An Efficient technique for fragmentation and allocation in distributed database management system

A Dissertation Submitted in the Partial Fulfillment
of the Requirements for the award of Degree of

Master of Technology
(Computer Technology and Application)
Department of Computer Engineering
Delhi Technological University, Delhi-110042

Submitted by

PRERNA PAHWA
22/CTA/2010

Under the Esteemed Guidance of
Mr. MANOJ SETHI
Department of Computer Engineering
Delhi Technological University, Delhi-110042

DELHI TECHNOLOGICAL UNIVERSITY, DELHI, INDIA

2012-2013

CERTIFICATE

Date……

This is to certify that work contained in this dissertation entitled "**An Efficient technique for fragmentation and allocation in distributed database management system**" submitted in the partial fulfillment, for the award of degree of Master of Technology in Computer Technology and Application at **DELHI TECHNOLOGICAL UNIVERISTY** by **PRERNA PAHWA, Roll No.:-22/CTA/10** is a bonafide record work done by her under my supervision and guidance in the academic year 2012-2013. The report embodies result of work and studies carried out by her and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.

Mr. Manoj Sethi
(Project Guide)
Department of Computer Engineering
Delhi Technological University, Delhi-110042

ACKNOWLEDGMENT

First of all, let me thank the almighty GOD, my parents and my In- Laws who are the most graceful and merciful for their blessings that contributed to the successful completion of this project.

I take the opportunity to offer sincere thanks and deep sense of gratitude to my learned supervisor **Mr. Manoj Sethi** for expressing his confidence in me by letting me work on a project of this magnitude and providing his support in implementing this project. His invaluable guidance, patient reviews and continuous encouragement only has made me complete this dissertation.

I also want to express my sincere thanks to our HOD, Dr. Daya Gupta, Department of Computer Engineering, for providing well equipped infrastructure support.

I am thankful to my husband Rahul Ajmani for his moral support all the time; he has been always around to cheer me up in the odd times. I would also like to take this opportunity to express the profound sense of gratitude and respect to all those who helped me throughout the duration of this project.

I would like to hear from any readers with comments, suggestions or bugs on my e-mail address pahwapre@gmail.com.

PRERNA PAHWA
22/CTA/2010
M.Tech. (COMPUTER TECHNOLOGY AND APPLICATION)
DEPARTMENT OF COMPUTER ENGINEERING

AN EFFICIENT TECHNIQUE FOR FRAGMENTATION AND ALLOCATION IN DISTRIBUTED DATABASE MANAGEMENT SYSTEM

ABSTRACT

Distributed database provides the better solution to large-scale data management problems. A very important research issues is database system performance. Recently developed cloud database (CDBMS) is defined as distributed database. Similarly, in 2012, Bigdata is designed as a distributed database architecture running over 100s of clusters gi.e. machines. Distribution of data is a cumulative process of fragmentation, allocation and replication.

The research objective of thesis is to propose an efficient technique for fragmentation and allocation in distributed database management system. The problem of fragmentation and allocation is considered as combined problem due to interdependency. However, replication is not considered in this research due to its high processing and communication cost as suggested by [4]. The allocation problem is NP-complete [3] and thus requires fast heuristics to provide efficient solution. Clustering of sites is done before fragmentation to ensure more efficiency, as clustering reduces the communication cost. The concept of clustering is proposed in [17] and fragmentation in [1]. Various strategies for allocation, such as cost based [4], preference based and nearest neighborhood allocation (NNA) [16], proposed are used to generate initial population for evolutionary algorithm developed. Finally the allocation of fragments is done using evolutionary algorithm developed in the thesis. Using the initial population of solutions, the algorithm proposed either generates a new solution for allocation or finds the best solution among the strategies considered. Much of attention has been paid to allocation as it is key factor in slashing query execution cost. Evolutionary algorithm is being developed due to its potential in solving NP- complete problems. The process of clustering, horizontal fragmentation, allocation strategies and evolutionary algorithm for allocation is taken as combined process and is applied to dataset of [1]. The system performance is enhanced using clustering at initial stages. Also, the performance of all the strategies and proposed algorithm is evaluated and the results show that the proposed algorithm provides near to the optimal solution for allocation of fragments to the clusters with the assumption of finding the average best solution.

TABLE OF CONTENTS

Abstract……...……....i
List of Tables……..……..iv
List of Figures……..……vi

1. Introduction…….……1
1.1 Distributed Database: Definition…….....2
1.2 Literature Survey……...…….....3
1.3 Problem Definition……...…….....5
1.4 Motivation ……...…….....7
1.5 The Outline of Thesis ……..…….....7
2. Fragmentation and Allocation…….....9
2.1 Fragmentation ……...……... 10
2.1.1 Degree of Fragmentation..... 10
2.1.2 Rules of Fragmentation…….....11
2.1.3 Types of Fragmentation …….…….....11
2.2 Horizontal Fragmentation…….…….…….....12
2.3 Vertical Fragmentation …….....13
2.4 Allocation of Fragments…….…….....14
2.5 Replication …….……...…….....14
3. Basics of Clustering and Evolutionary Algorithms……..... 16
3.1 Clustering: - Grouping of sites ……...……..……..17
3.2 Evolutionary algorithms for data allocation…….....17
3.2.1 Genetic algorithm for data allocation……..18
3.2.2 The Simulated Evolution algorithm for data allocation……..18
3.2.3 The mean field annealing algorithm for data allocation……...….....18
3.2.4 Random neighborhood Search algorithm for data allocation……...….....19
3.3 Summary…….....19

4. Research Methodology/Proposed Solution……..20

4.1 Phases…….....21

4.2 Clustering Sites……...22

4.3 Horizontal Fragmentation……..…23

4.4Allocation Solutions…….....24

4.4.1 First Strategy of allocation……...…….....24

4.4.2 Second Strategy of allocation …….....25

4.4.3 Third Strategy of allocation ……...…….....25

4.5Allocation …….....26

4.5.1Fragment Evolutionary Allocation Algorithm (FEAA) …….....27

5. Implementation and Simulation Results…….28

5.1 Specifications…….....29

5.2 Simulation Results for Horizontal fragmentation and Allocation……..29

5.2.1 Clustering Sites…….....30

5.2.2 Horizontal Fragmentation…….....33

5.2.3 Allocation Solutions…….....37

5.2.3.1 First Strategy of allocation…….....38

5.2.3.2 Second Strategy of allocation …….....42

5.2.3.3 Third Strategy of allocation …….....44

5.2.4 Allocation…….....46

5.3 Discussion…….……..…….55

6. Conclusion and Future Scope.....56

6.1 Conclusion…….…….....57

6.2 Future Scope…….…….....57

References……59

Annexure A Coding……...62

Annexure B Screenshots…….. 67

List of Tables

Table 5.1: Employee Relation….....29

Table 5.2: Network Sites with Constraints……30

Table 5.3: Distance Cost Matrix of sites……....30

Table 5.4: Clustering Sites……..30

Table 5.5: Cluster Communication Cost Matrix (CCCM)……...31

Table 5.6: Attribute Retrieval and Update matrix (ARUM)…..……..33

Table 5.7: Cluster Attribute Retrieval and Update Matrix (CARUM)…….....34

Table 5.8: Attribute Retrieval and Update Table (ARUST)…….....35

Table 5.9: Attribute Retrieval and Update Table (ARUST) with summation of values……..36

Table 5.10: Fragment 1 (F1) having Salary > 1500…….36

Table 5.11: Fragment 2 (F2) having Salary < 1500…….....37

Table 5.12: Fragment 3 (F3) having Salary = 1500…….37

Table 5.13: Clusters with Constraints…….....37

Table 5.14: RUSM (Retrieval Update Sum Matrix)…….....38

Table 5.15: Cost Allocation Matrix (CAM)…….38

Table 5.16: Allocation Table 1…….…….....38

Table 5.17: Comparison Table 1……..39

Table 5.18: Comparison Table 2…….....40

Table 5.19: Preference Table……42

Table 5.20: Allocation Table 2…….....43

Table 5.21: Comparison Table 3……...43

List of Tables

Table 5.22: Allocation Table 3…….....45

Table 5.23: Comparison Table 4…….45

Table 5.24: Initial population Table……47

Table 5.25: Fitness Table……....47

Table 5.26: Mutation Table 1……..50

Table 5.27: Mutation Table 2……..52

Table 5.28: Best Solution Table……..54

Table 5.29: Final Allocation Table……..54

Table 5.30: Comparison Table of various strategies of allocation with FEAA …….55

List of Figures/Charts

Figure 4.1: System development phases……21

Chart 5.1: Fragment allocation to clusters using first strategy of allocation……41

Chart 5.2: Fragment allocation to clusters using first strategy of allocation when……...41
fragment F3 allocated to S2 of C2

Chart 5.3: Fragment allocation to clusters using first strategy of allocation when……42
fragment F3 allocated to S4 of C2

Chart 5.4: Fragment allocation to clusters using second strategy of allocation when……44
fragment F1 & F2 allocated to S4 of C2

Chart 5.5: Fragment allocation to clusters using third strategy of allocation….....46

CHAPTER 1

INTRODUCTION

Introduction

Distributed database is an important research area. The reason to have distributed databases is that it is more reliable and responsive and many of the computer applications are inherently distributed. Web-based applications, electronic commerce business over the Internet, multimedia applications such as news-on-demand or medical imaging are some of the examples of applications that requires distributed database[27]. The traditional centralized database approach is restricted. The major problem with centralized database is if it fails then no user can access the data. The distributed database has been developed as solution to the problem. Distributed database performance is very important and critical. This chapter presents a brief of distributed database system. This chapter is organized as follows:- Distributed Database definition and literature survey is presented in Section 1.1 and Section 1.2 respectively. The problem definition is described in Section 1.3. Motivation behind carrying out the work is described in Section 1.4. The outline of thesis is presented in Section 1.5.

1.1 Distributed Database: Definition

M.Tamer Özsu, Patrick Valduriez[27] defined distributed database as a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (distributed DBMS) is then defined in[27], as the software system that permits the management of the distributed database and makes the distribution transparent to the users. According to, Özsu and Valduriez[27], distributed database must be considered as a tool that makes distributed processing easier and efficient. They explained that logical collection of files not only form distributed database but there should be a proper structure among the files and the files must be accessed via common interface. C.J. Date[13] defines distributed database (DDB) is a kind of virtual database, whose component parts are physically stored in a number of distinct "real" databases at a number of distinct sites (in effect. it is the logical union of those database). Ceri and Pelagatti[7] defined a distributed database as a collection of data that logically belongs to the same system but is spread over

the sites of a computer network. Nicoleta - Magdalena Iacob (Ciobanu)[19] described distributed database system as a collection of sites, connected together via some kind of communications network, in which:

a. Each site is a full database system site in its own right, but

b. The sites have agreed to work together so that a user at any site can access data anywhere in the network exactly as if the data were all stored at the user's own site.

Donald Kossmann[22] stated that distributed database is feasible with advancement in communication, hardware, software protocols and standards and it is needed too. Further[22] mentioned that almost all major database system vendors such as IBM, Oracle, Sybase, etc..., offer products to support distributed data processing, and large database application systems have a distributed architecture (e.g., business application systems such as Baan IV, Oracle Finance, Peoplesoft 7.5, and SAP R/3).

1.2 Literature Survey

A large amount of work exists in the area of distributed databases. The work in distributed database system (DDBS) has started as early as in 1970s[8, 31].The work in distributed database system can be roughly classified as: - work on distributed database architecture, fragmentation, allocation, replication, clustering of sites, work on query processing and much more. The thesis focuses on two main research areas of distributed database system, fragmentation and allocation. Fragmentation and allocation plays important roles in development of cost efficient system[27].

Fragmentation

Fragmentation can be classified as horizontal, vertical and mixed fragmentation. The thesis primarily focuses on horizontal fragmentation, HF. In[30] fragmentation is considered as a design technique which divides the single database in two or more partitions such that

combination of the partitions yields the original database without loss of information. Ceri[7] first proposed min-term predicate algorithm for primary horizontal fragmentation in which a set of disjoint and completed predicates P= {P1, ..., Pn} should be determined .Özsu and Valduriez[27] proposed an iterative algorithm COMMIN to generate a complete and minimal set of predicates from a given set of simple predicates. Bai˜ao et al.[6] and Zhang[35] first grouped the predicates using graph-based technique according to predicate affinity matrix to build a predicate affinity graph and then perform primary horizontal fragmentation and thus define horizontal class fragments. Ra[28] proposed a graph-based algorithm for horizontal fragmentation such that the predicates clustering based on the predicates affinities. [1] Proposed a horizontal fragmentation technique on the basis of retrieval and update frequencies of predicates. Cheng et al.[9] proposed a genetic algorithm based clustering approach for horizontal fragmentation. [9]Considered problem of horizontal fragmentation as travelling salesman problem (TSP). [9]Needs additional analysis to cluster predicates, as they have not considered data locality while clustering predicates.

Allocation

Data allocation is the critical and important research issues of the distributed database design and affects the performance of the system. Many researches have been done in allocation. In earlier days, Chu[10], proposed a non redundant simple model for allocation of files. In[10] a global optimization model was introduced to investigate file allocation problem. [27] Considered allocation as the process of assigning a node on the network to each fragment after the database has been properly fragmented. Ezeife and Barker[15] tried to enhance the performance of application by minimizing the amount of irrelevant data accessed. Kamalaakar [20] focuses on minimizing the amount of data transfer in processing and applications. [2] Proposed a mathematical model using greedy approach for data allocation. In [17, 4] the allocation is done on the basis of cost of allocating fragments. Cost includes cost of processing and cost of communication between clusters. Some researchers do not consider replication while making decision of allocation [3, 11, 21, 25]. In [5, 18] storage capabilities of network nodes are ignored while allocation.

