REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DYNAMIC DATA REPLICATION AND DISTRIBUTION IN DATABASE SYSTEMS

Saadi Hamad THALIJ ALLUHAIBI

DOCTOR OF PHILOSOPHY THESIS

Department of Computer Engineering

Computer Engineering Program

Advisor

Assoc.Prof. Dr. Veli HAKKOYMAZ

July, 2019

REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DYNAMIC DATA REPLICATION AND DISTRIBUTION IN DATABASE SYSTEM

A thesis submitted by Saadi Hamad THALIJ ALLUHAIBI in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY is approved by the committee on 16.07.2019 in Department of Computer Engineering, Computer Engineering Program.

Assoc.Prof.Dr. Veli HAKKOYMAZ

Yildiz Technical University

Advisor

Approved By the Examining Committee

Assoc.Prof.Dr. Veli HAKKOYMAZ, Advisor

Yildiz Technical University

Asst.Prof.Dr.Mustafa Utku KALAY

Yildiz Technical University

Prof. Dr.Nizamettin AYDIN

Yildiz Technical University

Asst.Prof.Dr. Selçuk SEVGEN

Istanbul University

Assoc.Prof.Dr. Musaria K. MAHMOOD

Istanbul Gelişim University

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Location Analysis of The Emergency Service Centers Of a Case Company supervised by my supervisor, Assoc.Prof. Veli HAKKOYMAZ. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Saadi Hamad THALIJ ALLUHAIBI

Signature

Dedicated to my family

and my best friends

ACKNOWLEDGEMENTS

I gratefully acknowledge the support and generosity of The Yildiz Technical University, without which the present study could not have been completed.

Thank you to my supervisor, Dr. Veli HAKKOYMAZ, for providing guidance and feedback throughout this work. Thanks also to all teachers in department of computer engineering and all employees in Yildiz Technical University, for helps me in study times.

Yildiz Technical University that far have been amazing. I have learned and experienced so many new things in such a last five years, and it has gone by so fast. I am still discovering all the opportunities that university offers me. Turkey country is a very welcoming place, and from the first day, I felt I'm in at my second country.

I have made many new friends from many different backgrounds. There are many students in YTU from out of turkey with me from Azerbaijan, Iran and various Arab countries and that has allowed me an insight into their cultures.

Saadi Hamad THALIJ ALLUHAIBI

$C_{io}+C_d$	Cost incurred for sending query and receiving response
A_d	Allocation pattern of data d
C_c	Cost incurred for sending query from client to sub-server
$R_d^{p_i}(k)$	servicing a read request
$W_d^{q}(k)$	servicing a write request
f_i	Fitness value of i-th chicken
f_k	Fitness value of k, a rooster's index
f_{r1}	Fitness value of r1, an index of the rooster
f_{r2}	Fitness value of r2, an index of the chicken
$l_i(t)$	Luciferin level associated with glow-worm *i* at time *t*
p_i	i-th processor
r_d^i	Neighborhood range
r_s	Neighborhood range with respect to steps size
x_t	Parameter used to control the number of neighbors
σ^2	Standard deviation
$\in$	is an element of
$\notin$	is not an element of
$\leq$	Less than or equal to
∞	Infinity
d'	Data of the read request
exp	Exponential
β	Constant parameter

γ	Luciferin enhancement constant
δ	Time slots
Φ	Element vector
FL	Food forage parameter of chicken
$J(x_i(t))$	Value of the objective function at glow-worm *i*'s location at time t
$O(k)$	Replicated data object k
$Rand$	Uniform random number over [0, 1]
$S(d)$	Fixed processor set
$S1$	Search patter of rooster with index r1
$TOC(k)$	Total operation cost of k
abs	Absolute value
$inv\_list(p, d)$	list maintaining the set of processors
$t(1 \leq t \leq n)$	Number of replicas defined between 1 to n
$win(d)$	Request window for a data d
α	Control parameter for the range of disturbance
ε	Smallest constant to avoid zero-division-error
ρ	Luciferin decay constant
τ	Window size at specified time units

LIST OF ABBREVIATIONS

3-LHA	3-Level Hierarchical Algorithm
ACID	Atomicity, Consistence, Isolation and Durability
ACO	Ant Colony Optimization
ANSI	American National Standards Institute
BBO	Biogeography Based Optimization
BHR	Bandwidth Hierarchy based Replication
BHR	Bandwidth Hierarchy based Replication
CAP	Consistency, Availability and Partition
CCA	Computer Corporation of America
CPU	Central Processing Unit
CRUD	Create, Read, Update and Delete
CSO	Chicken Swarm Optimization
DAC	Discretionary Access Control
DAP	Data Allocation Problem
DARE	Distributed Adaptive data Replication
DBAs	Database Administrators
DBMS	Database Management Systems
DDB	Distributed Database
DDBMS	Distributed Database Management Systems
DoS	Denial of Service
DYFRAM	Dynamic Fragmentation and Replica Management Model
EDDA	Efficient Distributed Data Replication
FAP	File Allocation Problem
FIFO	First in First Out
FPE	Final Prediction Error
FP-MAX	Frequent Patterns Maximum
GA	Genetic Algorithm
GB	Gigabyte
GSO	Glowworm swarm Optimization
Gwa	Glow worm agent
HAC	Hierarchical Agglomerative Clustering
I/O	Input/output
IHAC	Improved Hierarchical Agglomerative Clustering
IPS	Intrusion Prevention System
ISO	International Organization for Standardization
LAN	Local Area Network
MAC	Mandatory Access Control
MGSO	Multi-objective Glowworm swarm Optimization
NP-hard	Nondeterministic Polynomial hard
NTC	Network Transfer Cost

OLTP	On-Line Transaction Process
OS	Operating System
PCs	Personal Computers
PSO	Particle Swarm Optimization
QAP	Quadratic Assignment Problem
QEP	Query Execution Plan
QoS	Quality of Service
RDMSs	Replicated Database Management Systems
REQ	Request
RM	Reference Model
ROWA	Read One-Write-All
RSS	Residual Squared Error
SPARC	Standards Planning and Requirements Committee
SQL	Structured Query Language
SSD	Solid State Disk
SSE	Sum of Squared Error
TOC	Total Operation Cost
VANETs	Vehicular ad hoc Networks
WAL	Write Ahead Logging
XML	Extensible Markup Language

LIST OF TABLES

DYNAMIC DATA REPLICATION AND DISTRIBUTION IN DATABASE SYSTEMS

Saadi Hamad THALIJ ALLUHAIBI

Department of Computer Engineering
Ph.D. Thesis

Advisor: Assoc. Prof. Dr. Veli HAKKOYMAZ

The development of data base systems technology has created its own theoretical foundations and has qualified a large number of applications. Similarly, the growth of computer networks enabled the association of multiple computers for interchange of data and resources. The role of centralized Data Management System and its accessibility to multiple users concurrently has made it impossible for the Data Benefactors to focus the data at one large mainframe site. The superior network traffic and reduced efficiency have forced splitting of data at many sites with each location having their own storage and local processing abilities. This directed to the development of Distributed Databases (DDBs) that play a noteworthy role in today's era where dependence on reliable and accurate data has become a compulsion. The innovations in hardware, software, protocols, storage and networks have transformed the position of the business necessities by making the handling of DDBs a feasible and operational decision. The supremacy of distributed databases lies in the capability to deliver interconnected data from any physically separated site to any other site. Distributed Database Management System (DDBMS) fits to the class of system software that manages distributed database and offers transparent access ability to multiple users across multiple sites by integrating parallelism and modularity. Though efficient, the designing of

DDB has many practical limitations in selecting efficient methods for fragmentation, allocation and replication of data.

This research thesis focuses on developing efficient solutions for the DDB design issues. The main aim of this thesis is to propose powerful schemes for data fragmentation, allocation and replication for enhancing the query processing in DDBs for better performance. The first approach concentrates on the limitations of utilizing the observed data about the queries to decide the fragmentation issue ineffective at the preliminary distributed database design where the efficiency is estimated only through proper design and network communication cost between sites. To resolve this issue, we give the improved model of hierarchical agglomerative clustering (IHAC) algorithm to derive semantic fragmentation of the distributed databases. The IHAC constructs the data representation matrix by considering all data objects instead of data counts while the traditional hierarchical agglomerative clustering algorithm constructs the data representation matrix based on the data count or frequency to select and compute similarity measures. This enhances the performance of clustering the data objects and hence the data fragmentation can be achieved efficiently.

The second approach focuses on the performance degradation in DDBs due to the communication cost by remote access query and retrieval of data. This can be optimized through an efficient data allocation approach that will provide flexible retrieval of a query by low cost accessible sites. For this process, Chicken Swarm Optimization (CSO) algorithm is utilized which characterizes the Data Allocation Problem (DAP) into optimal problem of choosing the appropriate and minimal communication cost provoking sites for the data fragments. Then the CSO algorithm optimally chooses the sites for each of the data fragments without creating much overhead and data route diversions. This enhances the overall distributed database design and subsequently ensures quality replication.

The third approach considers the issue of optimal replica selection and placement. Initially, the snapshot replication and merge replication process for suitable databases are illustrated. Secondly, the MGSO approach is employed for selecting the location and number of replicas for placement in the network. This approach utilizes the random patterns of read-write requests for the dynamic window mechanism for replication while also modelling the replication problem and a multi-objective optimization problem that is resolved using MGSO.

Evaluation of the proposed techniques is performed in Hadoop cluster environment using master-slave dedicated machines. The evaluation study performed over a large dataset from three major sources; namely, Twitter, Facebook and YouTube containing various types of data such as text, audio and video files with varying sizes. The evaluation and comparison show that the proposed technique in this research thesis perform better than the compared fragmentation, allocation and replication techniques. Hence it is proved that this work significantly enhances the design performances of DDBs by solving the problems of data fragmentation, allocation and replication.

Keywords: Distributed database, replication, fragmentation allocation, replication, snapshot replication, algorithms, optimization.

VERİ TABANI SİSTEMLERİNDE DİNAMİK VERİ KOPYALAMA VE DAĞITIMI

Saadi Hamad THALIJ ALLUHAIBI

Bilgisayar Mühendisliği Bölümü
Doktora Tezi
Tez Danışmanı: Doç. Dr. Veli HAKKOYMAZ

Veri tabanı sistemleri teknolojisinin gelişmesi ile birlikte yeni teorik temeller oluşmuş ve çok sayıda uygulama kullanılır hale gelmiştir. Benzer biçimde, bilgisayar ağlarının gelişimi de çok sayıda bilgisayarın birbirlerine bağlanarak aralarında veri ve kaynak değişimi yapabilmelerini sağlamıştır. Merkezileştirilmiş Veri Yönetim Sistemi ve bu sisteme çok sayıda kullanıcının aynı anda bağlanabilmesi nedeni ile Veri Yararlanıcılarının veriye tek bir büyük merkezi sistemde odaklanmaları olanaksız hale gelmiştir. Artan ağ trafiği ve azalan etkinlik nedeni ile bir çok alanda verinin bölünmesi zorunlu hale gelmiştir, ve her bir lokasyonun kendi depolama ve lokal işleme becerileri oluşmuştur. Bunun devamında Dağıtılmış Veri Tabanları (Distributed Databases) (DDB) ortaya çıkmıştır. Günümüzde güvenilir ve doğru veriye ihtiyaç zorunluluğu olduğundan bu veri tabanları çok önemli bir rol oynamaktadır. Yaşanan yenilikçi gelişmeler sonucunda donanım, yazılım, protokol, depolama ve ağlar ticari gereksinim haline dönüşmüştür. Bunun ile birlikte DDB kullanımı yapılabilir ve operasyonel bir karar haline gelmiştir. Dağıtılmış veri tabanlarının üstünlüğü fiziksel olarak bağı bulunmayan herhangi bir konumdan başka bir konuma bağlı veriyi aktarabilmesidir. Dağıtılmış Veri Tabanı Yönetim Sistemi (Distributed Database Management System) (DDBMS), dağıtılmış veri tabanını yöneten ve paralellik ile modülariteyi entegre ederek birçok konumda birçok kullanıcıya şeffaf erişim imkanı sunan uygulama yazılımları sınıfına girmektedir. DDB tasarımı etkin olsa da

çok sayıda uygulama kısıtlamalarına da sahiptir. Bu kısıtlamalar: verinin parçalanması, tahsisi ve kopyalanması konusunda etkin yöntemlerin seçilebilmesidir.

