduce the total cost of a query.*
3. *Based on Stochastic remainder method select and give more chance to fitter members to enter a mating pool according to probability proportional to their fitness value.*
4. *To enable Elitism, enter the most fit member of previous generation in mating pool, by replacing it with least fit.*
5. *Apply crossover (probability=0.7) and mutation (0.2) to generate a new population pool.*
6. *Repeat steps 2 to 5 until maximum no of generations are generated.*

5. EXPERIMENTS & RESULTS

Experiments were conducted after coding the GA_OA simulator on an Intel® core™ 2 6420 @ 2.13 GHz machine with 1.00 GB RAM on WINDOWS-XP platform. Exhaustive Enumeration simulator program was developed with varying all possible permutations of sub-query allocation sequence. It was observed that Exhaustive Enumeration run time rises exponentially as compared to GA when we increase no. of joins or no. of sites. When no. of joins are increased from 4 to 10 Exhaustive_ Enumeration chokes very quickly but GA Run Time rises slowly. In case of increasing the number of sites ,Run Time for GA increases linearly whereas Exhaustive_ Enumeration rises exponentially and very quickly becomes almost intractable.

The performance of simulator varied as we vary genetic operators Crossover and Mutation as highlighted by the subsequent graphs, A crossover value of 0.6 was found giving optimal results and mutation parameter of 0.2 achieved optimal solution in least number of generations.





6. CONSLUSION & FUTURE WORK

The aim of this research paper is limited to proposing a stochastic solution to the operation allocation problem of Distributed Database Design. Most of the commercial vendors of Distributed DBMS to date use exhaustive enumeration procedures along with different heuristics. They also incorporate solutions based on other algorithm design techniques like Dynamic Programming, Backtracking etc. This paper highlights that exhaustive procedures quickly go intractable when No. of sites, or, No. of joins are increased suddenly. Exhaustive Enumerations along with heuristics guarantee an optimal solution but total time of query is too large to be practically viable.GA_OA does not guarantee the most optimal solution but provides very near to the best solution in a very short span of time.

In future efforts should be done to incorporate Genetic Based Solutions to allocation problems of Distributed Database. More work needed to be done to ensure that an optimal solution is guaranteed in most of situations by GA's. Furthermore Fragmentation, operation allocation and Data Allocation and Load Balancing need to be integrated in one robust Genetic Solution.

2. REFERENCES

[1] Ozsu & Valduriez. *"Principles of Distributed Database Systems" Pearson Education 2nd Edition,pp. 228-298.*

[2] March,Rho,"Characterisation and Analysis of a Nested Genetic Algorithm for Distributed Database Design".,Seoul Journal of Business pp 85-121 vol2,Number 1. 1995.

[3] Goldberg David.E "Genetic Algorithms in search, Optimization & Learning" *Pearson Education 2nd Edition,pp. 1-55.*

[4] Sacco,G. & Yao"Query Optimisation in Distributed Database Systems"1982,Advances in Computers,21,225-53.

[5] Yu,C.T,Chang" Distributed Query Processing " ACM Computing Surveys,16,399-433.

[6] Graefe,G"Query Evalution Texhniques for a large Database" ACM Computing Surveys,25,73-90, .1993.

[7] March,S.T.,Rho "Allocating Data and Operations to nodes in a distributed database design". IEEE Trans. On knowledge and Data Engg.,7(2). 1995.

[8] Kossman,D. "The state of the art in Distributed Query Processing".,ACM Computing Surveys.,32(4),422-469. 2000.

[9] Cheng,C.H.Lee,W-K,Wong,K-F, "A Genetic Agorithm based clustering approach for database partitioning " IEEE Transactions on System,Man,Cybernetics,32(3),215-230. 2002.

[10] Zehai Zhou,"Using Heuristics and Genetic Algorithms for Large Scale Database Query Optimization," Journal of Information and Computing Sciences,Acadeamim Press-2007.

[11] Apers,P.M.G,1988"Data Allocation in Distributed Database Systems",ACM Trans. On Database Syatems,.13(3),263-304

[12] Tamhankar,A.M & Ram"Database Fragmentation & Allocation: An Integrated Methodolgy and case study." IEEE Transactions on System,Man,Cybernetics,28(3),288-305.

[13] Martin,T,Lam& Russel"An Evaluation of Site Selection Algorithms for Distributed Query Processing"'The Compuer Journal,33(1),61-70,1990.

[14] Frieder, O. Baru"Site and Quey Sechduling policies in Microcomputer Database Systems" IEEE Trans. On knowledge and Data Engg.,6(4).1994.

[15] Johansson,JM,March,ST,Naumann"Modelling Network Latency Paralell Processing In Distributed Database design",Decision Sciences,34(4) 677-706 2003.