[4, 17] Considers the communication cost model for allocation. In [16] nearest neighborhood allocation scheme was proposed based on optimal algorithm. In [11, 29] the allocation is done by clustering sites and using genetic algorithm for allocation. Some authors [24] combined the security considerations into fragment allocation. Sarathy et al. [33] provide a nonlinear integer programming formulation and its linearization for solving allocation problem. Sacca et al. [32] assumed network and processing capacities are limited so considered transmission of fragments on the basis of query cost only. Tamhankar and Ram [34] proposed an integrated fragmentation and allocation approach considering replication and concurrency control cost. They suggested vertical fragmentation must be used for movement of data but did not discuss how to perform vertical fragmentation.

1.3 Problem Definition

Fragmentation and allocation are major issues in distributed database management system. Several researchers have proposed various algorithms and strategies to solve the problem of fragmentation and allocation. Much attention has been given to allocation, as it is very critical and can save the cost of query execution. Various strategies on the basis of cost models, nearest neighborhood search etc. are developed for solving allocation problem. But these approaches have their own limitations. In nearest neighborhood algorithm, the fragment is reallocated with respect to changing data access patterns with time constraints. Although, this algorithm shows better results than optimal allocation algorithm [16] but it has certain limitations. It does not consider cost factors while reallocation. And, too much reallocation may degrade the system performance. Apers[5] considered the allocation of the distributed database to the sites so as to minimize total data transfer cost but this approach is quite complicated. The greedy heuristics used may not give optimal solutions. Also, in many researches, the fragmentation and allocation problem is considered independently. However, they both are interdependent, so should be considered together. By reviewing the various strategies of fragmentation and allocations, this research tries to propose an efficient algorithm for allocation which is more reliable and consider various aspects of allocations such as cost models, nearest neighbor allocation approach, preference based allocation etc.

The research objective is to propose **“An Efficient Technique for fragmentation and allocation in Distributed Database Management System”.** Due to interdependency between fragmentation and allocation, both of them should be studied together in the research problem. The research problem is wide and NP-complete and requires fast heuristics to generate solution. The problem should include solutions from various allocation strategies on the basis of cost model proposed in[4], preference and nearest neighborhood allocation (NNA) proposed in[16]. Further, the said, allocation strategies should be scrutinized. And, either best of any of the above strategy or a new strategy should be found to allocate fragments to the clusters. In the research problem much emphasis is given on fragment allocation due to its importance in query execution. Usage of evolutionary algorithm is recommended while allocation due to its ability to solve NP-complete problems and fast results. In the research problem allocation of fragments should be done simultaneously. Fragmentation can be horizontal or vertical. The problem requires to do a horizontal fragmentation on a relation as proposed in[1], based on given read and update frequencies. The problem also requires clustering of sites as proposed in[17]. Clustering of sites is grouping of sites together to reduce the communication cost and to reduce the complexity of peer-to-peer connection for large number of sites. The problem should also develop cluster communication cost matrix (CCCM) for given set of sites. The problem will be divided into four phases. Phases start from clustering of sites to horizontal fragmentation to finding different allocation solutions and at last allocation using evolutionary algorithm developed. In the problem all the phases must be considered as interconnected. The second phase is relaxed in terms of network sites constraints while the other phases consider this constraint. The problem also needs to evaluate performance from time to time. The problem requires consideration of all the phases sequentially. Clustering of sites must be done at initial stage and allocation at final phase. Like [3, 11, 21, 25], the problem does not consider replication at time of allocation, since the replication slows down the system performance. The problem demands a thorough study and development of all the phases to find the effective and optimal solution.

1.4 Motivation

The introduction section revealed distributed database importance. Distributed databases (DDBs) are more reliable and responsive. DDBs provide better solution to large-scale data management problems. Distributed database is the need of an hour and its performance is a key research area. Cloud database is defined as distributed database. There are several reasons that motivate to work on the distributed databases. With the advancement in the communication techniques, global organizations are getting decentralized. With distributed databases, it is easy to grow the size of database with almost no impact on the existing system. It supports many applications like teleconferencing, e-commerce, multimedia application etc…With distributed databases query-response time is greatly reduced. Economically it is more affordable too. Its availability makes it more attractive. In, 2012 the concept of Bigdata is introduced. Bigdata is designed as distributed database architecture running over clusters. However, the full benefit of improved performance can only be achieved with right distributed database design. Distributed database uses concept of fragmentation and allocation. Both the fields are extensively explored by the researchers as discussed in the literature survey. Both the fields are challenging and together considered as NP-complete problem. The challenges lead the thesis to investigate the field in detail with the aim of improving performance of the system.

1.5 The Outline of the thesis:-

In this chapter problem is addressed and the motivation behind thesis is discussed. The remaining thesis is organized as follows:-

Chapter 2 discusses the fundamentals of fragmentation and allocation. In this chapter various techniques for fragmentation and allocation is discussed.

Chapter 3 will give an overview of various evolutionary algorithms.

Chapter 4 will present an algorithm for allocation of fragments in distributed database management system. This chapter will present a proper research methodology used for fragmentation and allocation.

Chapter 5 shows the simulation results and performance evaluation.

Chapter 6 finally concludes the work and proposes future research.

References are given to include all the sources that had been referred to.

CHAPTER 2

FRAGMENTATION AND ALLOCATION

This chapter discusses about basics of fragmentation in section 2.1 and allocation in section 2.5. Section 2.1.1, briefly discusses about degree of fragmentation and section 2.1.2, about rules of fragmentation. Horizontal fragmentation used in this research is discussed in detail section 2.2, while replication is explained in brief in section 2.3.1.

2.1 Fragmentation

Fragments are the subrelations of a relation, obtained by dividing the relation. It is more common to consider the fragments rather than distributing relations. Fragmentation is important due to many reasons such as:-

1) A relation is not viewed as whole by many applications. Rather a subset of relation is accessed by the applications.

2) If the applications that view a given relation are geographically distributed then the alternative of replication of entire relation is not desirable for limited storage.

3) A relation cannot be considered as a relation. Whereas the fragments treated as a unit, allows the number of transactions to execute concurrently.

2.1.1 Degree of Fragmentation

The extent to which the database should be fragmented is an important decision that affects the performance of query execution[27]. The degree of fragmentation should not be very low or also should not be extremely high. The fragmentation is dependent on the number of parameters characterized by the applications.

2.1.2 Rules of Fragmentation

1) Completeness condition:

The whole data of the global relation must be mapped into the fragments. No data item should be left. It should be part of any of the fragments.

2) Reconstruction condition:

The global relation must always be reconstructed from those fragments. The reconstruction must ensure that constraints on data must be preserved and moreover it must comply with this architecture.

3) Disjointness condition:

Horizontal fragmentation ensures that fragments should be disjoint, to explicitly control the replication of data at allocation. Vertical fragmentation may violate this condition.

2.1.3 Types of Fragmentation

Data fragmentation is decomposed mainly into: - horizontal fragmentation and vertical fragmentation. [26]Proposed mixed fragmentation. Mixed fragmentation can be obtained by combining these two types of fragmentation. In all types of fragmentation, a fragment can be obtained by some operation executed on global relations and the result produced is the fragment. The rules stated in section 2.1.2 must be followed when doing fragmentation.

2.2 Horizontal Fragmentation

Horizontal fragmentation, HF, partition the tuples of a global relation into subsets which are useful for the distributed database system. There are two types of horizontal fragmentation: primary and derived. Primary horizontal fragmentation of a relation is performed using predicates that are defined on that relation. Derived horizontal fragmentation, on the other hand, is the partitioning of a relation that results from predicates being defined on another relation[27]. Primary horizontal fragmentation can be achieved using minterm-predicate-based approach or affinity-based approach.

Example for primary horizontal fragmentation

Let a global relation be:-

ACCOUNTS (Type, Balance, Branch-loc)

Here the Accounts relation contains information such as type of account, balance and the branch-loc where the account is opened. However if the entire branch-loc is either Chicago city (“Chg”) or Florida city (“Fl”) then the horizontal fragmentation can be defined in the following way:

$$\text{Accounts1} = \text{Acc}_{\text{Branch-loc= “Chg”}} \text{ ACCOUNTS}$$

$$\text{Accounts2} = \text{Acc}_{\text{Branch-loc = “Fl”}} \text{ ACCOUNTS}$$

The above fragmentation satisfies the completeness condition because “Chg” and “Fl” are the only possible values of Branch-loc attribute. The reconstruction condition is also verified, because it is always possible to reconstruct the ACCOUNTS relation through the union operation:

$$\text{ACCCOUNTS} = \text{Accounts1} \cup \text{Accounts2}$$

Where, U is union operator.

Now, if another predicate say, Balance also need to be considered while fragmentation. However, Balance is either >3000 or <=3000.Then following minterm predicates {m1... m4} that can be defined

m1: Branch-loc = “Chg” ^ SAL <= 30000
m2: Branch-loc = “Chg” ^ SAL > 30000
m3: Branch-loc = “Fl” ^ SAL <= 30000
m4: Branch-loc = “Fl” ^ SAL > 30000

Then, the above minterm predicates are considered to fragment the global relation ACCOUNTS.

2.3 Vertical Fragmentation

The vertical fragmentation, VF, can be obtained by the subdivision of attributes of a global attributes. VF is useful when the subgroups have the same common geographical properties. It is more complex than the horizontal fragmentation. Two approaches that exist for vertical fragmentations are:-

a) Grouping: - Assigning each attribute to one fragment and joining some of fragments at each step until some condition is met. In 1979, grouping was first suggested by Hammer and Niamir for centralized databases and later on Sacca and Wiederhold in 1985 extended it to distributed database.

b) Splitting: - It decides beneficial partitioning based on the access behaviour of attributes by the applications. It came into existence in 1975 for centralized database and has been extended to distributed database in 1984 by Navathe et al.

2.4 Allocation of fragments:-

The data allocation problem is an important research area and has been explored extensively. Earlier it was characterized as the 'file allocation problem'. But this strategy has some problems as discussed below:-

1. Fragments cannot be modelled as individual files, as the fragments have the same structure or behaviour.

2. The fragments can be more than global relations and analytical models fail to solve the problems with too many variables.

3. In distributed database it is quite difficult to model the application behaviour.

Treating allocation as file allocation will not be able to give the proper solution. It is hard to solve problem 3 stated above, if data allocation is treated as file allocation. Rather data allocation should be thought of as optimized application work. Much work has already been done considering data allocation as optimized problem. For allocation problem some greedy approach, non linear solutions, evolutionary algorithm approaches etc. are used.

2.5 Replication:

While allocation it is important to decide whether one wants to replicate fragments on different sites or not. Replication is efficient and reliable in case of queries have more read frequencies on the other hand for more update queries replication can be unreliable and troublesome. Replication can be fully or partial. A non replicated database is also known as partitioned database. In full replication, the complete database is present at each site. In partial replication some of the fragments are may be repeated at certain sites.

Replication further introduces complexity in the distributed database design because:-

a) For partial fragmentation it is difficult to decide degree of fragmentation.

b) Applications may have several sites to read fragment, as due to replication fragments are present at multiple sites.

c) Update queries causes a real problem.

d) Concurrency control and directory management is difficult for partial replication.

The general solution to the replication problem is to determine the sites where it is beneficial to store a copy of fragments. The benefit of allocation must be higher than the cost of movement.

CHAPTER 3

BASICS OF CLUSTERING AND EVOLUTIONARY ALGORITHMS

In this chapter the concept of clustering is discussed in brief in section 3.1.Basics of evolutionary algorithm for data allocation is given in section 3.2. Further, various evolutionary algorithms like genetic algorithm in (3.2.1), simulated evolution algorithm in (3.2.2), mean field annealing in (3.2.3) and random search algorithm in (3.2.4) is discussed.

3.1 Clustering: - Grouping of sites

Clustering is grouping of sites according to certain criteria to increase the system performance .Clustering also decreases the storage overheads. Sites are grouped according to some criteria or threshold value. Many experiments verified that grouping of sites reduces the communication cost. Moreover, if number of sites is enormous then it will be difficult to set up peer-to-peer connections. Sites are grouped if there distance cost of communication is less than the threshold value. The threshold value determines the number of sites allowed to be together for maximum difference of communication cost. The threshold value is determined by the network of distributed database.

3.2 Evolutionary algorithms for data allocation

The data allocation problem is NP- Complete[3] and requires fast heuristics to generate optimal solutions. The capability of evolutionary algorithms to solve NP-complete problems makes logical to use it for data allocation. Data allocation problem has direct analogy to various other problems listed below, for which evolutionary algorithms provided optimal solutions.

a) Plant location problem in operations research.
b) Knapsack problems
c) Network flow problems
d) Travelling Salesman problem

Various evolutionary algorithms are proposed which has considerable potential to solve data allocation problem is discussed one by one below:-

3.2.1 Genetic algorithm for data allocation

Genetic algorithm (GA) based search methods are inspired by the mechanisms of natural genetics leading to the survival of the fittest individuals [3]. GA operates on initial population which is binary encoded representation of set of solutions. Then the fitness of each solution is evaluated and solutions with higher fitness are put in a mating pool. The selection is determined using some mechanisms like rank selection, roulette wheel or tournament selection. Selected solutions are then subjected for crossover and mutation to deliver next generation. [3]Suggests, Crossover operator exchanges portions between strings and mutation, causes sporadic and random alternation of the bits of strings. The process is repeated to generate better solutions.

3.2.2 The Simulated Evolution algorithm for data allocation

It is similar to the genetic algorithm (GA) described above with a slight difference that it uses mutation as primary search mechanism and relies less on crossover. Thus, the crossover rate is less in simulated evolution algorithm and mutation rate is high. For genetic algorithm it's vice-a versa. Also, in simulated evolution algorithm the chromosome representation is based on problem data.