Bu araştırma tezi, DDB tasarım sorunları ile alakalı etkin çözümlerin geliştirilmesi konusuna odaklanmaktadır. Tezin esas amacı ise, DDBlerde sorgulamayı güçlendirip daha iyi performans sağlayabilmek adına verinin parçalanması, tahsisi ve kopyalanması konusunda güçlü yöntemler sunabilmektir. Birinci yaklaşımda, sorgulamalar ile alakalı gözlemlenen verilerin kısıtlamalarına odaklanılır. Buradaki amaç, geçici dağıtılmış veri tabanı tasarımında etkisiz olan parçalama konusu hakkında bir karara varmaktır. Bu aşamada etkinliği hesaplama işi sadece doğru tasarım ve alanlar arasındaki ağ iletişim masrafları üzerinden hesaplanır. Bu sorunu çözebilmek adına geliştirilmiş Hiyerarşik Aglomeratif Kümele (hierarchical agglomerative clustering) (IHAC) algoritma modeli kullanılarak dağıtılmış veri tabanlarının semantik fragmantasyonuı türetilir. IHAC, veri sayıları yerine tüm veri objelerini göz önüne alarak veri temsil matrisini oluşturur. Geleneksel hiyerarşik aglomeratif kümeleme algoritması ise veri temsil matrisini oluştururken benzerlik ölçümlerini seçmek ve hesaplamak için veri sayısı veya sıklığını göz önüne alır. Bu sayede veri objelerinin kümeleme işlemi daha güçlü olur ve bunun sonucu olarak da veri parçalama işlemi daha etkin bir biçimde yapılır.

İkinci yaklaşımda, sorgulama uzaktan erişimi ve verinin geri alınması nedeni ile oluşan iletişim masraflarından doğan DDB performans bozulmasına odaklanılır. Bu işlemi optimize etmek için etkin bir veri tahsisi yaklaşımı kullanılabilir. Bu yaklaşımda düşük masraf ile erişilebilen alanlar üzerinden sorgulamanın esnek bir biçimde alınması sağlanır. Bu işlemi yapmak için Chicken Swarm Optimization (CSO) algoritması kullanılır. Bu algoritma, Veri Tahsis Problemi (Data Allocation Problem) (DAP)'ni uygun ve minimal iletişim masrafını seçebilecek bir optimal probleme dönüştürür. Sonrasında, CSO algoritması her bir veri parçası için alanı en uygun biçimde seçer. Bunu yaparken gereksiz yük oluşturmaz ve veri güzergah sapmasına neden olmaz. Bu sayede dağıtılmış veri tabanı tasarımı genel olarak iyileşir ve sonrasında kaliteli kopyalama gerçekleşir.

Üçüncü yaklaşımda ise optimal kopya seçimi ve yerleştirme konusu ele alınır. İlk olarak, uygun veri tabanlarına ait anlık (snapshot) kopya ile birleştirme (merge)

kopyalama süreçleri gösterilir. MGSO yaklaşımı, ağ içerisine yerleştirilecek kopyaların konumu ve adedini seçmek için kullanılır. Bu yaklaşım, kopyalamanın dinamik pencere mekanizması için read-write taleplerinin rastgele desenlerini kullanırken aynı zamanda MGSO kullanarak kopyalama problemini ve çok-hedefli optimizasyon problemini de modeller.

Önerilen tekniklerin değerlendirmesi Hadoop küme ortamında gerçekleştirilmiştir ve bunu yaparken "master-slave" adanmış makineler kullanılmıştır. Değerlendirme işlemleri üç ana kaynaktan büyük bir veri seti üzerinden gerçekleştirilmiştir. Bu kaynaklar Twitter, Facebook ve YouTube olup içlerinde farklı boyutlarda metin, ses ve video türünde veriler bulunmaktadır. Değerlendirme ve karşılaştırma sonuçları göstermektedir ki, bu araştırma tezinde tavsiye edilen teknikler karşılaştırma yapılan bölme, tahsis ve kopyalama tekniklerinden daha iyi sonuç vermektedir. Bu nedenle, bu çalışmanın veri bölme, veri tahsisi ve veri kopyalama sorunlarını çözerek DDB tasarımını çok güçlendirdiği söylenebilir.

Anahtar Kelimeler:Veri kopyalama , dağıtımı , bölünmüşlük ,tahsis.

1 Introduction

Globalization and massive Internet usage has led to various developments which sometimes require some demands to facilitate them. The development of multi-national organizations has created a unique style in data sharing. These organizations have multiple units in different countries or regions around the world. Each unit demands individual local data and also communication is necessitated between units to share their data and resources [1;2]. This objective requires synchronized usage of replicated databases. This process is vital in many aspects of organizations as well as other domains which have users or clients in distributed locations. Hence the distributed systems concept with the distributed and replicated databases is important in research communities [3]. This chapter illustrates the outline of distributed database (DDB), its benefits and weaknesses, mechanisms, outline of DDBMS and its policy.

1.1 LITERATURE REVIEW

The principles of distributed database systems (DBMS) are provided by Özsu and Valduriez [4]. A centralized database management system (CDBMS) is a complex software program that allows an enterprise to control its data on a single machine. In most organizations using CDBMS, users need to access their own workgroup database. However, users also need to access data in some other workgroup's database, or even in the larger, enterprise-level database. The need to share data that is dispersed across an enterprise cannot be satisfied by centralized DBMS software. To address this requirement, a new breed of software to manage distributed data called a distributed database management system (DDBMS) is required. A DDBMS maintains and manages data across multiple computers. A DDBMS can be thought of as a collection of multiple, separate DBMSs, each running on a separate computer, and utilizing some communication facility to coordinate their activities in providing shared access to the enterprise data. The fact presented by Tenopir and Douglass [2] is that data is dispersed across different computers and controlled by different DBMS products are completely hidden from the users. Users of such a system will access and use data as if the data were locally available and controlled by a single DBMS.

C. J., Date [1] describes about the data access in DBMS. Data access over the WAN and optimization of replicas is rarely addressed in database research. In a DBMS there is normally only one internal method of accessing data. For instance, a data server serves pages to a client. In detail, a client sends a request for data to the DBMS, and the DBMS uses the implemented method for serving the requested data. For read and write access the same method is used, independent of the amount of data accessed. For accessing large amounts of data of a remote file, the usage of a file transfer mechanism to transfer the entire file and access the file locally can be more efficient in terms of response time than remotely accessing and thus streaming many single objects of a file.

Özsu and Valduriez [4] describes about the challenges and procedure of the data storage in DBMS. In a distributed database system, the database is stored/spread physically crosswise over machines or sites in distinctive areas that are associated together by some type of information correspondence network. When dealing with intricate data or complex relationships, object databases are more commonly used. Object databases, in contrast to relational databases, store objects rather than data such as integers, strings or real numbers. Each object consists of attributes which define the characteristics of an object. Objects also contain methods that define the behavior of an object, also known as procedures and functions. When storing data in an object database there are two main types of methods, one technique labels each object with a unique ID. Every unique ID is defined in a subclass of its own base class where inheritance is used to determine attributes. A second method is utilizing virtual memory mapping for object storage and management. Advantages of object databases with regard to relational databases allow more concurrency control, a decrease in paging and easy navigation. However, there are some disadvantages of object databases compared to relational databases such as: less effective with simple data and relationships, slow access speed and the fact that relational databases provide suitable standards oppose to those for object database systems.

1.2 DISTRIBUTED DATABASE

Distributed database comprises of many files placed in many sites on the same network or dissimilar networks [4]. Shares of the database are kept in numerous physical sites and handling is distributed amid several database hubs. A distributed database configuration consists of sparsely-coupled warehouses of data. The traditional database systems keep all the storage devices connected to the main physical server in a single location as they are needed for multiple purposes. With the advancement of Internet, the distributed database functions as a single system although the database physical devices are placed across multiple locations to improve the hassle-free performance. The pattern of distributing the data across multiple servers enhances the performance at end-user worksites with user-friendly transactions at multiple destinations without crashes [5].

Replication [6] and duplication are the two vital processes required to maintain the distributed databases with exact relevance with the recent updated versions in all locations. Replication deploys specialized software for detecting the updates in the database in other locations and if changes detected, the replicated databases will also update the data through authorized replica update process. This complex and time-consuming process of replication depends on size and number of distributed databases requiring more resources. Meanwhile, duplication is the process with less complexity but less efficient due to lack of continuous monitoring. It fixes one database as main DDB and replicates it to different locations at set time. This process is repeated and the previous duplicated files are replaced after some time. Similarly, apart from distributed database replication, duplication and fragmentation, many tools are also commonly utilized for business and private data models with cost efficiency, security, consistency and integrity. Figure 1.1 shows a simple DDB system [4].

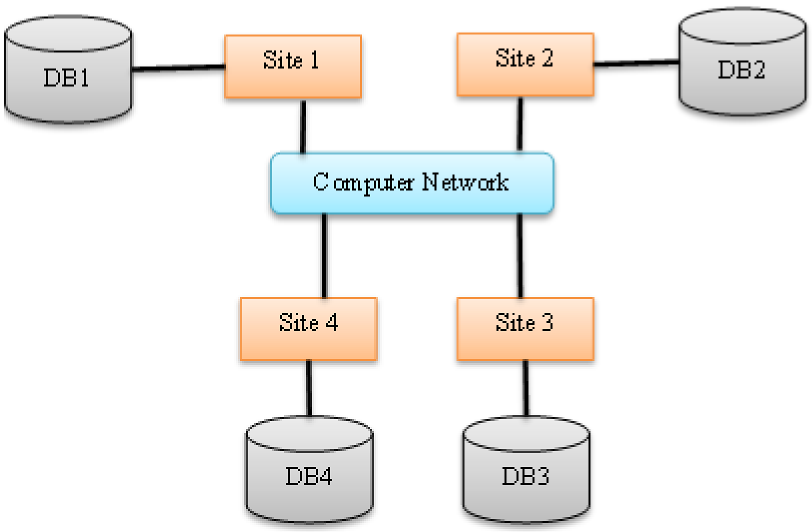


Figure 1.1 Distributed Database system

DDB is of two important types namely homogeneous and heterogeneous [7]. Homogeneous DDB has same kind of software and hardware explored for each of the database instances while the heterogeneous DDB has varying hardware, functioning schemes, database administration systems, and data models for diverse databases. Both these types of DDB, however, appear as solitary database through a single unified interface.

1.2.1 Homogeneous DDB

In homogeneous distributed database, similar software is placed in all sites to identify every site and provide cooperation. They submit their own self-sufficiency of altering the software. Homogeneous DBMS acts as a solitary system [8] and increases the user-friendly handling and design. Certain conditions are needed to be satisfied to utilize homogeneous DDB which are: 1) Similar and compatible data structures must be utilized at each location and 2) Similar and compatible DB application has to be utilized at each location. Homogeneous DDB is of two types:

- **Autonomous:** These DDB independently functions and are incorporated with controlling application and enable data sharing through messages.

- **Non-Autonomous:** These DDB are distributed at different nodes and require a master control to control and manage the data.

1.2.2 Heterogeneous DDB

Heterogeneous DDB employs different software in different locations [9] which is difficult for query and transaction processing. This creates a situation where the sites are not aware of the scheme used in other sites and increases difficulty in cooperation of transactions and data sharing. Heterogeneous DDB systems provide the facility to utilize different hardware and software at different nodes though they are incompatible but due to the ability of central master, the data sharing is flawlessly performed. So, different operating systems running on different computers can be used at each of the locations. For instance, traditional DDB in an old version of DDBMS system can connect with latest DDBMS in another location and likewise a system running on Windows can communicate with another system in another location running on UNIX.

Heterogeneous systems are generally employed where the sites in different location utilize their own unique hardware and software. Heterogeneous DDB utilize the user queries in modified versions to support the local sites and enable communication between different DBMS. The SQL database language is typically employed. When the hardware is dissimilar, the computer codes and word-length are modified and if the software is different, the language is modified. However, the heterogeneous system is not feasible both technically and economically as it is difficult to update the user at one location by a user at another location [10]. Heterogeneous DDB is of two types:

- **Federated**: These DDB are independent and incorporated to operate as a distinct database system.
- **Un-federated**: These DDB utilize a central organizing component to access the distributed data at different location.

1.3 ADVANTAGE AND DISADVANTAGE OF DDB

Distributed databases are commonly traced on more than one server, but interconnect to fulfil a mutual objective. The isolation of different system components and application servers provides advantages of cost, management, and performance yet, there are also disadvantages in them [11],[4].

1.3.1 Advantages

Reconfigurable: The original configuration of the machine is based on the load for stock web servers; for instance, essential to support a high volume of smaller exchanges, though a database server with an information distribution center needs to support a generally low volume of huge exchanges (i.e., complex inquiries). Isolating the web server from the database server in this precedent enables the framework directors to improve these machines without trade-off. A machine designed as a web server will contrast from a machine arranged as an information stockroom database server. In the event that exhibition issues emerge in a circulated design, it is a lot simpler not exclusively to distinguish issues yet in addition to illuminate them without the danger of trading off different segments [12].

Modular growth: The handling of expansion of storage n a distributed environment is easier. The adaptability of adding new sites allows a relationship to grow generally effectively. Processing and storing to the system can holder the expansion in database estimate [13].

Platform Autonomy: The DDBMS provide platform autonomy. The applications and databases do not depend on same machine. SQL*Net and Net8 provide a protocol-independent system edge permitting connectivity. This open benefit helps Database administrators (DBAs) and developers to select the platforms deprived of restrictions. Hence the platform whether major or minor does not become a constraint [14].

