Reducing Execution Time of Distributed SELECT Query in Heterogeneous Distributed Database using Genetic Algorithm

Nikhil S. Gajjam
Walchand Institute of Technology, Solapur

S. S. Apte, Ph.D
Head, CSE Dept.
Walchand Institute of Technology, Solapur

ABSTRACT
Centralized unit that coordinates different types of schema running on multiple sites is getting importance now-a-days. Heterogeneous Distributed Database System (HDDS) is the collection of multiple different databases management systems running on multiple systems that are linked together. Query processing is complicated in such cases.

In this work we concentrate on utilizing Genetic Algorithm for finding optimized query execution plan for distributed SELECT queries. Selecting the right set of plans for queries using Genetic Algorithm which minimizes the total execution time is the major goal of this work. Replication of schema is used in this work which gives multiple solutions for retrieval of the data. We used Chromosome for specifying plan for query. Chromosome structures consist of combination of data site and join order.

Aglet, Mobile agent, is used to connect all the database servers with centralized server.

By implementing this, we get the optimized plan for the select query.

Keywords
Heterogeneous Distributed Database, Genetic Algorithm, Aglet.

1. INTRODUCTION

1.1 Heterogeneous Distributed Database System (HDDS):

HDDS is the collection of multiple databases management systems (may be same or different) running on multiple systems that are linked together.

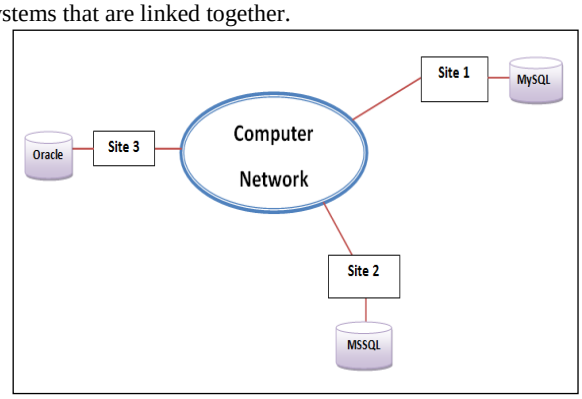

Fig 1: Heterogeneous Distributed Database

In Heterogeneous Distributed Database System, Heterogeneous term specifies different and distributed term specifies multiple. Most of the distributed queries require relations from multiple sites for their processing. The number of possible alternative query plans increases exponentially with increase in the number of relations required for processing the query [17]. Query processing cost also increased for distributed queries because of fragmentation or replication of the data and network cost.

In Fig 1, 3 sites are connected in network consisting of 3 different database management systems. Site 1 consists of MySQL database, site 2 consists of MSSQL database and site 3 consists of Oracle database.

To make database distributed, generally 2 methods are used.

Fragmentation: It is the process of dividing the table either row wise (horizontally) or column wise (vertically).

Replication: It is the process of keeping the table as it is on multiple sites.

1.1.1 Replication

In this research, replication method is used. Replication is the process of storing tables at multiple sites with same schema and same data in the table.

Advantages of replication are

- Increased availability
- Increased reliability
- Less response time
- Less network traffic

1.2 Aglet as a Mobile Agent

Agent: Agent performs task on behalf of user. That task may require single machine or multiple machines.

2 Broader types of agents are

Stationary: Agent that performs task only on single machine where it is created.

Mobile: A mobile agent is not bound to the system where it begins execution. It has the unique ability to transport itself from one system in a network to another [3]. It requires visiting multiple machines to complete its task.

Mobile agents can provide single uniform paradigm for distributed environment. Mobile Agents are the programs that can be dispatched from one machine to another machine for performing distributed task.

Along with mobility, Mobile Agent has following properties

- Autonomous
- Local Interaction
- Parallel Execution

1.2.1 Aglet

Recently Mobile Agent development has been done using Java and has proved to be powerful tool. Java is proved to be more suitable for creating mobile agents because of reasons like platform independence, multithreading programming [4].

Simply aglet is defined as mobile agent designed in Java or Java based mobile agent. Aglet is developed by IBM as Tokyo Lab Project.

Aglet is the class having some methods like start(), stop(), onArrival(), onReturn(), transfer() etc.In this work, Aglet acts as agent and performs tasks that are given by server(centralised or DBserver). Data transfer between the sites is done using Aglet.