3.2.3 The mean field annealing algorithm for data allocation

Mean field annealing (MFA) technique combines the collective computation property of the famous Hopfield Neural Network (HNN) with simulated annealing[3]. Originally, it was proposed for solving travelling salesman problem. The MFA algorithm is derived from an analogy to the Ising spin model which is used to estimate the state of a system of particles or

spins in thermal equilibrium[3]. As analogous to MFA, a matrix is constructed having m rows and n columns, where rows are data fragments and columns represent sites. The value 1 in matrix indicates fragment is allocated to site. The value is either 0 or 1. The energy function corresponds to the data transfer cost function. The mean field is computed using energy function and site matrix. And, site matrix is updated. The final allocation is determined with largest spin value. In a best solution each data fragment is allocated to one site.

3.2.4 Random neighborhood Search algorithm for data allocation

It is an effective optimization technique with low complexity[3]. In this initial solution is randomly generated with moderate quality. Then the algorithm starts searching the neighborhood for another solution. Then a comparison is drawn between initial solution and neighborhood solution. The better solution is then accepted and then again its neighborhood is searched. The search carries on until the best solution is found or number of steps reached upper limit. This technique relies heavily on the construction of the solution neighborhood[3].

3.3 Summary

In this chapter, various evolutionary algorithms proposed for data allocation are surveyed. A discussion on clustering of sites is also made. It is found that the clustering of sites enhances system performance. Also, the four evolutionary algorithms are discussed which can be used for data allocation.

CHAPTER 4

RESEARCH METHODOLOGY / PROPOSED SOLUTION

This chapter introduces the research methodology used while developing this project. Section 4.1 explains the phases used in developing the project. Moving on phase by phase, section 4.2 discusses the first phase i.e. clustering of sites. Horizontal fragmentation used in developing the project is discussed in section 4.3. The next section 4.4 provides the various allocation solutions details on the basis of cost (4.4.1), near neighborhood (4.4.2) or preference based solution (4.4.3) for allocation. In section 4.5, a new evolutionary based algorithm is designed to give near to optimal solution for allocation.

4.1 Phases

The research aims to perform horizontal fragmentation (HF) and allocation of fragment to sites or clusters and propose an efficient technique for fragmentation and allocation.
Various phases for system development are shown below:

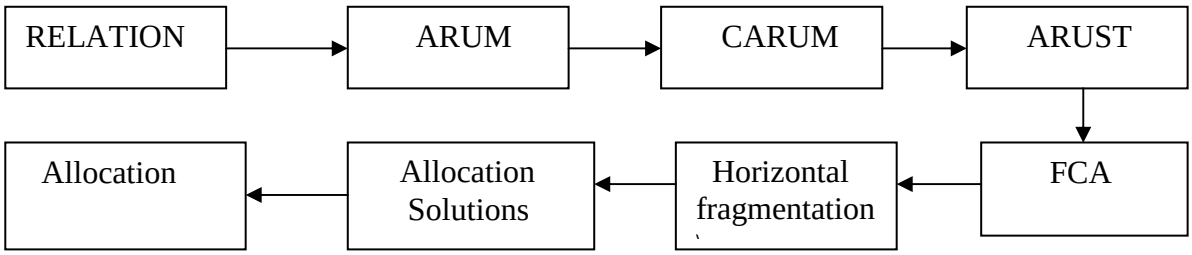


Figure 4.1: System development phases

The notations used in System development

ARUM	Attribute Retrieval and Update Matrix
CARUM	Cluster Attribute Retrieval and Update Matrix
ARUST	Attribute Retrieval and Update Summation Table
FCA	Fragment Candidate Attribute

The System primarily consists of four phases:

i) Clustering Sites,

ii) Horizontal Fragmentation,

iii) Allocation Solutions

iv) Allocation.

The second phase is relaxed in terms of sites or cluster constraints while in third and fourth phase the constraints will be maintained.

4.2 Clustering Sites

This is the first phase of the system and is very essential as grouping sites (clustering) according to certain criteria to enhance system performance. Grouping sites into clusters helps in reducing the communication costs between the sites during the process of data allocation[17]. Sites are clustered according to their communication cost, which finds whether or not a set of sites is assigned to a certain cluster. The method used for clustering is considered as a fast way to determine the data allocation to a set of sites rather than site by site[17].Moreover, if number of sites is large then it is difficult to maintain peer-to-peer connection.

The sites are clustered on the basis of communication cost which can be found from distance cost matrix of sites. Those sites are grouped together whose communication cost is less than or equal to CCR, where “CCR” is the number of communication units which are allowed for the maximum difference of communication cost between any two sites to be grouped in the same cluster [29].The number is determined by the DDBs network administrators[29]. For clustering, a clustering parameter, cp ($S_{i,}$ S_j), given below is used.

$$cp\ (S_{i,}\ S_j) = \begin{cases} 1;\ ccc\ (S_{i,}\ S_j) <= CCR \wedge i \neq j \\ 0;\ ccc\ (S_{i,}\ S_j) > CCR\ V\ i \neq j \end{cases} \quad (1)$$

Where, ccc ($S_{i,}$ S_j) is communication cost between each pair of sites S_i and S_j

S_i is Site number i, where i: 1<=i<=number of sites

S_j is Site number j, where j: 1<=j<=number of sites and i ≠ j.

4.3 Horizontal Fragmentation

The input relation is horizontally fragmented i.e. partitioning of tuples of a relation, as in[1]. The steps for horizontal fragmentation are as follows:-

Step 1:- Obtain cluster attribute retrieval and update matrix (CARUM) from attribute retrieval and update matrix (ARUM).

Step 2:- Fragment candidate generation on basis of Average Retrieval Updation and Summation table (ARUST). The attribute having highest value is considered as candidate attribute for fragmentation. For calculating ARUST, the formula given below is used.

ARUST = $\sum_l^c \sum_j^n \sum_i^m \sum_k^a (\boldsymbol{RFp} * \boldsymbol{freq} + \boldsymbol{Ufp} * \boldsymbol{freq})$ (2)

Where, p is set of predicates, 1<=p<=3,

a represents no. of attributes, 1<=k<=a,

i represents query index, i= {1…m},

j represents site index, j= {1…n},

l represents cluster index, l= {1…c} where, c is total number of clusters.

Formula (2) is derived from formula (5) and formula (6) in[1].

Step 3:- Perform fragmentation on relation on basis of Fragment candidate generated.

4.4 Allocation Solutions

After obtaining the fragments another step is allocation of fragments. Allocation of fragments has analogy to the knapsack problem or network flow problem. Allocation of fragments in distributed database requires much effort. It has the greater impact on the quality of the final solution and hence the operational efficiency of the system[4].

Initially the allocation solution using different strategies based on cost model proposed by[4], near neighborhood allocation (NNA) [16] and preference based on dataset[1] is found and then using these solutions and applying the evolutionary algorithm, final allocation is done. The allocation phase aims at minimizing cost factors and maximizes performance (in terms of throughput, response time etc...) considering the cluster/site constraints. The allocation phase also considers the cluster /site constraints.

4.4.1 First Strategy of allocation

Using this strategy the fragments are allocated on the basis of cost factors which includes cost of communication and cost of processing (read and update queries cost).In this strategy , the cluster which have maximum cost of allocation for a fragment is determined and the fragment is allocated to that cluster.

Cost of allocating fragment to cluster includes many factors. Majorly, cost is incurred at times of read and update by a query for the fragment at cluster times the number of frequency of read and update issued by a query for the fragment at cluster. This can be local or remote retrieval and update .Therefore, the cost of allocation includes both local and remote retrieval and update. As analogy to cost functions in [4] cost of allocation is calculated, which is obtained as matrix (CAM). Formula is shown below:-

Cost of Allocation Matrix (CAM) = Cluster Communication Cost Matrix (CCCM)
*** Retrieval Update Sum Matrix (RUSM) (3)**

While simulation, CCCM should be evaluated in phase 1 i.e. clustering of sites.

$$\mathbf{RUSM}=\sum_{\mathbf{ci}}^{\mathbf{c}}\sum_{\mathbf{si}}^{\mathbf{n}}\sum_{\mathbf{fi}}^{\mathbf{m}}(\mathbf{RFqi},\mathbf{fi},\mathbf{si},\mathbf{ci}*\mathbf{Freqfi},\mathbf{si},\mathbf{ci})+(\mathbf{UFqi},\mathbf{fi},\mathbf{si},\mathbf{ci}*\mathbf{Freqfi},\mathbf{si},\mathbf{ci}) \quad \textbf{(4)}$$

where,

fi represents fragment index, 1<=fi<=m ,

si represents site index, 1<=si<=n,

ci represents cluster index, 1<=ci<=c,

qi represents query index,

m represents total number of fragments,

n represents total number of sites,

c is total number of clusters.

4.4.2 Second Strategy of allocation

Using this strategy the fragments are allocated on the basis of its preference at particular site (or cluster). This strategy is considered because in dynamic allocation the preference of fragment at the sites (or clusters) plays an important role. To avoid frequent oscillations fragment is allocated to sites (or clusters) on basis of its preference. The preference is determined by retrieval update sum table (RUSM), i.e.it is found that the fragment is preferred more on which cluster. The concept has an analogy to attribute locality precedence (ALP) discussed in[4].

4.4.3 Third Strategy of allocation

In this strategy the fragments are allocated to the nearest neighbor of demanding site and in the path which has maximum access for the fragment. This strategy is also known as NNA algorithm[16].It is different from optimal algorithm as destination of fragment is the neighbor site. This approach will reduce delay of movement and thus improving the response time. This algorithm is suitable for DDS in the networks which have low bandwidth[16].

4.5 Allocation

With so many strategies described above to allocate the fragments, the process will merge them in order to find either the best of them or find a new strategy for allocation better than the three strategies described above using fragment evolutionary allocation algorithm (FEAA) for fragment allocation. **FEAA is developed in the thesis with the idea to find best solution for allocation**. Evolutionary algorithms easily solve NP-complete problems. Evolutionary algorithms are being already used to solve "the travelling salesman problem", "knapsack problem", which are NP-complete problems and data allocation has direct analogy to these problems. The data allocation problem, however, is NP-complete, and thus requires fast heuristics to generate efficient solutions[29]. Furthermore, the optimal allocation of database objects highly depends on the query execution strategy employed by a distributed database system, and the given query execution strategy usually assumes an allocation of the fragments[29]. The Fragment evolutionary allocation algorithm (FEAA) for fragment allocation uses some operators such as "mutation" and "crossover" which allows survival of the better solution. The operators "mutation" and "crossover" are used by genetic algorithm and evolutionary algorithms.

Proposed Algorithm Methodology

The Fragment evolutionary allocation algorithm (FEAA) for fragment allocation will work on binary encoded representation of allocation solutions obtained from using the above three strategies. For instance, if fragment is allocated to cluster C2 then it will be represented as 10. These encoded solutions will be considered as population. Population is equivalent to the genetic material of individuals in nature[29]. FEAA will manipulate the population of potential solution using the fitness function, "crossover" and "mutation" operator to find optimized solution for data allocation. The fitness function considered here will be cost function. From the population the individuals with best fitness are candidates for survival for

next generation and low fitness individual will be known as elite children, which are not passed further. The crossover operator, also know for recombination, will allow successful sharing of information between two successful individuals and mutation has role of restoring lost genetic material. The crossover rate is kept high and mutation rate is kept small. The strategy used for crossover is uniform. The process will keep on repeating until the best solution is obtained and number of generation is less than maximum generation.

4.5.1Fragment Evolutionary Allocation Algorithm (FEAA):-

Create a random initial population from the allocation solutions obtained. Encode each individual of population using binary strings.

1. Sequence of new populations is then created. At each step, the individuals in the current generation are used to create the next population. For generating the new population, the algorithm performs the following steps:
   a. Evaluate fitness value (cost function) for each individual in current population.
   b. Members, called parents, are chosen on the basis of their fitness.
   c. Lower fitness members are chosen as elite and are not passed to the next population.
   d. Using uniform crossover and bit flap mutation children are produced from the parents.
   e. Replaces the current population with the children to form the next generation.

3. The algorithm stops when number of generation is less than maximum generation.

4. Final fragment allocation will be selected by selecting individual with fittest value

CHAPTER 5

IMPLEMENTATION AND SIMULATION RESULTS

In this chapter the results found by simulating the dataset given in[1] are presented. Clustering, fragmentation and allocation with proposed algorithms and FEAA is done phase by phase. The output of one phase becomes input for other phase. Also, the performance is evaluated.

5.1 Specifications

Software: NetBeans IDE 7.2.1, MS Window Vista Version 2007, MS Access 2007
Hardware: Intel(R) Core(TM)2 Duo CPU T6500 @ 2.10 GHz, 4.00 GB of RAM.

5.2 Simulation Results for Horizontal fragmentation and Allocation

Candidate table for horizontal fragmentation and allocation:-

Name	Birth-date	Job-Id	Salary	Location	Dept-Id
Mich	4/2/1980	MGR	2000	Riyadh	2
Jone	2/6/1984	MAN	1400	Jeddah	3
Nancy	12/1/1978	MAN	1750	Makah	2
Den	22/9/1989	MAN	2100	Riyadh	1
Jone	22/9/1990	MAN	1500	Riyadh	3
Matth	2/2/1982	MAN	1150	Abha	4
Adam	4/2/1980	MGR	1300	Abha	2
Kevin	12/12/1979	MAN	1200	Jeddah	1
Zamel	22/9/1977	MAN	1500	Makah	1

Table 5.1: Employee Relation [1]

Total Sites taken is four. Sites Constraints is shown below:

Site	Capacity (C)	Fragment Limit (FL)
S_1	100	5
S_2	80	1
S_3	60	3
S_4	90	3

Table5.2: Network Sites with Constraints [1]

Distance Cost Matrix of sites as shown below:

Sites	S1	S2	S3	S4
S1	0	5	9	9
S2	5	0	14	4
S3	9	14	0	11
S4	9	4	11	0

Table 5.3: Distance Cost Matrix of sites [1]

5.2.1 Clustering Sites

By definition of CCR in 4.2.1, CCR=4 is used accordingly, and applying formula (1) on distance cost matrix of sites in table 5.3 clustering of sites is done:-

Site / Cluster	S1	S2	S3	S4
C1	1	0	0	0
C2	0	1	0	1
C3	0	0	1	0

Table 5.4: Clustering Sites

Table 5.4 shows generated clusters and respective sites in the clusters. From the above table three clusters C1,C2 and C3 are generated and sites S1 belongs to C1,S2 and S4 belongs to C2 and S3 belongs to S4.