Site Autonomy: DDB enable different areas to share their information without surrendering authoritative control [15]. In the event that a database example at central station contains especially delicate data or has high accessibility prerequisites, it can share information and also maintains security and accessibility. Moreover, some random site in a DDB environment can pursue its own managerial systems and upgrade paths. The admin from different sites are in correspondence with each other and that they arrange their activities without any commitments.

Fault Tolerance: The fault tolerance of the system is that when a solitary component fails in a distributed network, it is easier to tackle the situation than in a centralized system [16]. Admins can design the system with solution to this fault issues. For instance, when a DDB is corrupted in a single node, the other nodes compensate and avoid complete network failure.

Scalability: Taking advantage of the parallel query option, the processors enable scalable processing [17]. This is because the applications have their own host and assign hardware properties applicably.

Location Transparency: It enables the users and applications not to face difficulty in accessing the data location [18] with minimal influence on users and applications.

Enhanced Security: Distributed architecture of DDB enables independent processing and does not require sharing of accounts and passwords and hence the security is quite enhanced. The data can be kept private with higher protection [19],[20].

Improved Performance: As the data is available to the user at their will, the speed of DDBs and the parallel processing increases the performance in terms of CPU and I/O services [21].

Economical: Cost of the DDBs is very less as the single computer design of DDBMS based on network of distributed machines consuming less power to access remote distributed data [22], reducing the cost of transmission and positioning in local areas. The partition of the data in distributed manner also reduces the cost of storage devices being summed at one site location.

1.3.2 Disadvantages

Complexity: A distributed DBMS that conceals its property from the client and gives a satisfactory dimension of performance, dependability, accessibility is characteristically unpredictable than a central DBMS. The information can be duplicated also includes an additional dimension of unpredictability to the DDBMS. When data replication is not performed sufficiently, there will be inefficient accessibility, dependability and performance compared with the central system [23].

Integrity control: The database integrity based on the legitimacy and steadiness of stored data is not acceptable to be interrupted. Upholding reliability constrictions expects access to a lot of information that characterizes the constrictions [8]. In a DDBMS, the correspondence and preparing prices essential to implement reliability constrictions are great when contrasted with centralized system.

Query Processing: Query processing in DDBs analyses queries to convert them into data manipulation tasks [24]. However, this process is not completely effective in handling large number of queries.

Lack of Standards: The DDBMs provides effective communication, but the normal communication and data admission practices are not available due to the significantly limited the potential [4].

Update: Updating the software and data in DDBMS is not easy as it needs to be done in all locations to ensure the availability of current data to all the users [25].

Database design: The design issues of fragmentation of data, provision of fragmentation to particular sites, and data duplication [11] are significantly difficult than the centralized DDBs.

Concurrency control: It is the action of planning simultaneous accesses to a database in a multiuser DBMS [26] and licenses clients to get to a database in a multi-customized style framework. The fundamental issue in achieving this objective is to anticipate database appraises achieved by one client with database recoveries and appraises achieved by another. The concurrency control issue is aggravated in a DDBMS based on (1) clients may contact data in a wide range of processors in a distributed framework, and (2) a concurrency control at one processor cannot quickly cooperate at different computers.

1.4 PROBLEM DESCRIPTION

The employment of DDB is very challenging due to the contradictory necessities of preserving data steadiness and meeting transaction's targets. Each distributed operation retrieving a data item and takes the flexible quantity of time due to concurrency control, I/O and connection delay while preserving the evenness of underlying database. To meet the transaction's deadline, an effort should be made to manage timing constraints, access to CPU, main memory, I/O devices and shared data. Designing distributed databases mostly faces problems in the form of fragmentation, replication and allocation of the data.

Existing approaches for database design faces limitations in the form of cost, performance, reliability, flexibility and security. The real-time DDBs must preserve the steadiness constraints and also assure the time constraints mandatory for client requests. Though, when the distributed system size increases, the user access time also increases and in turn raises the energy of the replica placement. Hence replicating these databases is entirely different from the replication of other documents. Various research works have been provided with different methodologies to replicate distributed databases, however the major issue in those methods is the optimal selection of amount of replica and top position for replica assignment.

In this research, the limitations in fragmentation, replication and allocation process for distributed databases are primarily focused to consider reliability, cost, security and performance of DDBMS.

1.5 OBJECTIVE OF THESIS

The objectives of the thesis are as follows:

- To consider the limitations of models available for data fragmentation and develop an efficient data fragmentation technique for distributed databases.
- To minimize communication cost and develop an efficient data allocation technique for distributed databases.

- To analyze the limitations of available procedures for data replication and develop a new approach of data replication to improve the data reliability.

1.6 HYPOTHESIS

The contributions of this research thesis are:

- Improved Hierarchical Agglomerative Clustering Based Fragmentation Approach for DDBs.
- Chicken Swarm Optimization Based Data Allocation Technique for DDBs.
- Efficient Data Replication Strategy using Multi-objective Glowworm Swarm Optimization for Distributed Database Systems

1.7 THESIS ORGANIZATION

Chapter 1 has presented introduction and the background of the research. It provides an overview of databases, distributed databases, its advantages and disadvantages. It also provides the problem description, objectives of the research and thesis organization.

Chapter 2 discusses the literature review for data fragmentation, replication and allocation techniques. Various techniques are discussed and the limitations of them are highlighted which form the motivation of this thesis. The literature review is divided into three sections, each corresponding to Data fragmentation techniques, Data allocation techniques and Data replication techniques.

Chapter 3 describes the various concepts in DDBs such as components of DDB, overview of DDBMS, data distribution space, DDB design and security concepts.

Chapter 4 highlights the need for fragmentation and the issues faced in existing methodologies. Then the proposed data fragmentation technique for distributed databases is explained and evaluated.

Chapter 5 highlights the challenges in allocation process and the issues faced in existing methodologies. Then the proposed data allocation technique for distributed databases is explained and evaluated.

Chapter 6 highlights the need for replication, its types and the issues faced in existing methodologies. Then the proposed data replication technique for distributed databases is explained and evaluated.

Chapter 7 describes the performance evaluation results of the three design techniques by comparing them with existing techniques.

Chapter 8 concludes the thesis and also suggests some of the possible directions for future researches.

2
Literature Review

This chapter describes the recent research works about the data fragmentation, allocation and replication strategies. The study of these literature works provides an insight towards which of the limitations hinder the performance of distributed databases and it can be utilized to developed newer and better techniques. In short, this chapter provides the motivation for this research work.

2.1 DATA FRAGMENTATION TECHNIQUES

Cuzzocrea [27] developed a fragmentation technique using the k-means clustering algorithm for the large XML data warehouses. This technique is different from the classical fragmentation techniques as it utilizes clustering-based fragmentation through control setting process which is much suitable for the distributed nature of the XML warehouses. However, this model still requires much improvement in handling the global query processing.

Bhuyar and Deshmukh [100] developed a new method for database fragmentation in DDBs using modified CRUD (Create, Read, Update, and Delete) matrix and cost utilities. This model significantly improved the performance of the DDBMS by evading recurrent remote admission and high data transmission amid the sites.

Pazos and Pérez [28] suggested the use of non-linear round-trip response time for the improvement of the fragmentation model. This approach modelled the vertical fragmentation into a non-linear problem and was resolved using two metaheuristics: the threshold tolerant procedure (a variant of simulated annealing) and tabu search with significant efficiency.

Hauglid and Nørvåg [29] developed DYFRAM, the dynamic fragmentation and replica administration model for DDBs. This decentralized approach utilized the access patterns of sites and fragmented the database using current admission history, targeting at exploiting the number of local admissions. The advantage of this model is that it minimizes the communication costs. However, this model does not efficiently detect the periodically recurring access patterns.

Fauzi, Noraziah and Zin [30] developed a novel binary vote assignment grid quorum algorithm for improving the distributed database fragmentation. This model of fragmentation utilized the timeliness to achieve higher efficiency. As the transaction execution in replication problem is resolved, the consistency is achieved. It also provides alleviates the lock with small query size and updates the important data effectively.

Boukraâ, Boussaïd and Bentayeb [31] developed a vertical fragmentation method for XML data warehouses using frequent path sets. This approach uses association rules to divide and cluster recurrent path sets into fragments and also resolve the issue of restructuring the unique non-fragmented plan to guarantee the fragmentation reversibility.

Goli and Rankoohi [32] designed a novel vertical fragmentation process centered on ant cooperative activities in DDBs. This ant-based fragmentation model employs the swarm intelligence in resolving the fragmentation optimal problem. This process makes the transaction handling at each site as mush confined and also reduces the overall cost of operations. It is highly accurate in fragmentation while also less expensive.

Pazos and Martínez-Luna [33] developed a vertical fragmentation model to reduce the round-trip response time in distributed database systems. This model utilized a mathematical programming model with round trip response time as the parameter to be optimized. In round trip response time, the delay parameters are considered for estimation.

Wiese [34] presented the clustering-based fragmentation and data replication scheme for elastic query responding in DDB. This method offers fresh self-management and self-configuration methods for an accessible query management. This clustering-based fragmentation allows fault-tolerance and parallelization effectively. However, the fragmentation is completely dependent on single relaxation attribute which will lead to massive replication that requires control mechanisms. This is the major limitation of this clustering-based approach.

Dharavath and Kumar [35] developed an a priori-based vertical fragmentation technique that uses attribute usage matrix for dividing the relations. This approach provided proper fragmentation at the initial stage of a distributed database to overcome the constraint of higher difficulty as well as the difficulty of more calculation. Additionally, this method also has the advantages of more availability of fragment selection. However, this approach provided below par results when the minimum support value is determined optimally.

Luong, Nguyen and Le [36] presented an enhancement on fragmentation in DDB using the knowledge-oriented clustering methods. This approach was developed to improve the vertical and horizontal fragmentation using the classical clustering algorithms of k-means and hierarchical clustering. This approach proved that the integration of these clustering algorithms increased fragmentation effectiveness while also balancing the time complexity.

Thakur and Ram [37] utilized the FP-MAX data mining algorithm to mine frequent item set for vertical fragmentation. This approach proved its efficiency for small as well as larger databases with improved fragmentation but the performance is very low with the skewness of the data.

Hadden and Mahdy [38] developed a software paradigm named the royal split for enhancing the data fragmentation process with the prevention of data loss. This paradigm controls all the aspects of the network namely node population, topology, file size, and number of fragments. Based on these aspects, the data files are fragmented, distributed to nodes on the network, recovered, and defragmented with improved accuracy.

Mai [39] developed a heuristic algorithm for fragmentation and allocation in distributed object-oriented database. This simultaneous class fragmentation approach utilizes the costs of data communication for reduced processing cost. The only foreseeable limitation in this approach is the non-utilization of queries in processing these fragments valiantly.

Amer and Sewisyand [40] developed an optimized method for horizontal fragmentation which efficiently fragments the database using arithmetic models to solve database design issues. This model is highly cost effective and also ensures optimal fragmentation is achieved.

Drissi and Benslimane [41] developed a horizontal fragmentation approach for fuzzy querying databases using query execution strategies. This process of fragmentation minimizes the number of fragments accesses and hence reduces the costs. However, this model is not must applicable to transactional databases as it is not peculiar and complex in computing them.

Eladl [42] developed a modified vertical fragmentation strategy for distributed database by employing the advantages of the fuzzy logic concepts. This strategy of fuzzy logic is better than the crisp logic and it is reflected in the performance improvement with the database developer provided complete independence for the determination of importance of query attributes.

Sewisy and Amerand [43] proposed the development of a novel query-driven clustering-based technique for vertical fragmentation and allocation in DDBs. This technique is employed at the initial stages of DDBS designing with including the data statistics or empirical results in both the dynamic and static environment. The major limitation of this model is the general limitation of hierarchical clustering, i.e. complex processing structure.

Luong [44] developed a fragmentation method centered on K-means rough clustering for the distributed databases. However, the complexity of this algorithm is more optimal than general k-means clustering which becomes the high point of this fragmentation technique.

Mehta and Barlawala [45] proposed a differential bond energy procedure for optimal vertical fragmentation of DDBs to provide all the possible combination of solutions and hence has more likelihood to give better values. This algorithm optimizes data fragmentation in high dimensional datasets and provided better performance in terms of mean difference and good quality fragments.

2.2 DATA ALLOCATION TECHNIQUES

Hababeh and Bowring [46] developed a high-performance computing technique for data allocation in DDBs. This computing method uses high speed clustering and allocation process to regulate the fragment allocation in each cluster and site. This process ensures the data availability and reliability. The communication cost and execution time are minimized in this model which can also be implemented in larger systems where the input is large.

Adl and Rankoohi [47] designed and developed a new ant colony optimization based heuristic system for data allocation issue in DDB. The performance of the data allocation process is significantly improved with reduced time and under-memory capacity constraints. This algorithm has better convergence for the distributed systems but for large variety of values, the system is still under development.

Mamaghani and Moghaddam [48] proposed and developed a novel evolutionary algorithm for solving static data allocation problem in DDBs using two methods of genetic algorithm and learning automata. Thus, the data allocation problem is better resolved using this evolutionary algorithm with less overhead.

Jagannatha and Kanth [103] presented a Non-replicated dynamic fragment allocation model using the access threshold, time constraints of database accesses and the volume of data in distributed database systems for reallocation. This model will decline the migration of fragments and data transfer cost and maximize the overall throughput, performance and efficiency of the distributed database applications.