1.3 Genetic Algorithm

Genetic Algorithm is an evolutionary algorithm generally used to find an optimal solution among multiple solutions to the problem. It uses crossover and mutation operation for finding multiple solutions for given problem. Fitness function is useful to find out whether we are on track of getting best solution or not. Out of multiple crossover techniques, we used single point crossover technique. Genetic Algorithm has been already used for Distributed Query Plans Generation[15].We extended this work to find best plan for Distributed Query and also executed this plans on Heterogeneous Distributed Environment.

2. RELATED WORK

Many algorithms have been implemented for reducing response time of the running query like dynamic programming algorithms, Genetic Algorithms. Reza Ghaemi, Amin Milani Fard, Hamid Tabatabaee, and Mahdi Sadeghizadeh[13] have presented paper on Evolutionary Query Optimization for Heterogeneous Distributed Database Systems. They have proposed an evolutionary query optimization technique in distributed heterogeneous systems using multi-agent architecture and genetic algorithm approach.They concentrated on join order for optimizing the query, but in our work we concentrated on finding optimized execution plan instead of query optimization in combination with data site.

Ender Sevinç and Ahmet Cosar [5] proposed a new genetic algorithm (GA)-based query optimizer in which chromosome consists of Copy Id, Semi-join, Join-site, Join-order.In our work we concentrated on data site and join order.

Murat Ali Bayir, Ismail H. Toroslu, and Ahmet Cosar [9] presented the Genetic Algorithm for Multiple Query Optimization problem. In this paper they have solved Multiple Query Optimization Problem with different types of Genetic Algorithm. They also compared Genetic Algorithm with A* heuristic technique and proved that Genetic Algorithm works better than A* heuristic technique.

Danny B. Lange, Mitsuru Oshima [4] presented a paper on Mobile Agents with Java: The Aglet API1. They showed that why Java is a powerful technique for mobile agent development. They also specified working of an Aglet, life cycle methods of an Aglet and application of Aglet. In this research, we implemented Aglet life cycle model as specified in the paper.

Kristin Bennett ,Michael C. Ferris,Yannis E.Ioannidis[1]worked on Genetic Algorithm for Query Optimization.They presented a method for encoding arbitrary binary trees as chromosomes and describe several crossover operators for chromosomes. They want to improve GA to work for parallel databases.

Danny B. Lange [3] presented paper on Mobile Objects and Mobile Agents: The Future of Distributed Computing?

Mobile agents reduces network load, encapsulates protocols, executes asynchronously, adapt dynamically. Because of these reasons mobile agents are good in distributed systems.

3. METHODOLOGY

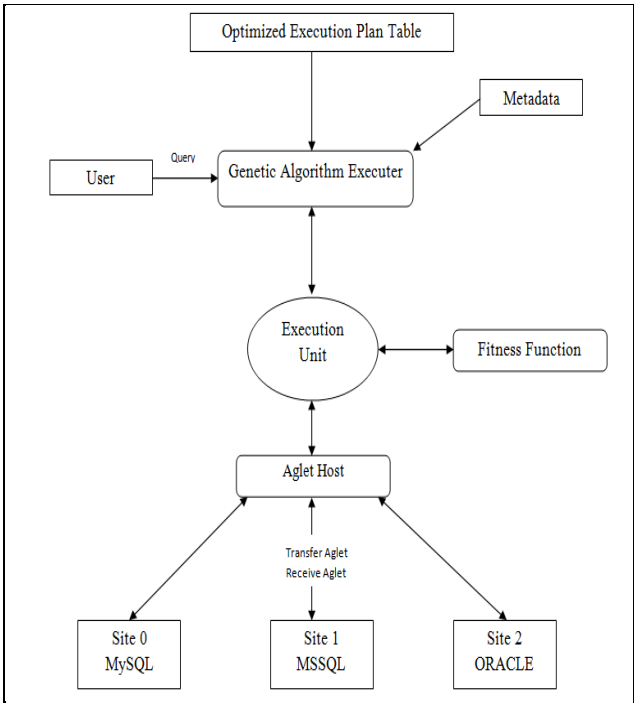

Fig 2: Developed Architecture

Fig 2 shows developed architecture & it consists of following modules.

- GENETIC ALGORITHM EXECUTER
- EXECUTION UNIT
- AGLET HOST
- DBSE RVER

When we start execution, first work that server does is to collect metadata from all the 3 sites. Metadata is nothing but overall schema of database. Metadata is required for knowing which table is stored at which site and columns of the corresponding table.

Client or user gives SELECT query as input to the server. Server is combination of Genetic Algorithm Executer, Execution Unit and Aglet Host.