Next, the average communication cost within and between the clusters is calculated and a cluster communication cost matrix (CCCM) is constructed using the formula below given by [29].

$$\text{Average cc} = \frac{\text{sum of cc between cs}}{\text{Number of communication between cs}} \quad (5)$$

Where, cc = communication costs
cs = clusters sites

Using (5), on table 5.3 and table 5.4 the cluster communication cost matrix (CCCM) is given in table below:

Cluster #	C1	C2	C3
C1	0	7	9
C2	7	8	12.5
C3	9	12.5	0

Table 5.5: Cluster Communication Cost Matrix (CCCM)

Performance Evaluation with Clustering

It is based on number of communications and communication cost. The performance evaluation is done on similar lines as in[29].

(i) Grouping sites into clusters reduces the number of communications costs [29].In the simulation, 4 sites are used and therefore, total number of communication will be (4 * 4 =16). **After clustering the communication reduced from 16 to 9** (3 * 3) as three clusters are there and each cluster will communicate with other two clusters and their sites will communicate together.

(ii) Communication cost is also reduced due to clustering .For example:-before clustering the communication cost between sites of cluster1 (C1) and cluster2 (C2) would have been cost of communication (table 5.3) between (S1,S2)+(S1,S4)+(S2,S1)+(S2,S4) which is equal to 28(5+9+5+9).After clustering the cost of communication between C1 and C2 will be (C1,C2)+(C2,C1) which is equal to 14(7+7). **Therefore, the communication cost is reduced from 28 to 14.**The improved performance is computed by formula below as given by [29].

$$\text{Improved percentage} = \frac{\text{reduced cost of communication}}{\text{Total cost of communication}}$$

$$= \frac{28-14}{28} = 0.50$$

Therefore, due to clustering the performance of the system is enhanced by 50 %.

5.2.2 Horizontal Fragmentation

The horizontal fragmentation will divide the input relation according to cluster attribute retrieval and update matrix (CARUM) which is obtained from attribute retrieval and update matrix (ARUM) [1] and shown below.

S	Q	Freq	Mod	Birth-date			Salary			Location		
			RF/UF	P1	P2	P3	P1	P2	P3	P1	P2	P3
S1	Q1	3	RF	1	0	0	1	1	2	0	0	1
			UF	2	0	1	1	0	0	2	2	0
	Q2	5	RF	3	1	0	2	3	1	1	2	0
			UF	0	1	1	2	2	0	1	2	0
S2	Q2	2	RF	3	1	0	2	3	1	1	2	0
			UF	0	1	1	2	2	0	1	2	0
	Q3	4	RF	0	2	1	2	5	0	1	3	1
			UF	1	0	0	2	1	0	2	0	1
S3	Q1	6	RF	1	0	0	1	1	2	0	0	1
			UF	2	0	1	1	0	0	2	2	0
	Q4	8	RF	2	0	2	0	1	1	1	1	0
			UF	0	2	1	1	2	0	1	3	0
S4	Q4	9	RF	2	0	2	0	1	1	1	1	0
			UF	0	2	1	1	2	0	1	3	0
	Q5	3	RF	1	0	1	2	0	1	2	2	1
			UF	1	0	2	3	1	0	0	0	3

Table 5.6: Attribute Retrieval and Update matrix (ARUM) [1]

The notations used in Table 5.6

S	Site
Q	Query
Freq	Access Frequency of query at site
RF	Retrieval Frequency
UF	Update Frequency
P	predicate index , p={1,..,3}

In ARUM table, attributes birth-date (p1: birth-date >82, p2: birth-date < 82, p3: birth-date = 82), salary (p1: salary > 1500, p2: salary < 1500, p3: salary =1500) and location (p1: location="S1", p2: location="S2", p3: location="S3") are considered as proposed in[1]. In the first step, CARUM matrix is constructed from the above ARUM matrix and same predicates are used.

a) Cluster Attribute Retrieval and Update Matrix(CARUM)

CARUM is created using ARUM given in table 5.6 and clustering sites table 5.4.

Cluster	Site	Query	Frequency	Mod RF/UF	Birth-date			Salary			Location		
					P1	P2	P3	P1	P2	P3	P1	P2	P3
C1	S1	Q1	3	RF	1	0	0	1	1	2	0	0	1
				UF	2	0	1	1	0	0	2	2	0
		Q2	5	RF	3	1	0	2	3	1	1	2	0
				UF	0	1	1	2	2	0	1	2	0
C2	S2	Q2	2	RF	3	1	0	2	3	1	1	2	0
				UF	0	1	1	2	2	0	1	2	0
		Q3	4	RF	0	2	1	2	5	0	1	3	1
				UF	1	0	0	2	1	0	2	0	1
	S4	Q4	9	RF	2	0	2	0	1	1	1	1	0
				UF	0	2	1	1	2	0	1	3	0
		Q5	3	RF	1	0	1	2	0	1	2	2	1
				UF	1	0	2	3	1	0	0	0	3
C3	S3	Q1	6	RF	1	0	0	1	1	2	0	0	1
				UF	2	0	1	1	0	0	2	2	0
		Q4	8	RF	2	0	2	0	1	1	1	1	0
				UF	0	2	1	1	2	0	1	3	0

Table 5.7: Cluster Attribute Retrieval and Update Matrix (CARUM)

The notations used in Table 5.7

C	Cluster,[index from{1,..,3}]
S	Site,[index from {1,…,4}]
Q	Query
Freq	Access Frequency of query at site
RF	Retrieval Frequency
UF	Update Frequency
P	predicate index , p={1,..,3}

In CARUM table attributes are same as in ARUM table 5.5 with same conditions for p1, p2, p3 and same values.

b) **Fragment Candidate Attribute (FCA)**

For finding fragment candidate attribute, first calculate Attribute Retrieval and Update Summation Table (ARUST).

Using formula (2) ARUST table is formulated as given below:-

Cluster/Attribute	Birth-date	Salary	Location
C1	42	65	45
C2	106	126	122
C3	80	70	78

Table 5.8: Attribute Retrieval and Update Table (ARUST)

Summation of all values of table 5.8 column-wise is done to find the candidate attributes for fragmentation.

Cluster/Attribute	Birth-date	Salary	Location
C1	42	65	45
C2	106	126	122
C3	80	70	78
	$\sum 228$	$\sum 261$	$\sum 245$

Table 5.9: Attribute Retrieval and Update Table (ARUST) with summation of values

From the above table 5.9, it is found that Fragment Candidate Attribute (FCA) is "Salary", as it has the highest value.

c) Fragment generation

Now, candidate for fragmentation is "Salary" and there will be three fragments as follows:-

Fragment 1(F1) contains all those rows having Salary>1500

Fragment 2(F2) rows having Salary < 1500

Fragment 3(F3) rows having Salary =1500

d) Fragmentation

Based on above FCA and the conditions, the input relation given in the table 5.1 is fragmented.

Name	Birth-date	Job-Id	Salary	Location	Dept-ID
Mich	4/2/1980	MGR	2000	Riyadh	2
Nancy	12/1/1978	MAN	1750	Makah	2
Den	22/9/1989	MAN	2100	Riyadh	1

Table 5.10: Fragment 1(F1) having Salary >1500

Name	Birth-date	Job-Id	Salary	Location	Dept-ID
Jone	2/6/1984	MAN	1400	Jeddah	3
Math	2/2/1982	MAN	1150	Abha	4
Adam	4/2/1980	MGR	1300	Abha	2
Kevin	12/12/1979	MAN	1200	Jeddah	1

Table 5.11: Fragment 2(F2) having Salary <1500

Name	Birth-date	Job-Id	Salary	Location	Dept-ID
Jone	22/9/1990	MAN	1500	Riyadh	3
Zamel	22/9/1977	MAN	1500	Makah	1

Table 5.12: Fragment 3(F3) having Salary =1500

So, the horizontal fragmentation on input relation (table 5.1) is done successfully and three fragments, table 5.10, 5.11 and 5.12 respectively, are generated.

5.2.3 Allocation Solutions

As stated in 4.2.3, the constraints on sites are first examined. From table 5.2 constraints on network sites could be found. Since, clustering on the sites is already done, shown in table 5.4. From these two tables constraints on clusters are found.

Clusters	Sites within clusters	Fragment Limit(FL)
C1	S1	5
C2	S2,S4	1,3
C3	S3	3

Table 5.13: Clusters with Constraints

5.2.3.1 First Strategy of allocation

So using formula (4) and table 5.7 (CARUM) RUSM is obtained for fragments F1, F2 and F3.RUSM is shown below:-

Cluster/Fragments	F1	F2	F3
C1	26	28	11
C2	48	72	15
C3	20	30	20

T

Table 5.14: RUSM (Retrieval Update Sum Matrix)

Using formula (3) ,the Cost of Allocation Matrix (CAM) is obtained as follows:-

Cluster/Fragments	F1	F2	F3
C1	516	774	285
C2	816	1147	**447**
C3	**834**	**1152**	286.5

Table 5.15: Cost Allocation Matrix (CAM)

From the table 5.14, it is observed that cost of allocation for F1, F2 and F3 are high for cluster C3, C3 and C2 respectively. So considering constraints on clusters (table 5.13) and above values for cost of allocation, the process allocates the fragments, F1, F2 and F3 to C3 and C2 respectively. Same is listed in allocation table below:-

Cluster/Fragments	F1	F2	F3
C1	-	-	-
C2	-	-	Allocated
C3	Allocated	Allocated	-

Table 5.16: Allocation Table 1

From the above allocation table it could be found that no fragment is allocated to cluster C1.Since F3 is allocated to C2 it could either be allocated to S2 or S4. For that the process have to check if Average Retrieval Cost for this fragment is greater or less than Average Update cost on particular site as suggested in [1].Using CARUM matrix, in table 5.7 it is found that Average Retrieval Cost > Average Update Cost for fragment F3 for both S2 and S4 .Therefore, F3 can be allocated to any of site or replicated to both the sites. However, the replication is avoided in this research due to its effect on system performance. So, fragment F3 can be allocated to either Site S2 or S4. Both the cases are presented in the next section.

Performance Evaluation of Allocation using First Strategy of Allocation

Performance is evaluated on similar lines as in [29] and compared with[1]. Number of allocation is compared using first strategy of allocation with allocation in [1] as shown below:-

Case1:- In first strategy of allocation, if F3 is allocated to C2 and S2

#sites	#allocation without clustering in [1]	#allocation with clustering using first strategy of allocation	Improvement (%)
S1	2	0	+100%
S2	1	1	No change
S3	2	2	No change
S4	3	0	+100%
	Total allocation=8	Total allocation = 3	

Table 5.17: Comparison Table 1

Case2:- In first strategy of allocation, if F3 is allocated to C2 and S4

#sites	#allocation without clustering in [1]	#allocation with clustering using first strategy	Improvement (%)
S1	2	0	+100%
S2	1	0	+100%
S3	2	2	No change
S4	3	1	+66.66%
	Total allocation=8	Total allocation=3	

Table 5.18: Comparison Table 2

Therefore using formula proposed in [29], the system performance can be measured:-

$$\text{improved percentage} = \frac{\text{reduced number of allocation}}{\text{initial number of allocation}} \quad (6)$$

$$= (8-3)/8 = 0.625 = 62.5\%$$

So, the system performance is improved by 62.5%

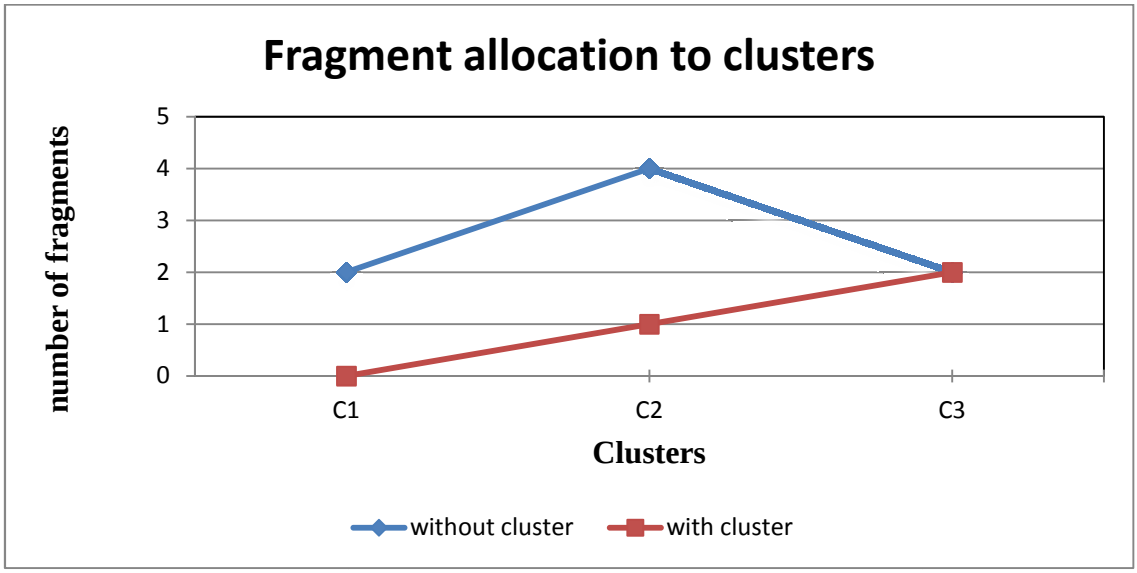


Chart 5.1: Fragment allocation to clusters using first strategy of allocation

Chart 5.1 is developed for cost based allocation. Clearly, it displays that using clustering the fragment allocation is less or equal to approach when sites are not clustered.