Jagannatha and Kanth [103] also presented synthesis of non-replicated dynamic fragment allocation. The data allocation is performed only when the site has higher data transfer than other sites and the accomplishment of the distributed database queries are minimized.

Mashwani and Salhi [49] developed a decomposition-based hybrid multi-objective evolutionary algorithm for dynamic resource allocation. This algorithm utilized the combination of two crossover operators to allocate resource. However, this algorithm is still in the early stages and hence requires subsequent research to verify its reliability.

Gope [105] developed a set of heuristic algorithms for fragment allocation in a DDB. These algorithms are developed to resolve the DAP problem which is modelled based on Quadratic Assignment Problem (QAP). This model has less cost and higher efficiency. However, this model does not consider replication of fragments to multiple sites.

Li and Wong [50] presented a data allocation in scalable distributed database systems based on time series forecasting. This model enhanced the short-term load forecasting processes to remove overloading and degradation. Thus, the DAP problem is resolved with efficient results.

Tosun and Cosar [104] proposed an allocation approach for load balancing in DDB. This approach utilized the concepts of sharing of resources, allocation of fragment replicas and transaction in distributed database system to balance the overloads. This process improved the uniform workload distribution and also reduced the cost computation.

Singh and Virk [51] proposed a non-replicated static data allocation scheme for the DDBs using biogeography-based optimization (BBO). This optimization-based strategy allocates the fragments during the design of distributed database system with minimal total data transmission cost. The proposed BBO algorithm significantly minimize the data transfer cost during the execution of a set of queries. However, sometimes the average cost of allocation for BBO is more than Genetic algorithm.

More and Nasik [52] designed a data allocation strategy assigning processors to real-time application tasks, based on real time scheduling strategy. This strategy also utilizes the combination of in-Memory Database towards every system as well as Shared Memory Database in Server side of Distributed System due to which the high availability property of database can be achieved.

Singh [53] designed and developed a combination of threshold and time constraints enables better allocation than considering either one constraint as the frequency of access patterns changes much faster. This algorithm also decreases the total cost of reallocation. The network traffic and data transfer cost are also reduced and thus the overall performance is significantly improved.

Sen and Narayanan [54] analyzed a simulated annealing method for data allocation utilizes a local search process for DDB systems. But the issue arises when the distance measure assumption is not exploited for the data allocation process.

Malyshkin and Schukin [55] proposed the development of scalable distributed data allocation for the LuNA fragmented programming system. This algorithm has been developed for large multi-computers. This approach models the data allocation problem into a workload imbalance problem and resolves it using dynamic load balancing procedure. The limitation of this algorithm lies in the static nature of supplementary information and single dimension of neighborhood host description.

Kumar and Gupta [56] developed a clustered approach to dynamic data allocation problem in th DDBs which overcomes the problem of network site clustering for enhancing the efficiency of data allocation. This approach initially reallocates the data fragments to the clusters in redundant and non-redundant distributed DBS in order to minimize the remote data access and network overhead based on changing data access patterns. The major limitation is that this approach is currently functional only in small scale networks with a smaller number of database sites.

Darabant and Varga [57] developed a model for access patterns optimization in DDBs using data reallocation. This reallocation model utilizes the data access patterns and database statistics resolves the reallocation of fragments problem for optimizing the query response time by the users. The advantage of this model is that it has higher efficiency and the response time is also less. However, the cost for reallocation is slightly higher in this model which should be minimized.

Fu and Sun [58] presented a dynamic non-redundant data allocation method that resolves the DAP problem with less cost of migration.

Kulba and Somov [59] Developed an allocation method for large-scale distributed systems. This model enriched the performance of the distributed systems but the major limitation lies in sub-optimal allocation due to higher iterations.

Mahi and Kodaz [60] developed a new approach to solve data allocation problem based on particle swarm optimization algorithm. This algorithm determines the optimal data allocation and thus minimizes the total transmission cost. This algorithm provides comparable results even when the dimensionality of the problem increases where the solution space grows exponentially. However, this problem is still a concern for huge size of dimensionality.

Lwin and Naing [61] presented a non-redundant dynamic fragment allocation using the horizontal segmentation in DDBs in order to reallocate the fragments based on access patterns. This approach is highly efficient and reduces the time consumption but the major degradation factor is the cost of reallocation and update process.

2.3 DATA REPLICATION TECHNIQUES

Curino and Madden [62] developed a model named Schism, a workload-driven approach to database replication and partitioning. This approach of replication graph partitioning model to replicate the databases and reduced the cost by 30% with ease of integration. However, this model is strictly for OLTP databases and less supportive of other databases.

Abad and Campbell [63] developed a replication model named DARE which is used as adaptive data replication. It solves the problems of how many replicas to allocate and also improves the turnaround time and locality.

Naing and Win [64] presented a distributed database partial replication approach that reduces the substantial amount of distributed transactions efficiently. The graph representations are employed for the query trace monitoring which are then partitioned to form replication clusters. This approach improves the scalability and throughput but the only drawback of proposed approach is that cross-partition distributed queries can produce distributed overhead.

Xhafa and Barolli [65] developed an intra-group optimistic data replication system for the P2P groupware systems. However, this system is currently suitable only for unicast connections and hence the reliability of data transmission in multicast systems is questionable.

Crain and Shapiro [66] designed and developed a geo-distributed partial replication utilizing the concept of CAP theorem. However, this replication strategy does not reduce the overall cost of each data centers considerably.

Varma and Khatri [67] developed QoS-aware data replication in Hadoop distributed file system which utilizes QoS parameters to replicate the data blocks. The main QoS parameters considered for this purpose is the expected replication time. This algorithm has better performance with reduced replication cost but the non-consideration of power dissipation and temperature of Data Nodes in Hadoop system requires much attention.

Liu and Chandler [68] developed a selective data replication approach for the online social networks with distributed datacenters to reduce inter-data center communications while accomplished low service latency. This approach avoids the workload congestions of datacenters through effective replication with the concept of load balancing. The limitation of this approach is that it is still not accomplished the concept of satisfying both the service latency and network load at the same time with less overhead.

Warhade and Raghuwanshi [69] introduced a dynamic data replication based on the modified bandwidth hierarchy-based replication (BHR) in grid systems. This dynamic replication provides minimized access cost and access time.

Yang and Mi [70] developed an automated and scalable data replication system utilizing the load balancing concept for the distributed computation and storage structure. This Auto Replica system utilizes three approaches to replicate data for supporting multiple SLAs. It utilizes the solid-state disks (SSD) of remote Cyber-Physical Systems server nodes and also supports the parallel processing of the data replication fetching process.

Sun, Simon, and Sens [71] developed a stochastic model of replication in large distributed storage systems to replicate the data in terms of three mechanisms namely random, least loaded and power of choice. Using these mechanisms, the replication has achieved higher accuracy and reliability. However, a certain random unbalanced placement strategy causes unexpected results.

Zhu, Guo and Fu [72] developed an efficient distributed data replication (EDDA) algorithm for VANETs utilizing distribution concept with a bounded number of message copies.

Cardueline and Russo [73] developed elastic distributed data stream processing replication scheme for the QoS-aware data stream applications. This model, however, has limitations due to the greedy reconfiguration process utilized.

Xiong and Zhao [74] suggested a data replication strategy based on distributed caching system to reduce the user access delay and increasing the user performance. However, its utilization in larger system is under research and requires more investigation.

3 Concepts in Dıstrıbuted Databases

The contents of a distributed database are spread across multiple locations. That means the contents may be stored in different systems that are located in the same place or geographically far away. However, the database still appears uniform to the users i.e. the fact that the database is stored at multiple locations is transparent to the users. In this chapter, the components of the DDB are discussed and the distribution space and design strategies are also presented.

3.1 Components of DDB

The central components of DDB designing are Query Processing, Concurrency control/Transaction management, security management, Replication management, Directory management and Database Recovery management.

3.1.1 Distributed Query Processing

In a DDBS, query processing must be optimized at both the global and the local level [75]. The query is taken as input and checked, translated, and optimized after the user is validated. The processing of query in any system depends on the DDBS designing and the conversion of them into data manipulation operations [76]. The main constraint in query processing is selecting the approach for performing each query with less cost. The sharing of data, communication costs, and local data are considered as factors to select optimal strategy to provide parallel processing and execution of transactions. The problem is NP-hard in nature, and the approaches are usually heuristic.

Choosing the optimal execution scheme for a query is NP-hard in the quantity of relations. For complex questions with numerous relations, this can acquire a restrictive enhancement cost. Thusly, the real goal of the analyzer is to discover a strategy near optimal and, maybe increasingly significant, to dodge terrible techniques. The yield of the streamlining agent is an advanced query execution plan comprising of the mathematical query determined on fragments and the correspondence activities. The selection of the optimal strategy for the most part requires the expectation of execution costs of the elective applicant orderings before really executing the query. The execution cost is communicated as a weighted mix of I/O, CPU, and correspondence costs. An ordinary improvement of the prior distributed query analyzers was to overlook nearby handling cost by expecting that the correspondence cost is overwhelming. Significant contributions to the streamlining agent for assessing execution costs are fragment insights and equations for evaluating the cardinalities of results of relational operations. Query optimization refers to the process of producing a query execution plan (QEP), an execution plan for the query [77] and reduces an objective cost function. Figure 3.1 shows the basic architecture of query processing and optimization in distributed databases.

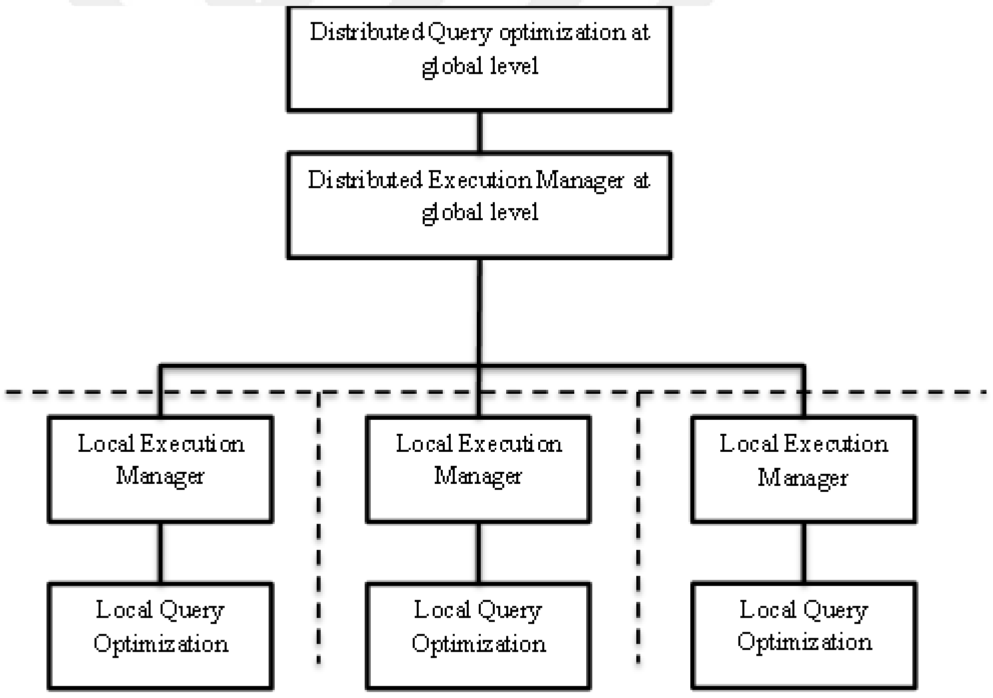


Figure 3.1 Distributed Query Processing

The mapping of the global queries into the local sites is achieved by pre-defined order with global and local data [78]. The local query results are combined together using union operation in case of horizontal fragments and joint operation for vertical fragments.

3.1.2 Distributed Transactions Management

Distributed transaction management (concurrency control) performs the process of achieving dependable DDB even when the transactions increase in number or communication link fails. This is realized through (i) distributed commit protocols that promise update; (ii) distributed concurrency control methods to confirm consistency and exclusion properties; and (iii) distributed recovery procedures to secure consistency and durability during the failures [79].