3.1 Genetic Algorithm Executer (GAE) :

Genetic Algorithm Executer retrieves the query and parses the query i.e. it finds out number of tables and name of the tables present in the query.

Major operations of GAE are selection, crossover and mutation.

Selection – between all individuals in the current population are chose those, who will continue and by means of crossover and mutation will produce offspring population.

Crossover – Different chromosomes are created using interchanging the genes of the previous chromosomes.

Mutation – by means of random change of some of the genes.

Each Chromosome has one fitness value associated with it.
Here fitness=1/time
Where time= time required to execute query using given chromosome structure.

3.1.1 Chromosome Structure

As specified chromosome is nothing but the plan for executing the query. We took Chromosome Structure as a combination of Data Site and Join Order.

Chromosome Structure=Data Site + Join Order

Here Data Site specifies location of the table and Join Order specifies Order for joining the data.

Eg: 3 tables are there in query and that are Student, Teacher, College.

Sample Chromosome is [[2,0,0] [2,1,0]]

Here [[2,0,0] [2,1,0]] means

[2, 0, 0] for Data Site means retrieve Student table from site 2, Teacher table from site 0, College table from site 0.
[2,1,0] for join order means joining of data is performed in order College with Teacher and then result is joined with table Student to find final result.

3.1.2 Validation of Chromosome

Validation of data site means table must be present at that site.

E.g.: if data site is [1, 2, 0] then 1st table must be present at site 1, 2nd table must be present at site 2, 3rd table must be present at site 0. If any one condition is false then chromosome is invalid.

Validation of join order means checking whether joining of table according to join order is possible or not. Joining of 2 tables is possible only if both the tables must have at least one common attribute.If both data site and join order are correct then only we can say that chromosome is valid.

3.2 Execution Unit

Execution Unit is another important module of this work. Main work of this module is to execute the query according to the plan suggested by corresponding chromosome.

Input to execution Unit is valid chromosome. Execution Unit will execute the query according to that chromosome and find out the time required for executing the query according to the plan suggested by chromosome.

First step that Execution Unit does after receiving valid chromosome is to parse the chromosome. I.e. to find out data site and join order. Then Execution unit will create number of aglet (Total number of aglet is equals to the number of database sites. Here it is 3). Then Execution Unit will parse the original query and find out different query for different table.

Next step that execution Unit does is to send aglet to the correspond site that is specified by the chromosome structure along with parsed query. These aglets are received at DBserver. After executing query DBserver will send aglet back to Execution Unit. After retrieving all the aglets from DBserver, Execution Unit will join the partial result in the order specified by the chromosome structure to get complete result.

Execution Unit finds out the time required to get complete result for query according to the plan specified by chromosome. So Execution Unit will contain table containing chromosome structure and time required for executing query with corresponding chromosome structure.

3.3 Aglet Host

Aglet host is the platform for aglets and whose job is to listen to the network for aglets. Aglet Host is present at centralized server and all the DB server site for receiving aglet and sending aglets.

3.4 Database Server (DBserver)

In this work we used 3 DBserver i. e. MySQL, MSSQL, Oracle. Main job of database server is to retrieve query from aglet and execute that query.

4. ALGORITHM

Step 1: Connect centralized server with all the distributed databases & collect metadata to centralised server
Step 2: Receive the query from client. Parse the query
&find valid chromosomes.
While (all valid chromosomes get executed)
{
-create partial queries
-create aglets based on number of tables in query.
-send aglets to corresponding sites
-execute query at DBserver
-Retrieve all the aglets and join partial results
-find out time required for executing chromosome
}
Step 3: Find out optimized chromosome & store it.
Step 4: Write back results back to client.

4.1 Processing of Query

Table 1. Processing of sample Query

Step	Process	Processing done at
1	Select * from student, college where student.college.id= college.college_id	client
2	Table list=student , college And parse query= select college_id from student select college_id from college	GAE
3	Chromosome : data site size=2 Join order size=2	GAE
4	Finds number of valid chromosomes like [[0,0] [1,0]] [[2,2] [0,1]] [[0,2] [0,1]]etc….	GAE
5	Execution Unit one by one from GAE and Execution Unit parse that chromosome Suppose chromosome is [[2,2] [0,1]] Data site=[2,2] Join order=[0,1]	Execution Unit