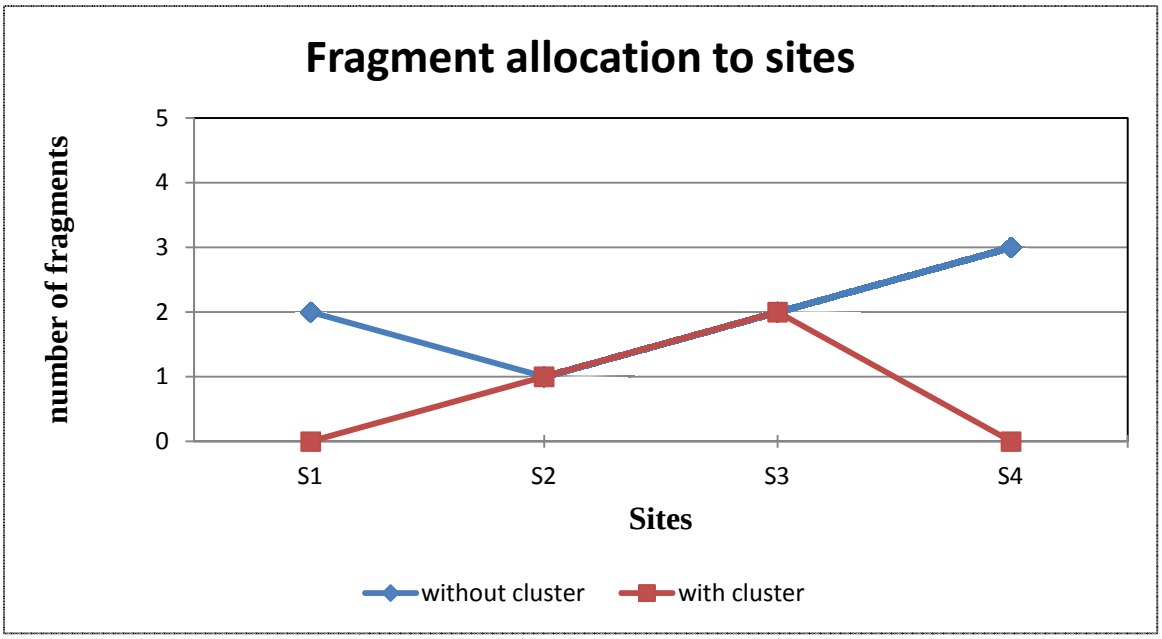


Chart 5.2: Fragment allocation to clusters using first strategy of allocation when fragment F3 allocated to S2 of C2

Since, Fragment F3 is allocated to cluster C2 and it can be allocated to any of sites S2 or S4 as found by its retrieval and Update frequencies. Chart 5.2 depicts this allocation of F3 to S2 and shows for site S2 allocation will be same with or without clusters.

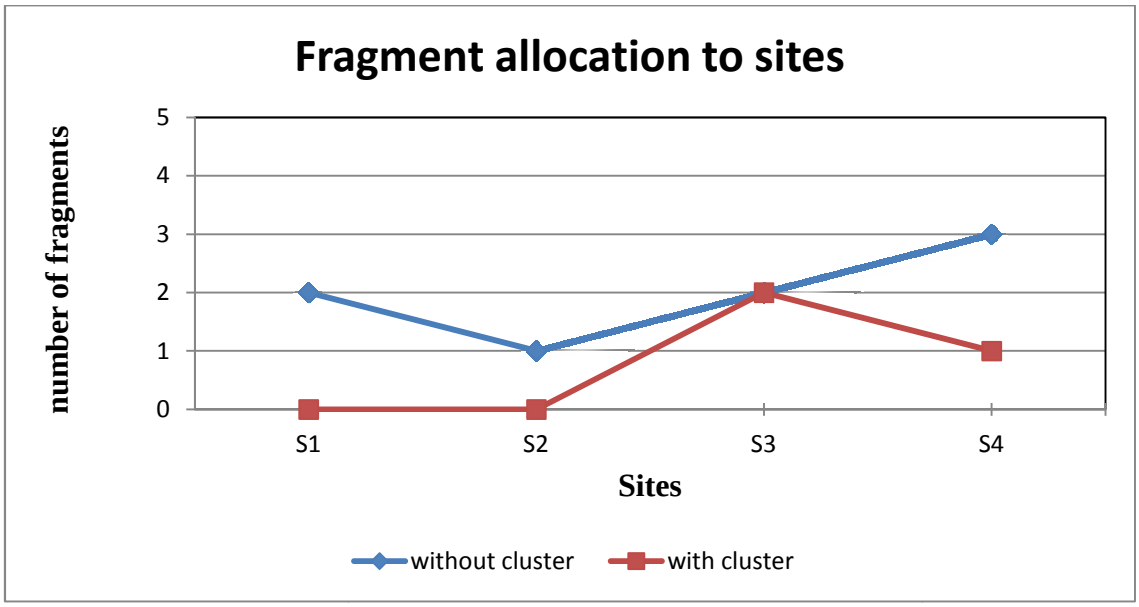


Chart 5.3: Fragment allocation to clusters using first strategy of allocation when fragment F3 allocated to S4 of C2

Chart 5.3 displays, F3 allocation for cluster C2, when F3 is allocated to site S4 of cluster 2.And, its clearly seen then a using clusters the fragment allocation is less or equal than the without cluster allocation. Thus, system performance is enhanced using clustering.

5.2.3.2 Second Strategy of allocation

In this strategy, the preference table for fragments is constructed using table 5.14(RUSM)

Cluster/Fragments	F1	F2	F3
C1	P2	P3	P3
C2	P1	P1	P2
C3	P3	P2	P1

Table 5.19: Preference Table

Where, P1, P2, P3 are preference number 1, 2, 3 respectively.

Based on preference table (5.19) above and optimal algorithm [16] an allocation table for second strategy of allocation is created as shown below:

Cluster/Fragments	F1	F2	F3
C1	-	-	-
C2	Allocated	Allocated	-
C3	-	-	Allocated

Table 5.20: Allocation Table 2

Performance Evaluation of Allocation using Second Strategy

Performance is evaluated as in [29] and compared with [1]. Using CARUM matrix in table 5.7 it is found that Average Retrieval Cost < Average Update Cost for S4 for both fragment F1&F2.Therefore, both fragments will be allocated to S4.Number of allocation is compared using second strategy with allocation in [1] as shown below:-

#sites	#allocation without clustering in [1]	#allocation with clustering using second strategy	Improvement (%)
S1	2	0	+100%
S2	1	0	+100%
S3	2	1	+50%
S4	3	2	+33.33%
	Total allocation=8	Total allocation = 3	

Table 5.21: Comparison Table 3

Using (6), system performance in this strategy is improved by 62.5% ,which is same as found using first strategy of allocation.

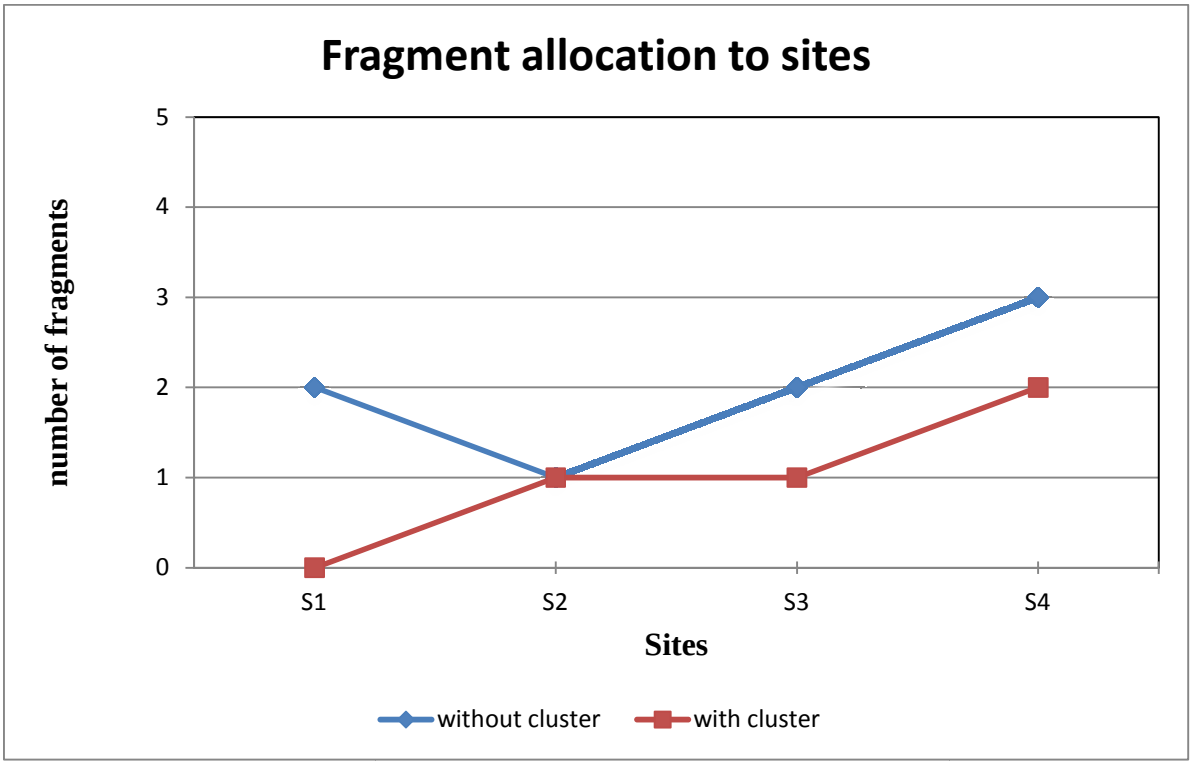


Chart 5.4: Fragment allocation to clusters using second strategy of allocation when fragment F1 &F2 allocated to S4 of C2

Chart 5.4 depicts the case when fragments F1 & F2 to be allocated to sites, S2 or S4, of cluster C2. It also, depicts using the strategy based on preference; the allocation is less in case of clustering.

5.2.3.3 Third Strategy of allocation

The frequent access for the fragment is determined by the preference table (5.19) and nearest neighbor is determined by Cluster Communication Cost Matrix (CCCM) (table 5.5). The process will construct allocation table referring these two tables. For instance, if F1is to be allocated to C2 (from table 5.19) , so nearest neighbor to C2 (table 5.5) is searched and if it is C1 then fragment F1 is allocated to C1 and after number of allocation exceeds some threshold value fragment will be moved to next nearest neighbor until it reaches at destination site.

So the fragment is basically moved along path from source to the site having high access frequency for it. So, in the case above, first the fragment F1 will be allocated to C1 and then moved to C2 (actual destination) if required. The allocation table for the third strategy is shown below:-

Cluster/Fragments	F1	F2	F3
C1	Allocated	Allocated	Allocated
C2	-	-	-
C3	-	-	-

Table 5.22: Allocation Table 3

As seen in table (5.21) all the fragments are allocated to C1 and not to any other cluster which is not the good way of allocation.

Performance Evaluation of Allocation using Third Strategy

Performance is evaluated as described in [29] and compared with [1]. Number of allocation is compared using third strategy with allocation in [1] as shown below:-

#sites	#allocation without clustering in [1]	#allocation with clustering using third strategy	Improvement (%)
S1	2	3	-50%
S2	1	0	+100%
S3	2	0	+100%
S4	3	0	+100%
	Total allocation=8	Total allocation = 3	

Table 5.23: Comparison Table 4

Using (6), system performance is found to be improved by 62.5% for third strategy of allocation.

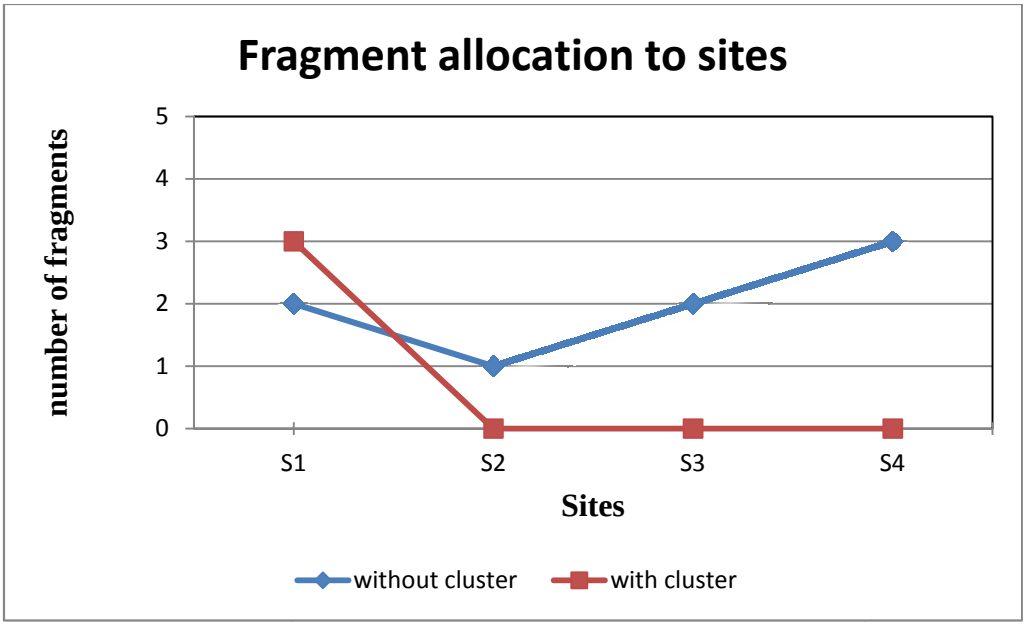


Chart 5.5: Fragment allocation to clusters using third strategy of allocation

Chart 5.5 shows that using third strategy and clustering, all the fragments are allocated to only one site, S1 which is not a good practice.