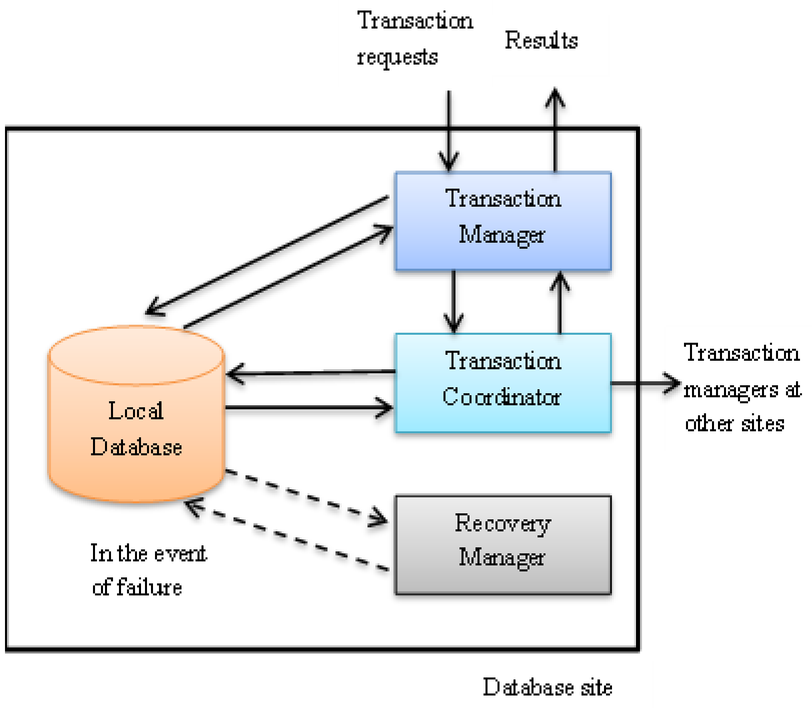


Figure 3.2 Distributed Transaction management

Figure 3.2 shows the different modules in the transaction management system at each database site [80]. Each site of a DDB framework comprises of: transaction manager, transaction coordinator, and recovery manager. To guarantee that the ACID (Atomicity, Consistency, Isolation, Durability) properties are ensured when different transactions are simultaneously executed, the transaction manager depends on concurrency control strategies. The transaction coordinator is utilized to plan and schedule sub transactions that are executed on numerous destinations.

The recovery manager is utilized to recover the database to a predictable state if failure happens [81].

Serializability theory is utilized to handle more problems for replicated databases as the local schedules are serializable and the consistency is considerable [82]. Similarly, for replicated, distributed databases, a replica control protocol (e.g., ROWA) is employed to guarantee the one-copy serializability of mutual consistency.

3.1.3 Distributed Security Management

DDB security management is the process of protecting the stored data from intruders or unauthorized access to avoid theft or modification of data. This can be achieved using security mechanisms that satisfy the security necessities of system and data sources [83]. Figure 3.3 shows the distributed security mechanism [84].

DDBMS utilizes the services of functioning system to manage the data and also to secure them. But data protection in DBMS is unlike from the data in operating system and hence the advanced techniques are utilized. The main reasons for using advanced techniques are: i) DBMSs and operating systems accept diverse data models. ii) DBMSs consist of many granularity levels for data security. iii) the objects in a database are semantically related and protecting these relations is imminent. As there are many performance issues related with security, the DBMS utilizes a set of its own security mechanisms to be applied at hardware and network level.

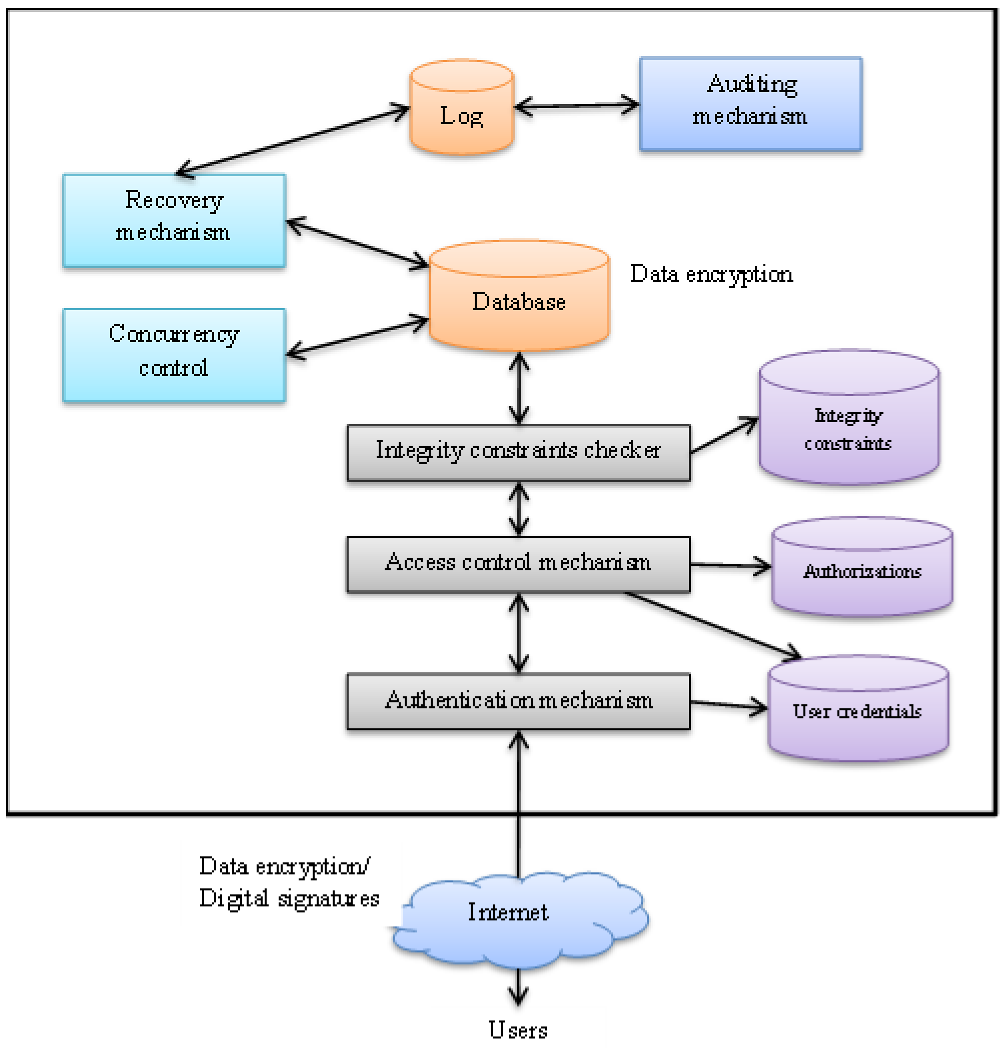


Figure 3.3 Distributed Security management

Normally, the security features are achieved with many mechanisms and especially the admission control tools are commonly used to deliver data secrecy. The authenticity of the Security Administrators or users is verified by the access control mechanism [85]. Only when there is no conflict of authorizations, the users are provided access and it is only based on the security policies of the providing organizations. Data confidentiality is achieved over the utilization of encryption methods applied on the network to prevent intruders to access data.

Data integrity is mutually guaranteed by the admission control tool. The confidentiality enforcement of the model is achieved through the modification of the data and the control mechanism validates the access to modification using specialized authentications. DBMS also allow access to set the integrity constraints using SQL only when the data has to be checked for correctness before updating the exact perfect data without modifications by intruders. Finally, the recovery process is also performed by the recovery subsystem and to verify and correct the hardware and programming failures using the log files of auditing properties.

3.1.4 Distributed Replication Management

The foremost assignment of replication is to maintain the replicas with consistent updates which are achieved by replica control. It translates the 'read and write' operations into physical operations and performs them on individual replica for each read/write queries. The read operation is done on single replica while write can be achieved on all replicas. The consistency of the models is ensured by reducing the difference between the ideal and actual replica modifications to support various applications [86]. The replica allocation systems are also employed additionally to select when and where to place the replica or remove them to new locations without disturbing the trade-off between the performance benefits and overhead. Replica control is the main and influential component in the replication area of the DBMS.

The replica control integrates the concurrency control to work correctly with DBMS of transactional semantics. It is suitable for the non-replicated and non-distributed system to allow concurrent execution of transactions and may interleave to provide constraint-based transaction operations. When the data are replicated, each transaction starts execution on different replicas and increases the challenges. In non-replicated databases, the isolation level is serializability indicating the equality between the concurrent execution and serial execution of transactions. Figure 3.4 shows the distributed replication management model [4].

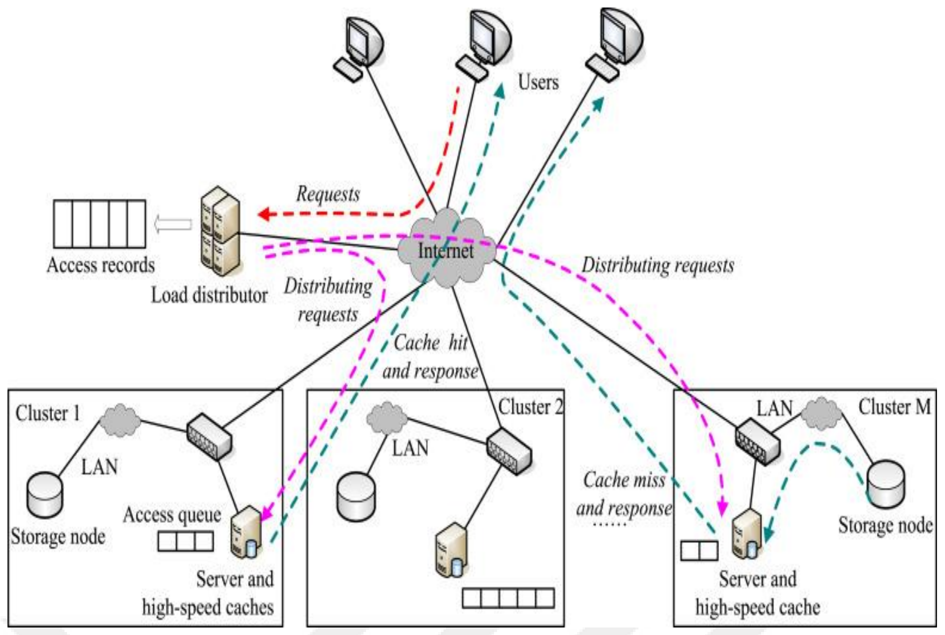


Figure 3.4 Distributed Replication management

Utilizing full replication, each site has duplicates of every single existing data object. This disentangles the execution of read operations however have high update costs. In partial replication, only few copies of some data are placed to obtain the advantage of the cost-effective way to reduce operational cost at all sites. Eager replication fuses coordination among replicas before exchange submit. On the other hand, lazy replication enables an exchange to submit data changes at one replica without coordination with different replicas. For example, all update exchanges could be executed and submitted at a particular essential replica which at that point changes to different replicas at some point. At that point, auxiliary replicas fall behind the present state at the essential. On the other hand, all replicas may acknowledge and execute updates, and the progressions to the remainder of the replicas. For this situation, replicas become inconsistent. Specifically, much formalism exists that enable the characterizing of points of confinement to the permitted uniqueness between the duplicates of a data item.

3.1.5 Distributed Directory Management

A distributed directory management system incorporates a majority of service spaces on a system each having a database archive. A service database monitoring system for each service space deals with the database storage and executes recovery of a service database. A service space association component for each service space executes message correspondence with the service database overseeing instruments to collect common association data of the service spaces and deals with the collected shared association data. It sends a recovery message from a client to the service database in a similar service space to demand execution of recovery of a service database. Upon failure in recovery, the service database recovery managing component demands another service database managing system to execute recovery of the service database [87]. A directory contains data and the issues identified with directory are comparable to the database position issue examined.

3.1.6 Distributed Database Recovery Management

In a DDBS, failures amidst a transaction processing cause the system to provide below par performance leading to an inconsistent database. Distributed recovery is more complicated than centralized database recovery due to frequent failures [88]. While existing recovery components on incorporated frameworks can be utilized to deal with the previous sort of disappointments, the last kind is increasingly entangled to manage. To guarantee exchange atomicity, distributed submit conventions have been proposed. These incorporate Two-Phase Commit and its variations and Three-Phase Commit. Along these lines, each site knows the execution status of an exchange preceding the failures, and can decide whether an exchange is submitted or not before making the essential move. Figure 3.5 shows the distributed recovery management process [89].

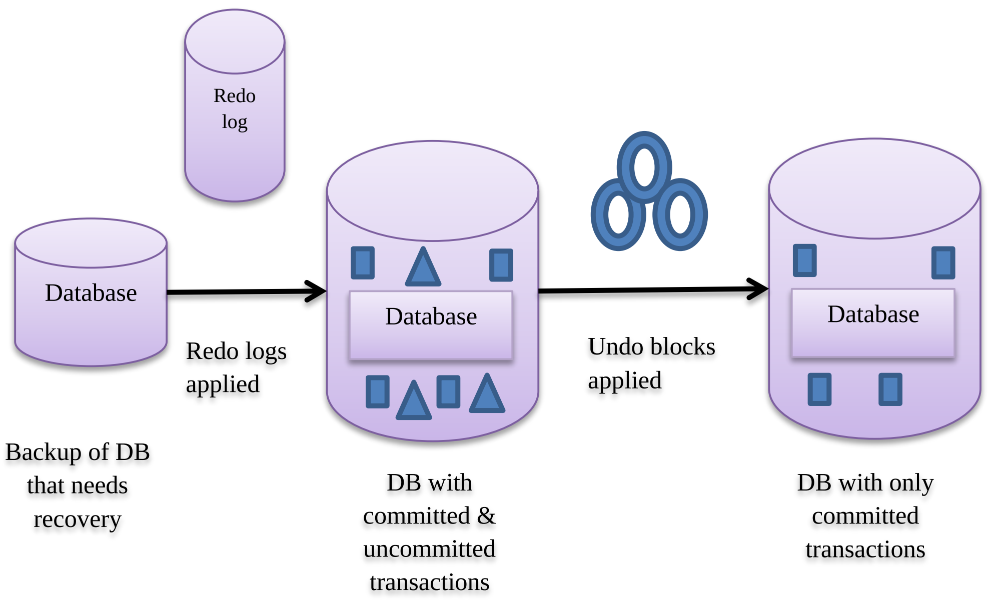

Redo log
Database
Redo logs applied
Backup of DB that needs recovery
Database
DB with committed & uncommitted transactions
Undo blocks applied
Database
DB with only committed transactions

Figure 3.5 Distributed database recovery management

In DDBS, log-based techniques like write ahead logging (WAL) scheme are used for durability. The log files are maintained in these WAL scheme to record, rework and update operations to restore the database to a consistent state.