6	Execution Unit create 2 aglet and sends those aglets to site 2 and site 2 with Aglet1 having query select college_id from student Aglet 2 having query select college_id from college	Execution Unit
7	After receiving aglet, Database server retrieves queries select collge_id from student and select college_id from college and execute those queries Result are stored in corresponding aglet and those aglets are sent back to Execution Unit	Database Server
8	Execution Unit will join those partial result in order student data join with college data	Execution Unit
9	Execution Unit will find ids where condition holds true	Execution Unit
10	Creation of in queries Select * from student where collge_id in(-,-,-) Select * from college where collge_id in(-,-,-)	Execution Unit
11	Again Execution Unit create 2 aglet and sends those aglets to site 2 and site 2 with Aglet1 having query Select * from student where collge_id in(-,-,-) Aglet 2 having query Select * from college where collge_id in(-,-,-)	Execution Unit
12	After receiving aglet, Database server retrieves queries Select * from student where collge_id in(-,-,-) and Aglet 2 having query Select * from college where collge_id in(-,-,-) and execute those queries Result are stored in corresponding aglet and those aglets are sent back to Execution Unit	Database Server
13	Execution Unit will join those partial result in order student data join with college data	Execution Unit
14	GAE calculate time required for chromosome	GAE
15	Similarly all chromosomes are executed one by one. then GAE will find fittest chromosome	GAE
16	Finally results are sent to client	Execution Unit

5. RESULT AND ANALYSIS

Experiments were carried out on following scenario.

Table 2. Replication of table at sites 0, 1, 2

Site 0	Site 1	Site2
Student Teacher College	Teacher	Student Teacher College

Table 3. Databases at sites 0, 1, 2

Site	Database
Site 0	MySQL
Site 1	MSSQL
Site 2	Oracle

Table 4. Parameters setting for Genetic Algorithm

Parameter	Value
Number of Generation	100
Population Size	20
Crossover Percentage	75
Mutation rate	4

Table 5: Result analysis of Query1

Query1 : Select * from student , college	Chromosome		Time Required for executing chromosome	Fitness (fitness=1/time)
	Data Site	Join Order		
	[0, 0]	[0, 1]	0.124	8.064
	[2, 2]	[0, 1]	0.825	1.2121
	[0, 0]	[1, 0]	0.059	16.6461
	[2, 0]	[1, 0]	0.335	2.9850
	[0, 2]	[1, 0]	0.364	2.7472
	[2, 2]	[1, 0]	0.750	1.3333
	[2, 0]	[0, 1]	0.406	2.4630

Here for query "select * from student, college" Genetic Algorithm has created 7 chromosomes. Execution Unit has executed all these valid chromosomes and finds out the time required for executing each chromosome.

Here third chromosome with data site = [0,0] and join order is [1,0] requires 0.059 sec. which is minimum amongst all the chromosome

Chromosome with maximum fitness value is the best chromosome. So chromosome [[0,0] [1,0]] gives optimal plan for given query. And that plan is [[0,0] [1,0]] indicating retrieve table student from site 0 and table college from site 0.And perform join in order college table joining with student table.

Table 6: Result Analysis of Query2

Query2: Select * from student, Teacher, college	Chromosome		Time Required for executing chromosome	Fitness (fitness=1/time)
	Data Site	Join Order		
	[2, 0,0]	[2,1,0]	0.321	3.11
	[2,2,0]	[2,1, 0]	0.693	1.44
	[2, 2,0]	[0,2, 1]	1.054	0.9487
	[0,1, 2]	[1,2,0]	0.394	2.5380
	[0,1, 0]	[2,1,0]	0.163	6.134
	[2,1, 0]	[0,2,1]	0.282	3.5460
	[0, 2,0]	[1,2,0]	0.308	3.2467

Here query is "select * from student, teacher, college". Here fifth chromosome with data site = [0,1,0] and join order is [2,1,0] requires 0.163sec which is minimum amongst all the chromosomes. Chromosome [[2,0,2] [2,0,1]] gives optimal plan for given query with fitness value=6.134.

We store queries along with its optimized plan at the server side. We use these optimized plan when same query is fired again for execution.