5.2.4 Allocation

Allocation using Fragment Evolutionary Allocation Algorithm (FEAA):-

Iteration 1

a) Initial Population

The first step is to create a random initial population from the allocation solutions obtained and encode them in binary strings. The three set of initial population is obtained from three strategies above encoded in binary string.

For e.g.:- if in first strategy the F1,F2 and F3 is allocated to clusters C3,C3 and C2 respectively ,so the cluster representation would be C3C3C2. And, if fragment is allocated to C3 then its binary encoding would be 11. Therefore, the initial population would be as follows:-

Solution #	Cluster Representation	Binary Encoding
S1	C3C3C2	111110
S2	C2C2C3	101011
S3	C1C1C1	010101

Table 5.24: Initial population Table

Where, S1, S2 and S3 are Solution numbers obtained from three strategies explained above.

b) Fitness Value

In the next step the process has to evaluate the fitness value for each individual in the current population. The fitness value is found from cost of allocation of fragment to clusters, as suggested in [29].The fitness function is represented as f() and evaluated for particular solution by adding cost of allocating each fragment to cluster (using table 5.15). Fitness table showing f() for each solution is described below:-

Solution #	Binary Encoding	Fitness function f()
S1	111110	f(S1)=2433
S2	101011	f(S2)=2244.5
S3	010101	f(S3)=1575
		$\sum 6252.5$

Table 5.25: Fitness Table

c) Selection of Parents

Selection mechanism roulette wheel is used for selection of parents. The objective is to "select the best and discard the rest" [14].The parents could be any combination of S1, S2 and S3 i.e. 3 X 3=9 but the rule will keep only those combinations of solutions which have higher fitness value (table 5.25) as in compliance with objective. So,t he process obtained parents from the combination of S1 and S2 i.e. 2 X 2 = 4 .Candidates for parents are (S1, S2), (S2, S1), (S1, S1) and (S2, S2) .Out of all the candidates, only (S1, S2) are valid candidate. Since (S2, S1) is same as (S1, S2) and the children cannot be generated from single parents, so discard (S1, S1) and (S2, S2).So, the parents will be:

S1=111110 f (S1) = 2433
S2=101011 f (S2) = 2244.5
∑4677.5

d) Crossover

Uniform crossover method is used. In uniform crossover the selection point is in the middle. Since, each parent with binary string length 6, therefore the selection point would be 3 for each string. Crossover is done as follows:-

S1 = 1 1 1 | 1 1 0

S2 = 1 0 1 | 0 1 1

The selection point is 3 i.e. keep the bits same before it and invert the bits beyond this point to obtain children C1 and C2.

C1 = 1 1 1 0 1 1 f (C1) = 2267.5

C2 = 1 0 1 1 1 0 f (C2) = 2415

∑4682.5

It could be seen in one generation only the total population fitness is improved from 4677.5 to 4682.5, thus improved ~ 0.1%

But from the above, C1 found from S1 has reduced its fitness, so the mutation is applied over it to restore lost information.

e) Mutation

Mutation will help in restoring lost information. Mutation rate is kept very small. Since fitness value of C1 derived from S1 is reduced after crossover. Therefore, C1 is subject to mutation. Bit flap method is used for mutation. Only one bit can be flapped at a time. Mutation operator will just flip the bit i.e. 0 for 1 and vice-a –versa.

C1 = 1 1 1 0 1 1 f (C1) = 2267.5

Several candidate bits are available for mutation. The process will consider some cases and if fitness value is improved then the new candidate after mutation can be considered otherwise rejected. The candidate having highest fitness will be considered.

The mutation table is shown below:-

Case #	Bit number to be flapped	New child generated after Mutation (C1')	Fitness value	Consideration
1	1	011011	1949.5	Rejected
2	2	101011	2249	Rejected
3	4	111111	2272	Considered
4	5	111001	2266	Rejected
5	6	111010	2428	Considered

Table 5.26: Mutation Table 1

From the mutation table 5.26, it is found, case 3 and case 5 can be considered but clearly case 5 has high fitness value and is fitter. Therefore, case 5 will be considered and case 3 will be rejected.

C1' = 1 1 1 0 1 0 f (C1') = 2428
C2 = 1 0 1 1 1 0 f (C2) = 2415
∑4843

After mutation the total population fitness is further improved from 4677.5 to 4843, thus improved ~ 4%.

At this iteration highest fitness value child is C1'.

Again these children would be considered parents for new generation and the steps c to e of algorithm will be repeated till the number of generation < maximum generationor no new generation can be produced further.

Iteration 2

a) Parents

The parents for this generation would be children of iteration 1 i.e. C1' and C2.

S1 = 1 1 1 0 1 0 f (S1) = 2428
S2 = 1 0 1 1 1 0 f (S2) = 2415
∑4843

b) Crossover

Again, uniform crossover method is used. The selection point is 3 i.e. keep the bits same before it and invert the bits beyond this point to obtain children C1 and C2. Crossover is done as follows:-

S1 = 1 1 1 | 0 1 0

S2 = 1 0 1 | 1 1 0

C1 = 1 1 1 1 1 0 f (C1) = 2433
C2 = 1 0 1 0 1 0 f (C2) = 2410
∑4843

No improvement in fitness value found after crossover. But the fitness value of C2 decreased so C2 will be mutated.

c) Mutation

Since fitness value of C2 derived from S2 is reduced after crossover. Therefore, C2 is subject to mutation. Bit flap method is used for mutation. Only one bit can be flapped at a time. Mutation operator will just flip the bit i.e. 0 for 1 and vice-a –versa.

C2 = 1 0 1 0 1 0 f (C2) = 2410

Several candidate bits are available for mutation. The process will consider some cases and if fitness value is improved then the new candidate after mutation can be considered otherwise rejected. The candidate having highest fitness will be considered. The mutation table is shown below:-

Case #	Bit number to be flapped	New child generated after Mutation (C1')	Fitness value	Consideration
1	2	111010	2428	Considered
2	4	101110	2415	Considered
3	6	101011	2249	Considered

Table 5.27: Mutation Table 2

From the mutation table 5.25, it could be found that case 1 has high fitness value and is fitter. Therefore, case 1 will be considered and rest cases.

C1 = 1 1 1 1 1 0 f (C1) = 2433
C2' = 1 1 1 0 1 0 f (C2') =2428
∑4862

After mutation the total population fitness is improved from 4843 to 4862, thus improved ~ 0.4%.

At this iteration highest fitness value child is C1.

Again these children would be considered parents for new generation and the steps c to e of algorithm will be repeated.

Iteration 3

a) Parents

The parents for this generation would be children of iteration 2 i.e. C1 and C2'.

S1 = 1 1 1 1 1 0 f (S1) = 2433
S2 = 1 1 1 0 1 0 f (S2) = 2428
∑4862

b) Crossover

Again, the process will use uniform crossover method. The selection point is 3 i.e. keep the bits same before it and invert the bits beyond this point to obtain children C1 and C2. Crossover is done as follows:-

S1 = 1 1 1 | 1 1 0

S2 = 1 1 1 | 0 1 0

C1 = 1 1 1 0 1 0 f (C1) = 2428
C2 = 1 1 1 1 1 0 f (C2) = 2433
∑4862

No improvement in fitness value found after crossover. **And, the process finds that C1 and C2 are just reverse of parents. So no new generation is formed and the process will stop here.**

The process would retrieve the best solutions (higher fitness child) from Iteration 1 and Iteration 2, as shown below:-

Iteration #	Highest fitness value children
1	C1' = 1 1 1 0 1 0 f (C1') = 2428
2	C1 = 1 1 1 1 1 0 f (C1) = 2433

Table 5.28: Best Solution Table

The best solution is one with higher fitness value. From the table above, the process would analyze that the solution having fitness value 2433 is best of both. But this is one of the parents and obtained from first strategy. Considering all the three strategies while obtaining the solution for allocating fragments to clusters, then the process will not choose the solution obtained from any single strategy. The process should find the average best solution. Therefore, the process will consider the solution having fitness value 2428.

So, the solution found using FEAA for fragment allocation is:-

Solution of allocation = C1' = 1 1 1 0 1 0 = C3 C2 C2

Fragment/Cluster	F1	F2	F3
C1	-	-	-
C2	-	Allocated	Allocated
C3	Allocated	-	-

Table 5.29: Final Allocation Table

Performance Evaluation of Allocation using FEAA

The fragment evolutionary allocation algorithm generates a solution C3C2C2 for allocation having fitness value 2428.

#Strategy	#Solution cluster representation	Binary Encoding	Fitness Value	Improvement Using FEAA compared to #Strategy
1	C3C3C2	111110	2433	~ -0.2%
2	C2C2C3	101011	2249.5	~ +7.3%
3	C1C1C1	010101	1575	~ +54%

Table 5.30: Comparison Table of various strategies of allocation with FEAA

From the above table 5.30, it can be seen that if allocation is done using FEAA strategy solution then the system performance is 7.3% & 54% better than second strategy solution and third strategy solution respectively. However, performance is only 0.2% less when compared to first strategy. But first strategy is purely cost based while FEAA tries to find average best solution. Therefore, with negligence of 0.2%, FEAA strategy solution is considered near to the optimal solution for allocation.

5.3 Discussion

This chapter presents the result obtained by simulating the problem for dataset [1]. It is observed all the phases are developed. The proposed evolutionary algorithm, FEAA, produces near to optimal solution.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

In this research project the proposed work is analyzed in two domains of distributed database management system i.e. fragmentation and allocation. The problem of horizontal fragmentation and allocation of distributed databases is studied collectively on dataset[1]. The research project combines clustering of sites with horizontal fragmentation and allocation. Clustering of sites is done on the methodology described in[17]. Clustering further minimizes the communication cost and reduces the complexity of peer-to-peer connection of sites. In this work clustering of sites is done at initial phase. Cluster Communication Cost matrix (CCCM) is generated and used in the project. As required in the problem, all the phases are considered sequentially. The horizontal fragmentation methodology used has analogy to [1]. For horizontal fragmentation a cluster attribute retrieval and update matrix (CARUM) is generated from retrieval and update frequencies. The process takes solution from various allocation strategies, based on cost model, preference and nearest neighborhood allocation (NNA). These solutions served as initial population for fragment evolutionary allocation algorithm (FEAA) developed. The results developed a new strategy of allocation. The proposed work has contribution in formulating the new strategy using evolutionary algorithm for allocation. In the proposed work much attention has been provided to allocate the fragments to clusters. A proper allocation of fragments plays a major role in saving cost of query execution. The whole process of clustering, fragmentation, formulating allocation strategies and final allocation is taken together for a dataset proposed in[1]. The process is simulated using java. With the assumption to find average best solution or near to the optimal solution, the developed model appears to be promising.

6.2 FUTURE SCOPE

The research can be extended to use vertical fragmentation for obtaining solution. The cost of space can be considered when using cost model for allocating fragments. The research can also incorporate replication. While doing selection in evolutionary algorithm proposed, other strategies like tournament, rank selection can also be tested. For crossover other strategies like single point crossover or two point crossover can also be used. The research can be more

extensively tested for different crossover and mutation rate. Further large datasets can be used. This research can further be extended to the next area of query processing.

REFERENCES:

[1] Abdalla, H. I., and Amer, A. A., "Dynamic Horizontal Fragmentation, Replication and Allocation Model In DDBSs," 2012 International Conference on Information Technology and e-Services, IEEE, 2012.

[2] Abdalla, H., et al., "Using a Greedy-Based Approach for Solving Data Allocation Problem in a Distributed Environment," International Conference on Parallel and Distributed Processing Techniques and. Applications (PDPTA'08), 2008.

[3] Ahmad, I., and Karlapalem, K., "Evolutionary Algorithms for Allocating Data in Distributed Database Systems," Distributed and Parallel Databases, pp. 5–32, 2002.

[4] Amer, A. A., and Abdalla, H. I., "An Integrated Design Scheme for Performance Optimization in Distributed Environments," 2012 International Conference Education and e-Learning Innovations, IEEE, 2012.

[5] Apers, P. M.G, "Data allocation in distributed database systems," ACM Transactions on Database Systems, vol. 13,no. 3, pp.263-304,1988.

[6] Bai˜ao, F., Mattoso, M., and Zaverucha, G., "A distribution design methodology for object DBMS," Distributed and Parallel Databases, Springer, Vol. 16, No. 1, pp. 45-90,2004.

[7] Ceri, S., and Pelagatti, G., "Distributed Databases Principles and System," McGraw-Hill, New York, 1984.

[8] Chang, S.-K., "Database decomposition in a hierarchical computer system," In Proceedings of the 1975 ACM SIGMOD international conference on Management of data, ACM Press, pp.48-53,1975.

[9] Cheng, C.-H., Lee, W. K., and Wong, K. F., "A genetic algorithm-based clustering approach for database partitioning," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 32, No. 3,pp. 215–230, 2002.

[10] CHU, W.W., "Optimal file allocation in a multiple computer system," IEEE Transactions on Cheng C. H Computers, vol.18, no.10, pp. 885-889, 1969.

[11] Corcoran, A.L., and Hale, J., "A genetic algorithm for fragment allocation in a distributed database system," In Proceedings of the 1994 ACM symposium on Applied computing (SAC) , ACM Press, pp.247-250,1994.

[12] Cornell, D. W., and YU, P.S., "On optimal site assignment for relations in distributed database environment," IEEE Transactions on Software Engineering, vol. 15, no. 8, pp.1004-1009, 1989.

[13] Date, J.C., "An introduction to Database Systems," 8th ed., Addison Wesley, 2003.

[14] Eiben, E.A. and Smith, E.J., "Introduction to Evolutionary Computing Genetic Algorithms," Springer, Natural Computing Series, 2003.