3.2 Distributed Database Management Systems

A brought together DBMS is a composite programming package that enables a venture to regulate its data on a solitary machine. For instance, usually for a venture to claim a centralized computer database that is constrained by IBM DB2 and a couple of other databases constrained by Oracle, Microsoft SQL Server, Sybase, or different sellers. In some cases, clients need to get to data in some other workgroup's database, or even in the bigger, undertaking level database. DDBMS enables utilizing communication capacity to organize their activities in case of shared admission to the enterprise data [11]. The way that data is distributed over various PCs and constrained by various DBMS items is totally avoided the clients. Clients of such a framework will access and utilize data as though the data were locally accessible and constrained by a solitary DBMS.

3.2.1 Based on components

The components of the system are definite together with the interrelationships between components. These are generally good for design and implementation of the system. A DBMS consists of a number of components, each of which provides some functionality. It might be difficult to determine the functionality of the system from its components. The Computer Corporation of America (CCA) developed “straw man" architecture of reference model (RM) to categorize DBMS-related components into both internal and external components for integrating multiple data models [90].

3.2.2 Based on functions

The different classes of users are identified and the functions that the system will perform for each class are defined. The system specifications within this category typically specify a hierarchical structure for the user classes. The ISO/TC is an example of this RM for DBMS standardization [91]. The flexibility of functions may allow the phased transition from existing implementations to DBMS standards.

3.2.3 Based on data

The different types of data are recognized, and an architectural outline is quantified for the serviceable units that will understand data patterns. This approach also referred as the data logical approach and is desirable choice for normalization activities. The ANSI/X3/SPARC DBMS framework is an example of this type of RM [90].

3.3 Data Distribution Space

3.3.1 Autonomy

Autonomy designates the circulation of control of the DDB and the degree to which each component DBMS can operate autonomously. Major autonomy concepts in distributed database include design, communication and execution autonomy [92].

3.3.1.1 Design autonomy

The sites vary due to data models, physical design, data definition and manipulation languages, query processing strategies, concurrency control, recovery mechanisms etc. because of the low knowledge of intended interconnections.

3.3.1.2 Communication autonomy

The sites can communicate with the other sites on their own whenever the need arises based on bandwidth availability and data locality.

3.3.1.3 Execution autonomy

Each site also performs the execution operations on their own and all the transactions will be treated equally. Thus, the site supports a global transaction processing.

3.3.2 Distribution

Distribution denotes to the physical sharing of data over several sites or server. When distribution of data is initiated, there are two approaches of distribution.

3.3.2.1 Client-Server distribution

Data are focused on the server, while clients offer application environment/user interface [94]. The data processing is divided into distinct parts. A part is either requester (client) or provider (server).

3.3.2.2 Peer-to-peer distribution

In this approach, there is no distinction between client and server machine with each machine having full DBMS functionality. Each peer performs both as a client and a server for informing database services [95]. A peer-to-peer distribution system is perceived as an assembly of self-governing local warehouses.

3.3.3 Heterogeneity

Heterogeneity occurs in several forms in distributed systems, extending from hardware heterogeneity and differences in networking protocols [96].

Heterogeneity in query languages occurs both in hardware as well as software protocols for communication. In a distributed system, the transaction manager should convey the decision to commit to all the servers in the various sites where the transaction is being executed and uniformly enforce the decision [4].

3.3.3.1 Queries

DDBMS Query Processing is required to solve a specific problem which may be distributed at various sites. The various factors [97] which effect the query optimization are: 1) I/O Access: It involves accessing the physical data stored on disk. 2) CPU Processing time: It is associated with the transmission of data among nodes in distributed database systems. 3) Communication Cost: It is linked with the handling overhead of dealing distributed transaction. Query Optimization will be used to state policies envisioned to increase the effectiveness of query assessment procedure.

3.4 DDB Design

The design of a distributed system involves making decisions on the placement of data and programs across the sites of a computer network, as well as possibly designing the network itself [5].

3.4.1 Design Problem

In DDBMS, the distribution of applications involves Distribution of the DDBMS software and Distribution of applications that run on the database. In terms of the level of sharing, there are three possibilities [5]. The two fundamental design issues are fragmentation, the separation of the database into partitions called fragments, and distribution, the optimum distribution of fragments.

3.4.2 Design Strategies

Top-Down Design Process: This design is a process of creating data models that contain high-level entities and relationships, to which successive refinements are applied, in order to identify the corresponding low-level entities, relationships and attributes. The top-down approach is illustrated by using the concepts of the entity-relationship model as in Figure 3.6, [98].

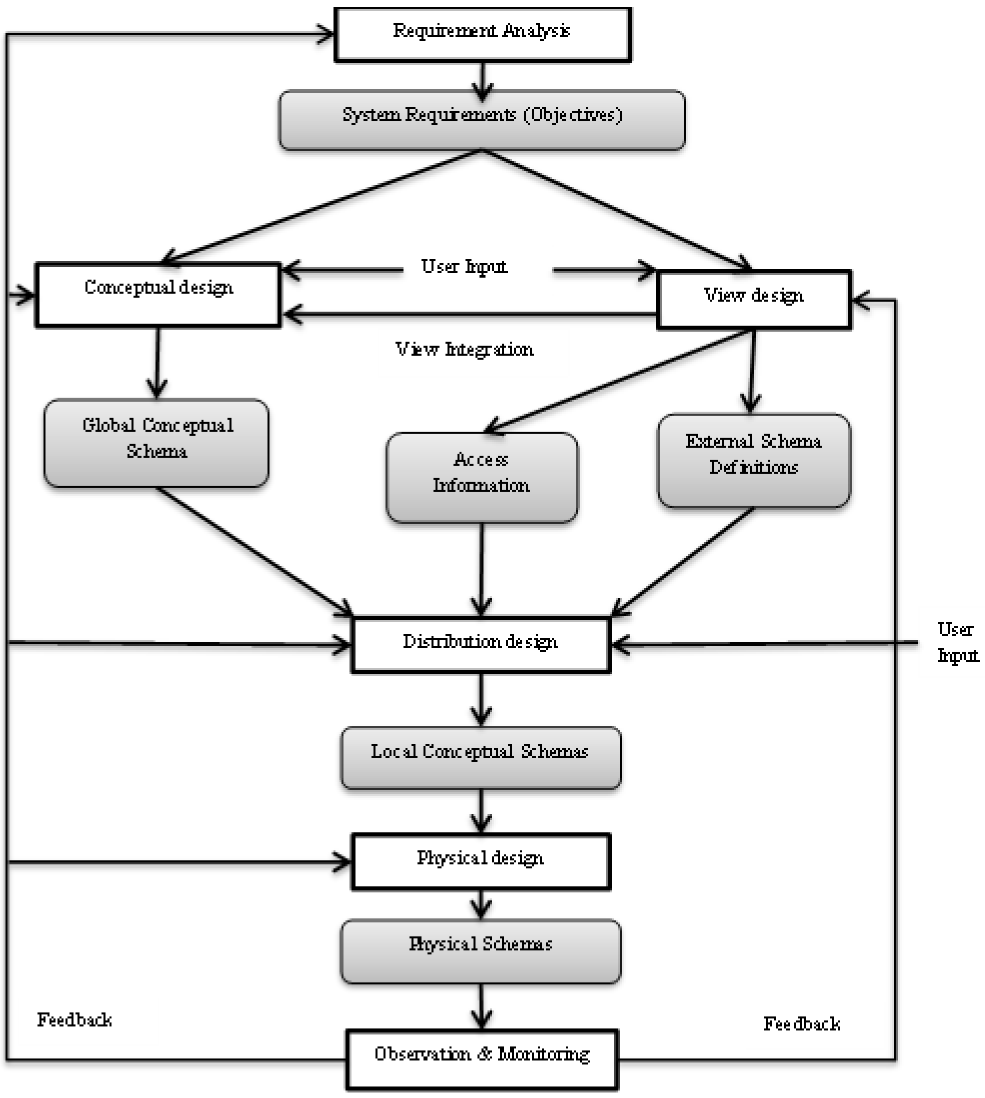


Figure 3.6 Top-down design process

Bottom-up design process: This approach starts from the fundamental level of attributes, which are grouped in relationships that represent entity types and associations between them, as a result of analyzing associations between attributes. The normalization process constitutes a bottom-up approach. Normalization implies identifying attributes and placing them in normalized tables, based on functional dependences between attributes. Figure 3.7 shows the bottom-up design process [99].

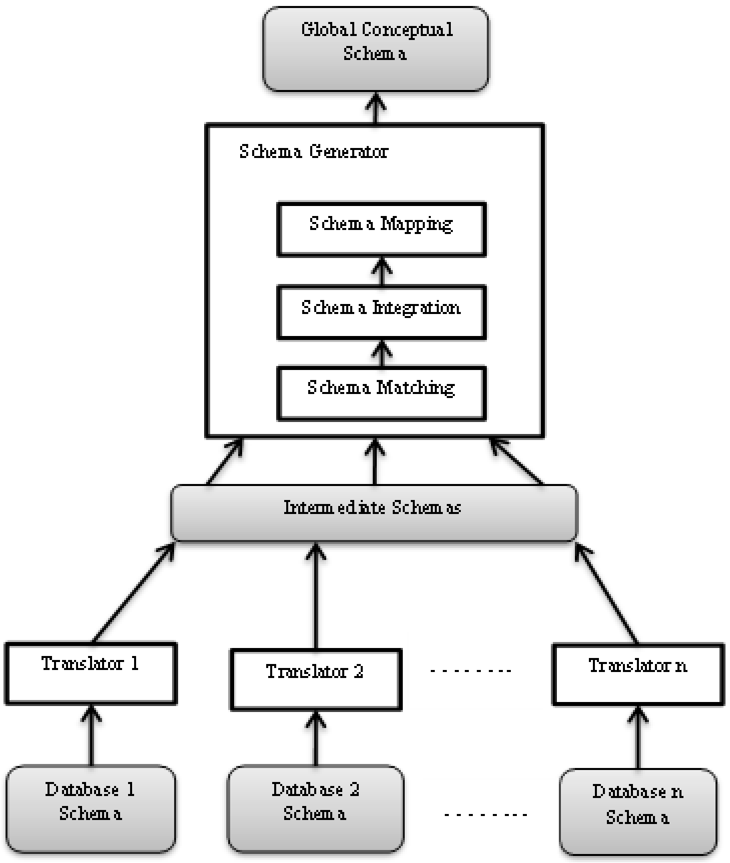


Figure 3.7 Bottom-up design process

Bottom-up approach consists of the following steps:

- Choosing a common data model for describing the global schema;
- Translation of each local scheme intro a common data model;
- Integration of the local schemas into a global schema.

3.4.3 Fragmentation

Data fragmentation is the main challenge in DDB design which is essential for allocation and replication of the fragments in different sites of the distributed system. Fragmentation is essential to divide a database into partitions without any loss of information [100]. This reduces the amount of irrelevant data accessed by the applications of the database, thus reducing the number of disk accesses. Data fragmentation strategies are based at the table level and comprise of dividing a table into logical fragments. There are three types of fragmentation namely Horizontal fragmentation [101], Vertical fragmentation [102] and mixed fragmentation, a combination of horizontal and vertical fragmentation techniques are used [103].

3.4.4 Allocation

The fragment allocation design is an essential issue that improves the performance of the applications processing in the DDBs. DDBs sites allocation is to reduce the communication cost during the applications execution and handle their operational processing [104]. The fragments are allocated based on the data allocation and query processing cost functions [105]. This method attempts to minimize the communication costs by distributing the global database over the sites and increasing availability and reliability where multiple copies of the same data are allocated [106].

3.4.5 Replication

The most important goal of networking is to provide high reliability by having alternate sources of supply. All files could be duplicated on two or three machines, so that, if one of them is not available (due to a hardware failures), the other copies can be used. By storing the copies of data on processors, the probability that at least one copy of the data will be accessible, increases. Supporting failure free operations requires maintenance of copies of data at more than one site. This approach is known as the data replication. The main advantage of distributed database system is that, possibility of supporting data replication as well as data distribution. In the same light, a specific area of new advance is that of replicated database management systems (RDMSs) that offer potential benefits, such as reliability, increased availability and growth possibility [107].