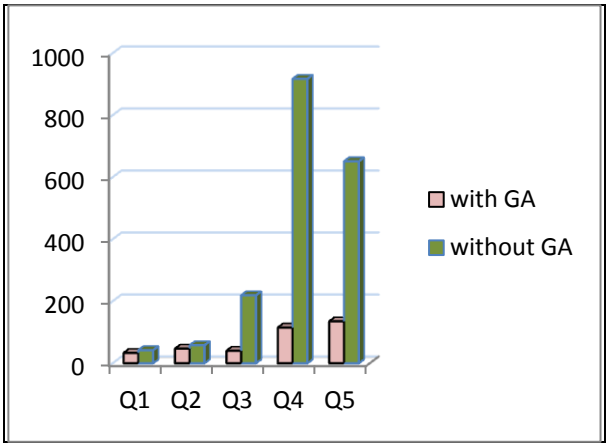

Fig 3. comparison of time required for execution With GA and without GA

Above figure shows the difference in time(msec) when query is executed with optimized plan of Genetic Algorithm and without using Genetic Algorithm where

Q1: select * from student
Q2: select * from student,teacher
where student.student_id>2
Q3: select * from student,college
Q4: select * from student, college
where student.college_name= college.college_name
Q5: select * from student,teacher,college

6. CONCLUSION AND FUTURE WORK

In this work, multiple plans were generated for given SELECT query using Genetic Algorithm. And retrieved optimized plan from all the plans on the basis of fitness value and time. This optimized plan is useful when similar query is fired for execution. At that time, we execute that query with optimized plan.

We also compared time required for executing query using Genetic Algorithm and without using Genetic Algorithm (joining of the data is performed in order of the tables that is specified in query). And it is observed that execution time is reduced by minimum 20% when query is executed using Genetic Algorithm.

This work is more efficient in environment where Database having large data and replication of tables.

As a future work, we are planning to extend this work for horizontal and vertical fragmentation of the tables. Natural language processing techniques can be included in this work.

7. REFERENCES

[1] Bennet K, Ferris MC, Ioannidis YE (1991) A genetic algorithm for database query optimization. In: Proc 4th Int Conf Genetic Algorithms, SanDiego,Calif,pp400–407 2011

[2] B. Lange, D. T. Chang, ÏBM Aglets Workbench - Programming Mobile Agents in Java", IBM Corporation White Paper, September 1996..

[3] D. B. Lange, Mobile Objects and mobile agents: The future of distributed computing? In Proceedings of The European Conference on Object-Oriented Programming, 1998.

[4] D. B. Lange and M. Oshima, "Mobile agents with Java: The Aglet API," World Wide Web Journal,1998.

[5] Ender Sevinc and Ahmet Co¸sar(2011). An Evolutionary Genetic Algorithm for Optimization of Distributed Database Queries. The Computer Journal, Vol. 54 No. 5

[6] E.-P. Lim and J. Srivastava(1993), 'Query optimization/processing in federated database systems', in Conference of Information and Knowledge Management.

[7] Joachim Baumann, Fritz Hohl, Kurt Rothermel, and Markus Straßer. Mole— concepts of a mobile agent system. World Wide Web Journal, 1(3):123–137,1998.

[8] Michael L. Rupley, Jr. (2008): Introduction to Query Processing and Optimization. http://www.cs.iusb.edu/technical_repots/TR- 20080105-1.pdf

[9] Murat Ali Bayir, Ismail H. Toroslu, and Ahmet Cosar(2007). Genetic Algorithm for the Multiple-Query Optimization Problem. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, VOL. 37, NO. 1

[10] M. Stillger and M. Spiliopoulou, "Genetic programming in database query optimization," in Proc First Annu. Conf. Genetic Programming,Stanford, CA, July 1996

[11] N. B. Herodotos Herodotou and S. Babu. Query optimization techniques for partitioned tables. In SIGMOD Conference,pages49–60,

[12] Papastavrou, S., Samaras, G., Pitoura, E.: Mobile Agents for WWW Distributed Database Access. In: Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia (1999) Available at<http://ada.cs.ucy.ac.cy/˜cssamara/DBMS-agents/Paper/papastavrous.ps>

[13] Reza Ghaemi, Amin Milani Fard, Hamid Tabatabaee, and Mahdi Sadeghizadeh September 2008, Evolutionary Query Optimization for Heterogeneous Distributed Database Systems. World Academy of Science

[14] Smith, J. M., P.A. Bernstein, U. Dayal, N. Goodman, T. Landers, K.W.T. Lin, E. Wong. MULTIBASE -- Integrating heterogeneous distributed database systems. Proceedings of 1981 National Computer Conference, AFIPS Press, 487-499

[15] T.V.Vijay Kumar, Vikram Singh, "Distributed Query Processing Plans Generation Using GA", IJCTE, Vol 3. No.1, Feb 2011

[16] Wiesman, M., et al. Database Replication Techniques: A Three Paramater Classification. in 19th IEEE Symposium on Reliable Distributed Systems. 2000. Nuernberg, Germany

[17] Y..E. Ioannidis and Y.C. Kang, "Randomized algorithms for optimizing large join queries, ACM 1990