[15] Ezeife, C.I. and Barker, K., "Distributed Object Based Design: Vertical Fragmentation of classes," International Journal of Distributed and Parallel Databases, Kluwer Academic Publishers ,Vol. 6, No. 4, October 1998.

[16] Gope, Dejan Chandra, "Dynamic Data Allocation Methods in Distributed Database System," American Academic & Scholarly Research Journal, Vol. 4, No. 6, pp.1-8, Nov. 2012.

[17] Hababeh I. O, Bowring N., "A Method for Fragment Allocation Design in the Distributed Database Systems," UGRU-4, The Sixth Annual U.A.E University Research Conference, 2003.

[18] Huang, Y. - F., and Chen, J.-H., "Fragment allocation in distributed database design," Information Science and Engineering, vol. 17, pp.491-506, 2001.

[19] Iacob Nicoleta - Magdalena, "Fragmentation and Data Allocation in the Distributed Environments," Annals of the University of Craiova, Mathematics and Computer Science Series, Volume 38(3), pp. 76-83, 2011.

[20] Karlapalem, K., Navathe, S. B., and Morsi, M. M. A., "Issues in Distribution Design of Object- Oriented Databases," In IWDOM (1992), pp.148-164, 1992.

[21] Karlapalem, K., and Pun, N.M., "Query-driven data allocation algorithms for distributed database system," In Database and Expert Systems Applications, pp.347-356, 1997.

[22] Kosmann, D., "The State of the Art in Distributed Query Processing," ACM Computing Surveys (CSUR), vol.32, no. 4, pp.422-469, 2000.

[23] Mahboubi, H., and Darmont, J., "Enhancing XML Data Warehouse Query Performance by Fragmentation," in Proc.ACM SAC09, pp.1555-1565, 2009.

[24] Mei, A., Mancini, L., and Jajodia, S., "Secure Dynamic Fragment and Replica Allocation in Large-Scale Distributed File Systems," IEEE Trans. Parallel and Distributed Systems, vol. 14, no. 9, Sept 2003.

[25] Menon, M.-S., "Allocating fragments in distributed databases," IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 7, pp.577-585, 2005.

[26] Navathe, S., Karlapalem, K., and Ra, M., "A mixed fragmentation methodology for initial distributed database design," Journal of Computer and Software Engineering, vol. 3, no. 4, pp. 395–426, 1995.

[27] Özsu M.Tamer and Valdureiz Patrick, "Principles of distributed database System," Third ed., Prentice-Hall, New Jersey, 1999.

[28] Ra, M., "Horizontal partitioning for distributed database design," Advances in Database Research, World Scientific Publishing, pp. 101–120, 1993.

[29] Rahmani, S., Torkzaban, V., and Haghighat, A. T., "A New Method of Genetic Algorithm for Data Allocation in Distributed Database Systems," 2009 First International Workshop on Education Technology and Computer Science, IEEE, 2009.

[30] Ramakrishnan, R., and Gehrke, J., "Database Management Systems," McGraw- Hill Boston, 1998.

[31] Rothne, J.B., and Goodman, N., "An overview of the preliminary design of sdd-1: A system for distributed databases," in Berkeley Workshop (1977), pp.39-57, 1977.

[32] Sacca, D., and Wiederhold, G., "Database partitioning in a cluster of processors," ACM Transactions on Database Systems (TODS), vol. 10, no. 1, pp.29-56, 1985.

[33] Sarathy, R., Shetty, B. and Sen, A., "A Constrained Nonlinear 0-1 Program for Data Allocation," European J. Operational Research, vol. 102, pp. 626-647, 1997.

[34] Tamhankar, A. and Ram, S., "Database Fragmentation and Allocation: An Integrated Methodology and Case Study," IEEE Trans. Systems, Man and Cybernetics—Part A, vol. 28, no. 3, pp.194-207, May 1998.

[35] Zhang, Y., "On horizontal fragmentation of distributed database design," in Advances in Database Research (1993), M. Orlowska and M. Papazoglou, Eds. , World Scientific Publishing, pp. 121-130, 1993.

ANNEXURE A: CODING

Here is a glimpse of coding done to implement the project .The code is being developed in java and database connectivity is done to fulfill the need of project. Software NetBeans IDE 7.2.1 is used to develop the code in java.

```
package clusteredfragmentation;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Scanner;

public class Clusteredfragmentation {

public static int clusters(int site)
  {
     Scanner console= new Scanner(System.in);
      int k,p,count=0,clusterindex=0,ccr=0,scount=0;
     int [][]cost= new int[site+1][site+1];
     int [][]cluster=new int[site+1][site+1];
     int [][]ccm = new int[site+1][];
    float [][]ccc=new float[10][10];
     //Finding distance cost of sites
```

```
        for(int i=1;i<=site;i++)
         for(int j=1;j<=site;j++)
         {
            System.out.println("Enter the distance cost of site"+i+j);
            cost[i][j]=console.nextInt();
         }


       //Creating distance cost matrix
     System.out.println();
     System.out.println("Distance cost matrix of sites:");
     System.out.print("sites");
      for(int i=1;i<=site;i++)
      {
         System.out.print("\tsite"+i+"\t");
      }

      System.out.println();
      for(int i=1;i<=site;i++)
      {   System.out.print("site"+i+"\t");
         for(int j=1;j<=site;j++)
         {
           System.out.print(cost[i][j]+"\t\t");
         }
         System.out.println();
    }
//clustering sites
  System.out.println("Enter the number of communication units"); // allowed to communicate
     ccr=console.nextInt();


     for(int i=1;i<=site;i++)
     {
        count+=1;
```

```
for(int j=1;j<=site;j++)
{
   if(cost[i][j] <= ccr)
   {
   cluster[i][j]=1;
   clusterindex=j;
  }

   else
   { cluster[i][j]=0;}

}

k=clusterindex;
p=i+1;
if(k >= p && p <= site)
{
   count-=1;

 }

}


//System.out.println(count);
 // creating cluster sites matrix

 System.out.println();
 System.out.println("Clustering sites:");
 System.out.print("clusters/sites");
  for(int i=1;i<=site;i++)
  {System.out.print("\tsite"+i+"\t");
  }

  System.out.println();


 for(int i=1;i<=count;i++)
  {   System.out.print("cluster"+i+"\t");
     for( int l=1;l<=site;l++)
     {

              System.out.print(cluster[i][l]+"\t\t");

      }            System.out.println();}
```

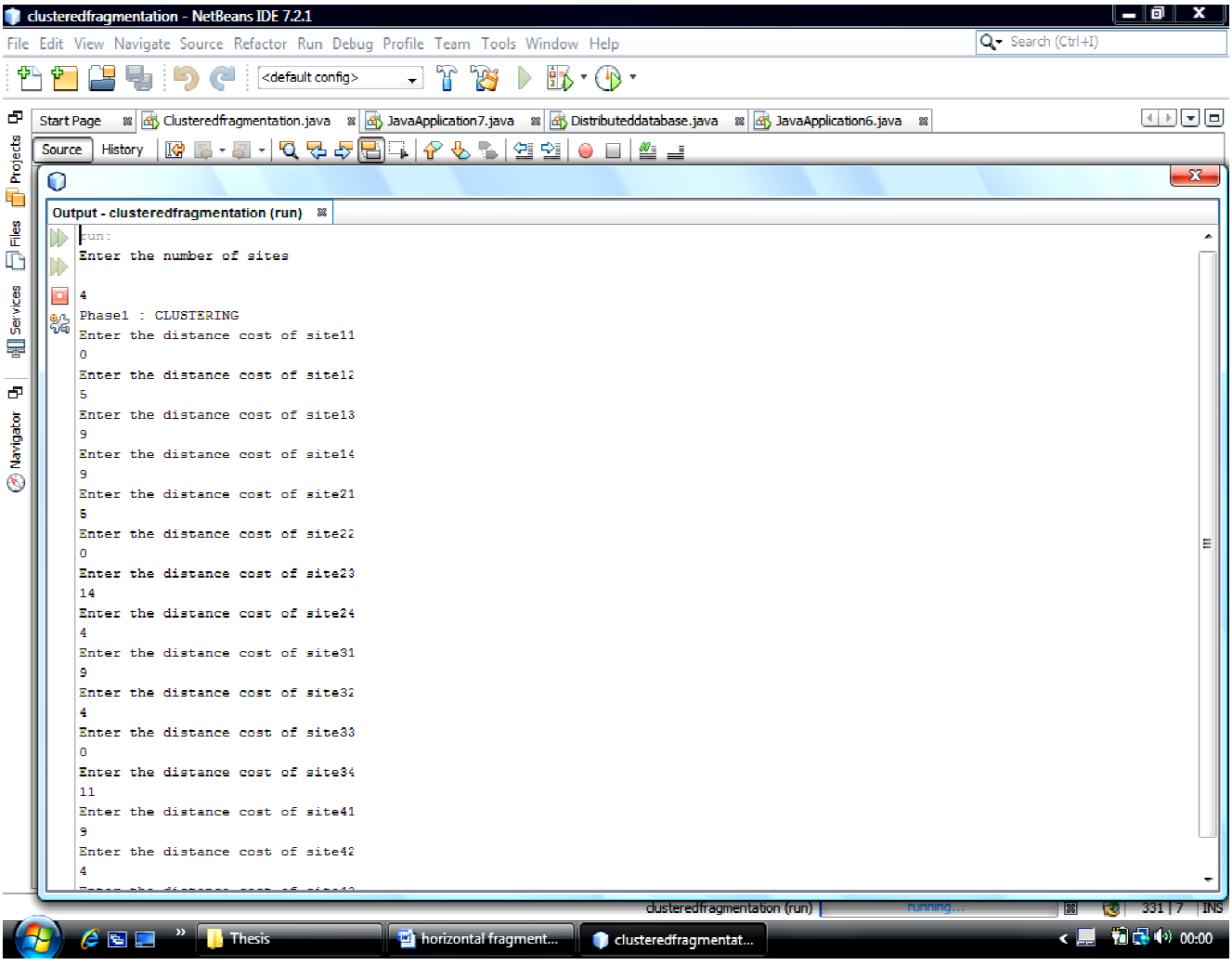
```
  //calculating cluster communication
 for(int s=1;s<=count;s++)
 {
    for(int i=1;i<=count;i++)
   {
    for(int j=1;j<=site;j++)
    {
       if(s == i)
       {

          if(cluster[i][j]== 1)
      {
         if(i==j)
         {
            ccc[s][i]=0;
        //     System.out.print("cluster"+s+i+""+ccc[s][i]);
         }
         else
         {

       ccc[s][i]+=(cost[i][j]+cost[j][i])/1;
      // System.out.print("cluster"+s+i+""+ccc[s][i]);
         }
      }
       }
       else
        {
         if(cluster[i][j]==1 || (cluster[s][j]==1 && s!=j))
         {
          scount+=2;

            if(cluster[s][j]==1 && s!=j)
        {
           ccc[s][i]+=(cost[i][j]+cost[j][i]);
        }
           else
           {
              ccc[s][i]+=(cost[s][j]+cost[j][s]);
           }}}}
    if(s!=i)
    {
          ccc[s][i]=ccc[s][i]/scount;

//System.out.print("cluster"+s+i+"\t\t"+ccc[s][i]+"\t\t");
scount=0;  System.out.println();}
```

```
    // cluster communication cost matrix

  System.out.println();

    System.out.println("Cluster communication matrix:");
    System.out.print("clusters#");
     for(int s=1;s<=count;s++)
     {System.out.print("\tcluster"+s+"\t");
     }
     System.out.println();
    for(int s=1;s<=count;s++)
     {   System.out.print("cluster"+s+"\t");
         for(int i=1;i<=count;i++)
         {
             System.out.print(ccc[s][i]+"\t\t\t");

         }
         System.out.println();
   }
      return count;
  }
```

ANNEXURE B: SCREENSHOTS

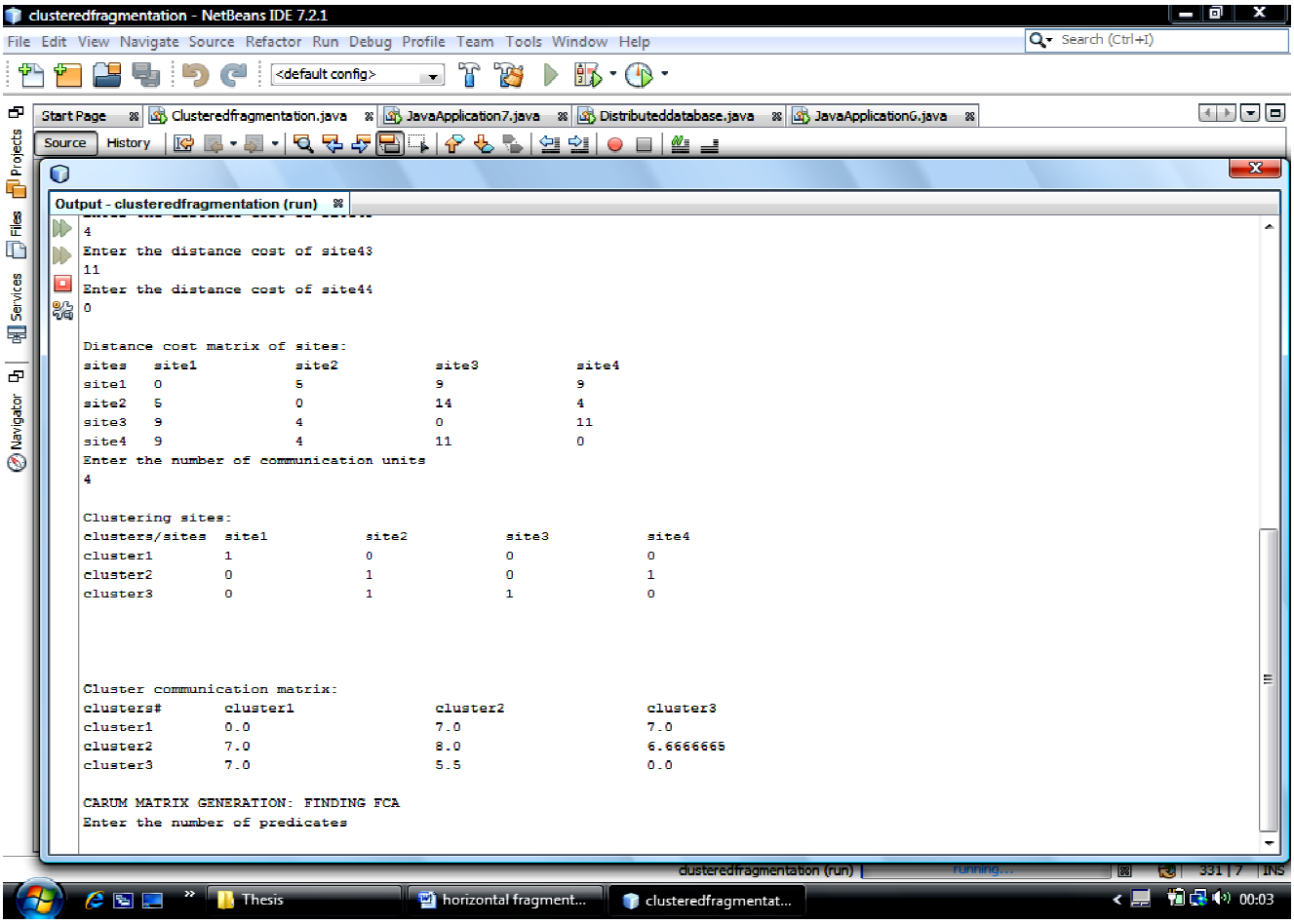
In this ANNEXURE B, some screenshots are presented to show working of the project.