Though replication enhances the reliability and data availability for read only access, increased number of data copies imposes considerable overhead for processing the update transactions. The principal goal of replication control mechanism is, to guarantee that updates are applied to copies of replicated data. The task of transaction management leads to adoption of concurrency control, crash recovery and network partition handling in a replicated database environment. Replication management techniques are needed to resolve such scenarios. They can be divided as centralized and distributed [108]. Distributed replication is of two types and three schemes. The types are synchronous replication and asynchronous replication. The replication schemes utilized currently are No replication, Partial replication and full replication schemes.

3.5 Secure DDBMS

Database security is the practice of ensuring that the data stored in the database is safe from unauthorized access and it is available for use when needed. Privacy is the appropriate use of information, and is the true objective of security. For systematic security, a security plan should be developed. The first step in developing the plan is the identification of fields to be protected. Next step identifies the weak points that compromise the database security [109]. This step is followed by risk elimination/ reduction and the actual implementation and monitoring of the system. The attackers or intruders, who intrude the data, are broadly classified as:

Internal Attackers: Fired or unhappy employees usually play a major role in being the internal attackers, as they know major systems and internal relations between systems. These are the most dangerous and the attacks are hard to trace.

External Attackers: The external attackers do it just for fun or for getting monetary benefits. The main objectives of such attackers are disclosure of critical information or alter the critical data.

The functional areas are Security policies, Security mechanisms, and Security system assurance. Security policies are broadly classified into two categories: Discretionary Access Control (DAC) and Mandatory Access Control (MAC). Security mechanisms implement the security policies [110]. A security mechanism prevents any security violation from occurring during the operation by the use of access control mechanisms. Security requirements are classified into the categories of Identification, Authentication, Authorization, Access Controls, Integrity, Consistency, and Auditing. Security System Assurance provides steadiness and integrity of the security mechanism.

4
Data Fragmentatıon in DDBs

4.1 Introduction

Data fragmentation is considered to be the most vita challenge along with data allocation. The main reason for the development of fragmentation techniques is that the large databases are usually harder to store and process. But the issue in fragmenting data lies in the fact that the data can be divided unevenly or with the loss of important details [38]. In some cases, the fragments are distributed unevenly in different location which reduces the relational structure of databases. To resolve these data issues, DDBMS allows for load balancing in which the data can be distributed according to the capacities of servers and higher availability.

However the perfect fragmentation technique is still needed to be developed for initial database designs [111]. The distributed nature of processors or sites in DDBS, there needs to be more compatible techniques and hence the research is open. The use of clustering algorithms for data fragmentation has been studied in the recent past with techniques based on k-means clustering [112]; latent discriminant analysis, etc. have been employed [34]. Also some of the machine learning techniques has also been employed for data fragmentation. Along these lines of research, the concept of utilizing hierarchical agglomerative clustering (HAC) for data fragmentation is suggested in this chapter. The hierarchical agglomerative clustering is conceptually and mathematically simple approach and has been applied for clustering in many domains successfully. The model of HAC also provides flexibility and scalability on the available data. In this chapter, the traditional HAC model is tailored in order to efficiently fragment the data in a distributed manner. The improved HAC intended to be presented in this chapter has utilized Silhouette index as additional similarity metric along with the Jaccard coefficient.

4.2 NEED FOR DATA FRAGMENTATION TECHNIQUES

Data fragmentation is an essential process in any distributed system as the main objective of distributed systems is to distribute the massive data, which is generally increasing with time, to different locations. When there are such massive data, the procedure in how to divide and distribute them effectively is anticipated. There arises the need for data fragmentation and it resolves this data distribution issue along with the data location process [113].

In many systems, the data handling has been a major problem with the end result of inefficient memory usage becoming a concern. This automatically reduces the capacity of the processor and sometimes causes the processor to halt its operations. The main preliminaries maintained in larger organizations are that all the systems associated are allocated the same degree of storage irrespective of the degree of processing. Even the degree of data distribution varies in such systems but the limitation in storage capacity becomes the major factor decision. At these scenarios, the most effective and viable solution is to utilize the distributive nature of DDBs and distribute the data to available processors. But before distribution, the data has to be divided or fragmented into smaller and effective packages to ensure effective distribution. This leads to the research of developing efficient fragmentation techniques [100].

Fragmentation allows the databases to easily utilize the data and increases the efficiency of query. It also improves the security of data by not exposing complete data to the users. The reliability is also improved while parallel processing is possible due to the data distribution. Another important aspect is that fragmentation allows the DDBs to ensure balanced storage. The important factor in deciding the fragmentation technique is the nature and capacity of the processors that handle these data. Based on these parameters, many techniques have been developed. Recently the data clustering techniques have been employed for fragmentation as both the concepts are quite similar and require only minimal modifications. However the choice of utilizing such clustering techniques must be carefully undertaken in order to reduce complexities. The proposed model utilized an improved HAC based clustering approach for enable data fragmentation which has been suggested for effective data distribution.

4.3 IMPROVED HIERARCHICAL AGGLOMERATIVE CLUSTERING FOR DATA FRAGMENTATION

The proposed model of data fragmentation utilizes the improved HAC (IHAC). The idea of the IHAC is to cluster the data using the item-based clustering approach based on the traditional HAC algorithm. Many existing clustering systems use term-based clustering approach [114]. They construct data representation matrix based on the data frequency. According to data matrix, they choose the right similarity measures and then calculate similarity between clusters. The right choice of similarity measure and clustering algorithm is very important for clustering process. Moreover, constructing data matrix based on term-based or item-based approach is the main part of the clustering because this can affect the accuracy of clustering [115]. Unlike existing systems, the proposed IHAC method constructs data matrix without considering counts of data. By neglecting the data counts, the proposed method can merge the similar clusters into the same cluster efficiently than the existing systems with less processing time. Additionally, the Silhouette index is also utilized along with Jaccard index for similarity estimation.

4.3.1 Hierarchical Agglomerative Clustering

HAC algorithm offers informative descriptions and visualization in the data fragmentation due to the real ordered relationships of data [116]. Input data sets for HAC are the data objects that are needed to be split into clusters based on their similarities. The data attributes are the properties of the components such as the location of data, its size, the data connectivity, or other features which are categorized into quantitative data and qualitative data. Then the similarity coefficient is utilized to identify the similar data that are needed to be grouped together. Jaccard coefficient or Jaccard index is utilized for this purpose based on the Euclidean distance [117]. If A and B resembles two data objects, then the Jaccard coefficient is given by

$$Jaccard\ coeffcieint\ of\ data\ A, B = \frac{|A \cap B|}{|A \cup B|} \qquad (4.1)$$

If two data objects have maximum values of Jaccard index, they are grouped together. Likewise the process is continued for all the data objects. This process continues and continues without any limitation. Based on these obtained values, the HAC forms the similarity matrix. Then the HAC is repeated to recognize the minimal coefficient in the matrix and performs the clustering algorithm to allocate the data into a tree structure. In each step, the similar clusters are merged and the matrix is updated using the single linkage, complete linkage, un-weighted pair group or weighted pair group approach.

The outcomes of the HAC algorithm are typically portrayed by a binary tree. The root node denotes the whole data set and each leaf is regarded as a data cluster. The intermediate nodes describe the extent that the data objects are proximal to each other and the height of the tree. The important process in cutting the HAC tree is by restricting the level of merging to limit the size of clusters. For this purpose, a threshold is determined usually based on the number of clusters, or cluster density. In modern researches, this threshold is also optimally determined. Based on this threshold, the size of cluster is maintained to avoid oversize. Thus the data is now effective clustered together with each cluster resembling a fragment of data. This is exploited in the improved HAC.

4.3.2 Improved Hierarchical Agglomerative Clustering

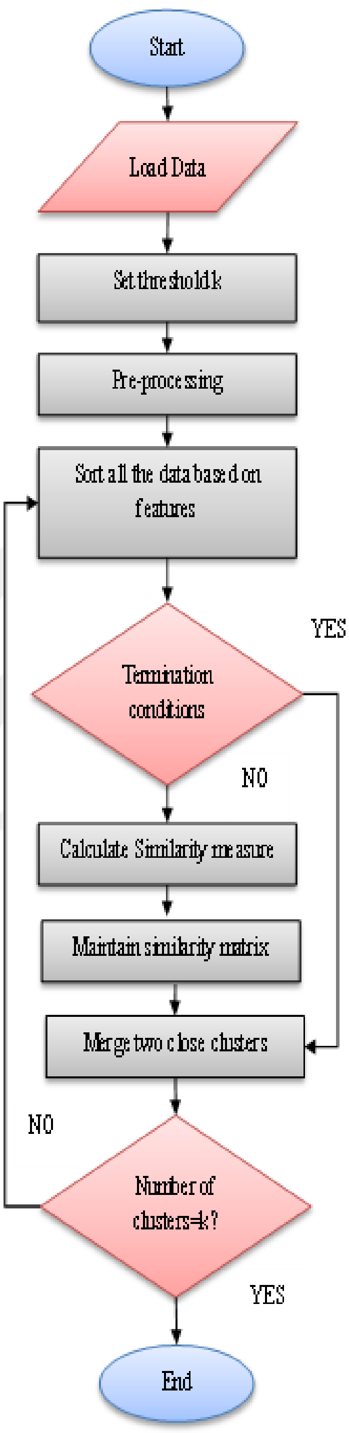


Figure 4.1 IHAC for data fragmentation

The proposed IHAC algorithm constructs the similarity data matrix without considering the data counts. This process enhances the cluster merging process with less processing time. The reduction in this processing time can enhance the data distribution process. Figure 4.1 shows the process of this proposed IHAC modeled based on HAC.

Initially, the data objects are assigned with the features of data to determine the similarity. Then the similarity matrix is formed at each of the data agents assigned to perform this task. Each of the agents has its own similarity matrix by comparing the data with its neighboring data objects. As each data now has its own knowledge about the most similar data object, the grouping takes less time.

The proposed method shown in Figure 4.1 mainly consists of three phases: data collection, data preprocessing and clustering process. In the proposed method, dataset which need to be clustered are loaded. After that, those data are pre-processed. After preprocessing data, the IHAC algorithm is performed. In IHAC algorithm, the number of cluster, k, is specified. After that data are sorted and the clusters are formed based on Jaccard coefficient and Silhouette index [118]. Silhouette index is given as the difference between the highest and lowest average dissimilarity by the maximum similarity. It is expressed as

$$Silhouette\ index = \frac{Highest\ average\ dissimialrity\ (b) - Lowest\ average\ dissimilarity(a)}{\max(a,b)} \tag{4.2}$$

Where a denotes the lowest average similarity while b is the highest average similarity.

This similarity score is employed to build the similarity matrix for each data agent who contains a single data or a group of data as required for the overall size of the database. As each data now knows its most closer it aligns itself with it. If the data are similar, they are clustered together. The same approach is performed for each cluster of data. Based on the similarity scores, the two most similar clusters are merged into the same cluster. But the merging of clusters is performed in such a way the number of clusters k is not overlooked. In this way each of the data is either formed into a new cluster or an available cluster with similar data. The complete process of IHAC is given in Algorithm 4.1.

Algorithm 4.1. IHAC

```
Input: N Data objects with features (location, size, data connectivity)
Output: Optimal clusters
Initialize the dataset
Set threshold k (number of clusters)
For all data objects n
    Sort all data based on the features
    Form initial data clusters
    Compute similarity scores (Eq.4.1 and 4.2) for each data object n
    Construct similarity matrix for each data object n
    If $(X \cap Y) = X \ or \ (X \cap Y) = Y$ then
        Group data X and Y
    Else $X = X + 1$
    End if
    Calculate similarity measures of clusters
    Merge the most similar (closest) clusters
    Repeat until k-clusters
End for
```

The efficient clustering is achieved with IHAC algorithm. It is executed especially at the case of large datasets. The clustering results illustrate the source for distribution of the whole data sets and support supervised classification of datasets centered on their selection of features. Thus, groups are employed to choose optimal and useful features also subsequently to add into sample datasets to improve the performance of data distribution in clustering process. The outcomes of the IHAC algorithm are usually depicted by a binary tree as in HAC. The root node denotes the whole data set and each leaf is regarded as a data cluster. The intermediate nodes define the degree that the data objects are proximal to each other and the height of the tree. The data clusters (nodes) are the data fragments aimed in the objective of data fragmentation. These fragmented data can be distributed evenly based on data allocation process with minimized processing time. Thus IHAC increases the convergence speed of the fragmentation approach.

4.3.3 Compatibility for distributed data fragmentation

The compatibility of IHAC for data fragmentation is quite simple as initially described, both the clustering and fragmentation process are quite similar. As mentioned above, the final clustering results of IHAC are provided in the form of a binary tree where each leaf node form a data cluster. This data cluster is actually a fragment of the complete data stored at the root node. Likewise this output tree contains k number of leaf nodes with each representing a data fragment.

As the inclusion of the second similarity measure (Silhouette index) improves the accuracy of determining the similarity between data objects, the distributive nature is preserved by the introduction of the data individuality concept. This concept is the creation of data similarity matrix for individual data objects or data agents based on its own sense of local knowledge about neighboring data objects. Thus the primary objective of extending the IHAC to data fragmentation in the DDBs has been accomplished successfully.