SCREENSHOT 1:



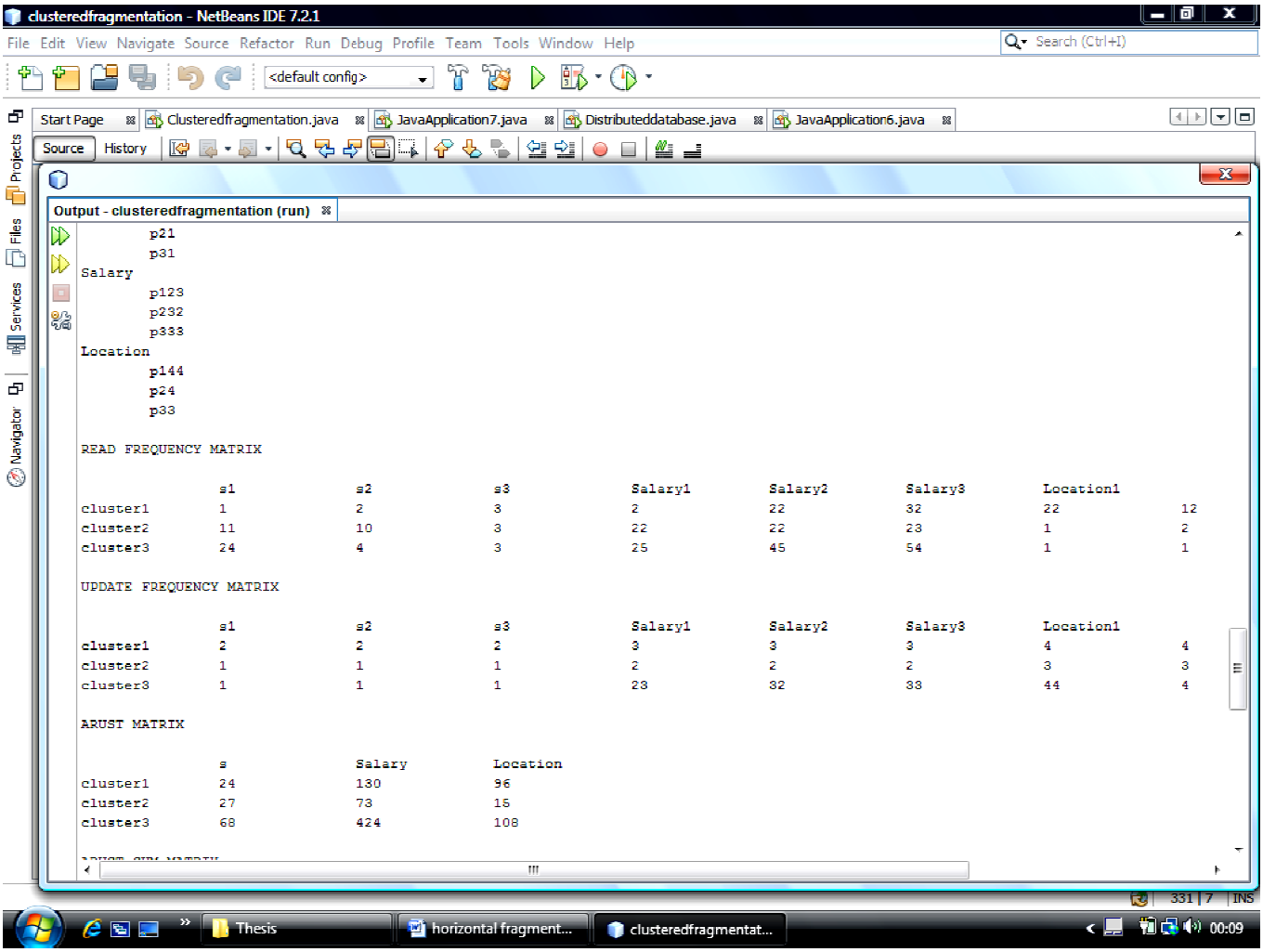
This screenshot displays, the code finds the number of sites from users and distance of respective sites.

SCREENSHOT 2:



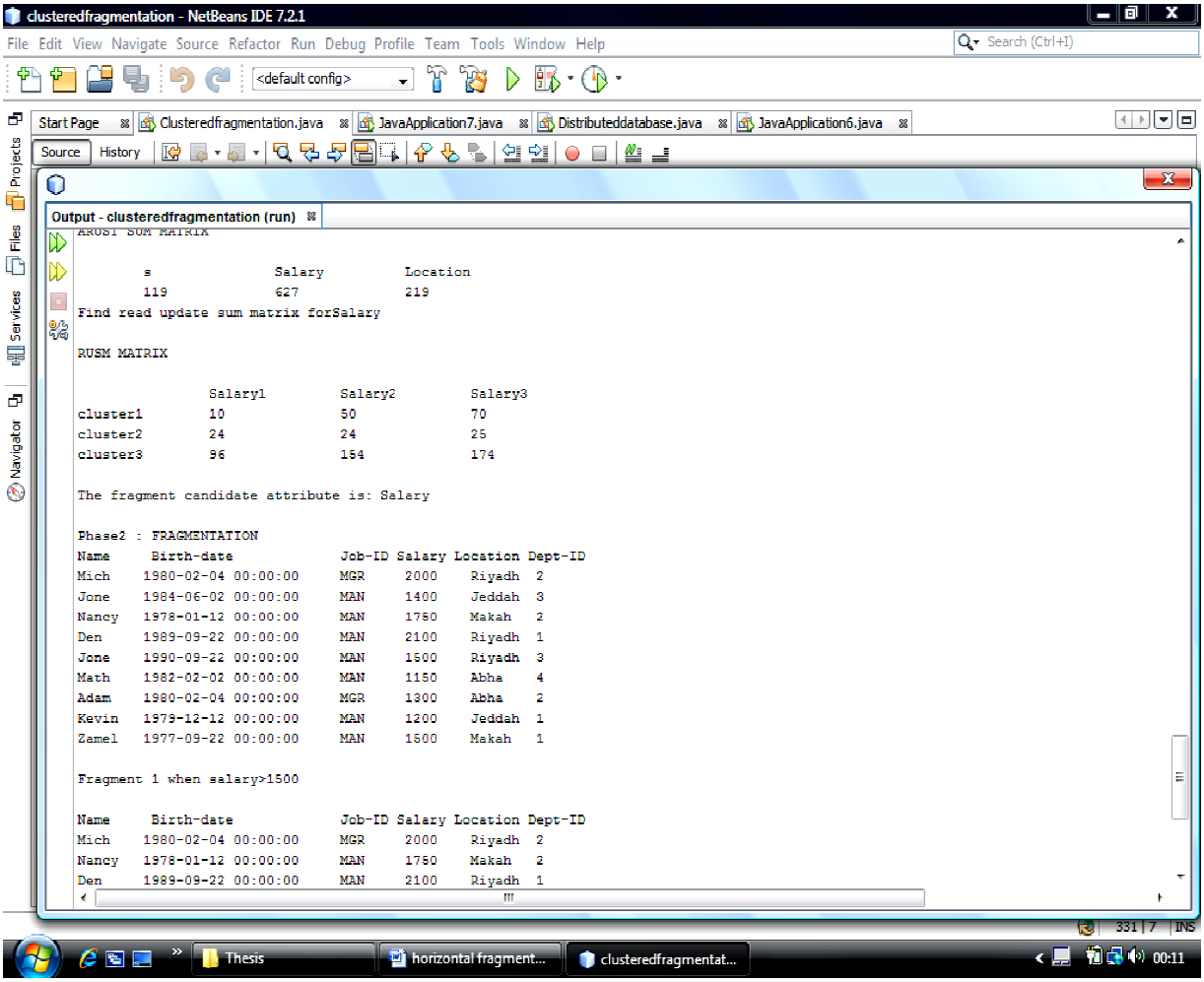
The screenshot displaying the distance cost matrix and clusters generated. Also, it shows the generation of cluster communication matrix.

SCREENSHOT 3:



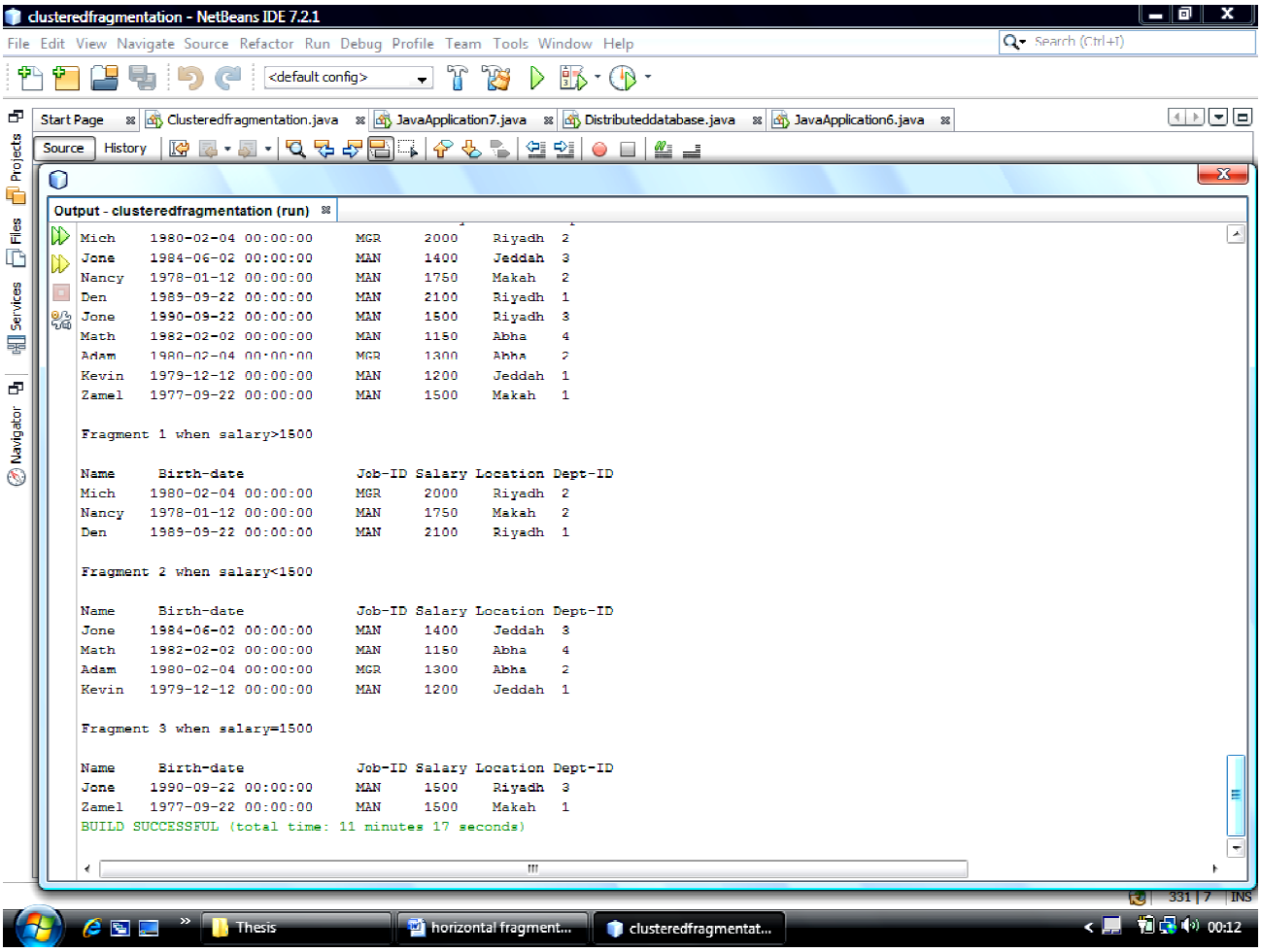
This screenshot displays the read frequency matrix, update frequency matrix and ARUST matrix evaluated.

SCREENSHOT 4:



This screenshot displays the RUSM matrix evaluated in the code and then it finds fragment candidate attribute.

SCREENSHOT 5:



This screenshot displays the various fragment generated for fragment candidate attribute.