4.4 EVALUATION

For the evaluation of the proposed IHAC based fragmentation approach, the original dataset is determined and then the processing is performed in Hadoop environment. The performance of the IHAC fragmentation approach is compared with that of those using k-means, k-means rough clustering, etc. in terms of time and maximum memory usage, the IHAC outperformed the other models. The complete description and comparison of these results are presented in Chapter 7.

5
Data Allocation in DDBs

5.1 Introduction

The concept of distributed database has gathered immense attraction from the research community. However the designing of the distributed databases is considerable the most complicated problem. As discussed in Chapter 1, the major problems in DDB designing are the classical centralized design issues, fragmentation, allocation and replication. Data allocation problem is analyzed in this chapter. In the static environment, optimum solution for data allocation problem can be obtained through static data allocation in which the retrieval and processing of queries from different processors to fragments certainly do not change. However, in a dynamic scenario where these access patterns vary over time, the static allocation solutions would degrade the performance of DDBs and in dynamic environment, reduces the data transmission cost.

The data allocation problem (DAP) is an NP-Hard problem and also it is quite similar to the Quadratic Assignment Problem (QAP) [119] and File allocation problem (FAP) [120]. When the data transmissions aspect of the system is considered, the DAP and FAP significantly the same problems and QAP is quite similar to that of DAP. Both DAP and QAP has main aims to provide the locality of the data. Hence the DAP can be formulated based on QAP and vice-versa. In this chapter, the Dap problem is modelled into an optimization problem based on the QAP concept and it is intended to be resolved using chicken swarm optimization while the performance of the proposed scheme is compared with that of existing optimization algorithms like Particle swarm optimization, ant colony optimization, etc. [121].

5.2 OPEN ISSUES IN DATA ALLOCATION FOR DISTRIBUTED DATABASES

The main issue with DAP in distributed databases is that it is an NP-Hard problem and hence the problem has to resolved only in polynomial time [55]. In distributed databases, this issue can be handy as the polynomial time is not restricted. Hence the solution can be achieved mostly through modelling DAP into search problems or optimization problems. This calls for determination between these two variants of solution.

The other major issue with DAP solution is that most DAP solutions are framed and tested in static environments which are quite incompetent in the dynamic scenarios. Hence the researchers have suggested the utilization of solutions that are suitable in both static and dynamic scenarios as most systems prefer dynamic but at some analysis stages prefer static scenario. Apart from these issues, the solutions to DAP problem has other issues in the form of grouping clusters, cost, disk drive speed, parallelism of the queries, network traffic, load balancing of servers, etc. These issues are not always prominent as these are generally taken care at the initial stages of DDB design but are also capable of degrading the performance when not considered.

5.3 CHICKEN SWARM OPTIMIZATION BASED DATA ALLOCATION FOR DISTRIBUTED DATABASES

The proposed model of data allocation utilizes a swarm optimization model named Chicken Swarm optimization (CSO) for allocating the data fragments in the distributed database systems. The DAP problem is modelled into the optimization problem and resolved by this CSO algorithm.

5.3.1 Chicken Swarm Optimization

CSO algorithm has been developed by [122] based on the behavior of the chicken flocks. The dominant chickens would be almost steady with the head roosters to search for food. The accommodating ones, be that as it may, would reluctantly remain at the fringe of the group to look for food. Every chicken is excessively straightforward, making it impossible to coordinate with each other. However considering them as swarms, they may arrange themselves as a group to scan for food under particular hierarchal order. This swarm insight can be related with the objective issue to be upgraded, and another algorithm has been composed based on this concept.

The CSO algorithm is formed based on this concept [123]. In the initialization phase, considered the chicken swarm of size X and each rooster acts as an agent of its own cluster in virtual search space. The agent is responsible for the movement of chicken and the position update. Assume *RX, HX, CX and MX* indicates the number of the roosters, the hens, the chicks and the mother hens, respectively. The best *RX* chickens would be assumed to be roosters, while the worst *CX* ones would be regarded as chicks. The rest are treated as hens. All *X* virtual chickens, depicted by their positions $x_{i,j}^{t}$ $(i \in [1, \dots, X], j \in [1, \dots, D])$ at time step *t*, search for food in a *D*-dimensional space. In this algorithm, the optimization problems are the minimal ones. Thus the best *RX* chickens correspond to the ones with *RX* minimal fitness values. The roosters with better fitness values have need for food access than the ones with worst fitness values. This can be evaluated using Eq.5.1 and Eq.5.2.

$$x_{i,j}^{t+1} = x_{i,j}^{t} * (1 + Randn(0, \sigma^2)) \tag{5.1}$$

$$\sigma^2 = \begin{cases} 1, & if\ f_i \le f_k, \\ \exp\left(\frac{(f_k - f_i)}{|f_i| + \varepsilon}\right), & otherwise, \end{cases} \quad k \in [1, X], k \ne i \tag{5.2}$$

Where $Randn(0, \sigma^2)$ is a Gaussian distribution with mean 0 and standard deviation σ^2. ε, which is used to avoid zero-division-error, is the smallest constant in the computer. *k*, a rooster's index, is randomly selected from the roosters group, *f* is the fitness value of the corresponding *x*.

With respect to the hens, they can take after their group-mate roosters to look for food. Additionally, they would likewise arbitrarily take the great food found by different chickens; however they would be quelled by alternate chickens. The more dominant hens would have advantage in going after food than the more resigned ones. These phenomena of hen can be formulated mathematically as follows.

$$x_{i,j}^{t+1} = x_{i,j}^{t} + S1 * Rand * \left(x_{r1,j}^{t} - x_{i,j}^{t}\right) + S2 * Rand * (x_{r2,j}^{t} - x_{i,j}^{t})$$
(5.3)

$$S1 = \exp(f_i - f_{r1}) / (abs(f_i) + \varepsilon)) \quad (5.4)$$

$$S2 = \exp(f_{r2} - f_i) \quad (5.5)$$

Here *Rand* is a uniform random number over [0, 1]. $r1 \in [1, \dots, X]$ is an index of the rooster, which is the ith hen's group-mate, while $r2 \in [1, \dots, X]$ is an index of the chicken (rooster or hen), which is randomly chosen from the swarm. $r1 \neq r2$.

The more dominant hens would be more likely than the more submissive ones to eat the food. The chicks move around their mother to forage for food. This is given by

$$x_{i,j}^{t+1} = x_{i,j}^{t} + FL * (x_{m,j}^{t} - x_{i,j}^{t}) \quad (5.6)$$

Where $x_{m,j}^{t}$ stands for the position of the ith chick's mother $(m \in [1, X]$. FL $(FL \in (0,2))$ is a parameter, which means that the chick would follow its mother to forage for food. Consider the individual differences, the *FL* of each chick would randomly choose between 0 and 2. The convergence metric is utilized to evaluate the optimization model whose result showed that the CSO based clustering has better convergence.

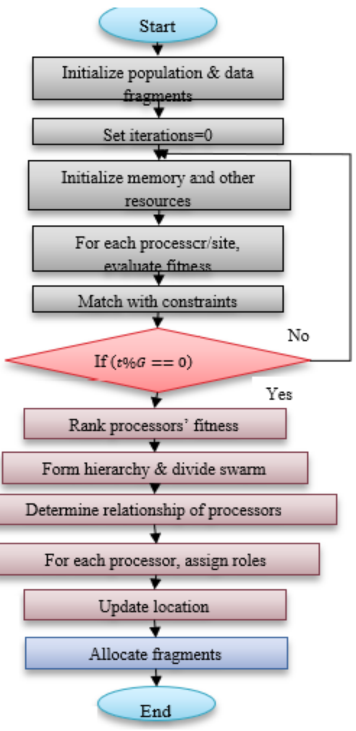

Start
Initialize population & data fragments
Set iterations=0
Initialize memory and other resources
For each processor/site, evaluate fitness
Match with constraints
If (t%G == 0)
No
Yes
Rank processors' fitness
Form hierarchy & divide swarm
Determine relationship of processors
For each processor, assign roles
Update location
Allocate fragments
End

Figure 5.1 CSO for data allocation

5.3.2 Data allocation problem

The major objective of the DAP is to analyze and identify the best processors for the placements of data fragments with minimum total transaction cost and less delay to the users. The DAP is modelled based on two kinds of dependencies between transactions and fragments namely direct and indirect transaction-fragment dependencies. The dependency of processors to fragments can be inferred from transaction-fragment and processor-transaction dependencies. Figure 5.1 shows the dependencies of transactions on fragments and processors on transactions on distributed database systems [60]. S represents processors, F represents fragments, T represents transactions, freq is the number of requests for the execution of a transaction at a processor, trfr is the direct dependency of a transaction on a fragment, and q is the indirect dependency of a transaction on two fragments.

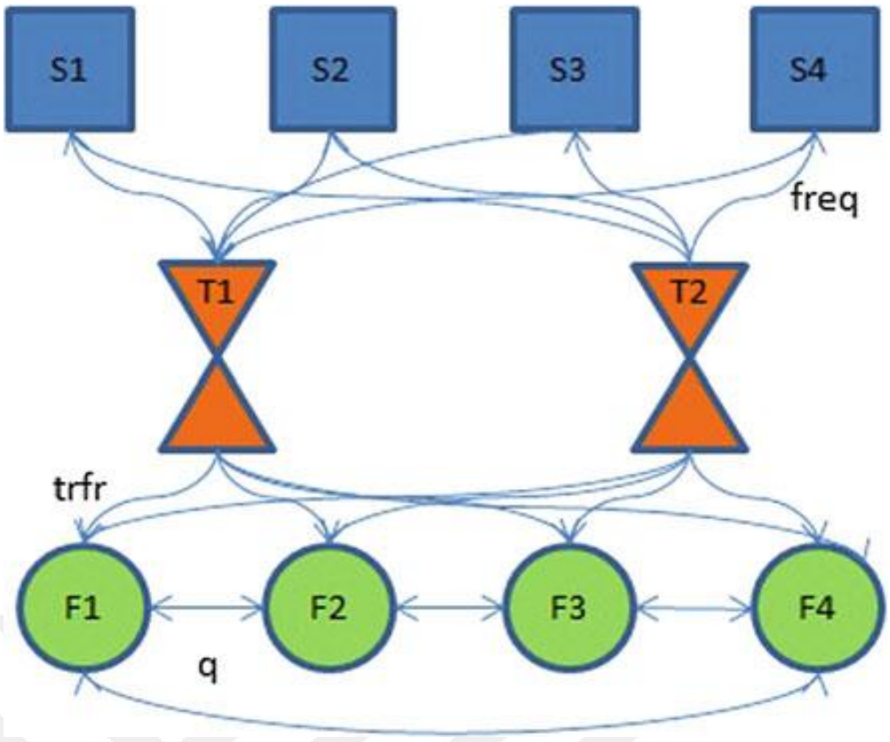


Figure 5.2 Dependencies of transactions on fragments and processors on transactions

The cost function of the DAP is modelled as the sum of two costs, direct and indirect transaction-fragment dependencies [47]. The total cost of data allocation $Cost$ is the sum of two costs $Cost$ 1 and $Cost$ 2.

$$Cost\ (\Phi) = Cost\ 1(\Phi) + Cost\ 2(\Phi) \qquad (5.7)$$

Here Φ denotes the m element vector wherever Φj specifies the processor to which $frag$j is allocated. $Cost$ 1 is denoted by the volume of processor-fragment dependencies which can be expressed by the product of two matrices 1) Matrix that stores processor fragment dependencies (*stfr)* and 2) Matrix that stores the unit communication cost among the processors (*uc)*. The cost of loading a fragment $frag$j in processor pi is denoted by partial cost matrix $pcost1n \times m$. The unit partial cost matrix is expressed as

$$pcost1_{ij} = \sum_{q=1}^{n} \mathrm{uc}_{iq} \times \mathrm{stfr}_{qj} \qquad (5.8)$$

Based on these parameters, the $Cost$ 1 can be computed by evaluating the unit $pcost1_{ij}$ for each i and j.

$$Cost\ 1(\Phi) = \sum_{j=1}^{m} pcost1_{ij} \qquad (5.9)$$

Similarly, for the computation of $Cost$ 2 the inter-fragment dependency matrix (*ifdm)* is utilized. The *ifdm* matrix representing the inter-fragment dependency is the multiplication of the matrix *qfr* with the matrix *q*. It is given by

$$ifdm = qfr_{l\times m\times n} \times q_{l\times m\times n} \qquad (5.10)$$

Where matrix *qfr* represents the execution frequencies of the transactions and *q* denotes the indirect transaction fragment dependency. Based on this matrix, $Cost$ 2 is derived.

$$Cost\ 2(\Phi) = \sum_{j1=1}^{m} \sum_{j2=1}^{m} \mathrm{ifdm}_{j1j2} \times \mathrm{uc}_{\Phi_{j1}\Phi_{j2}} \qquad (5.11)$$

Hence the DAP can be modelled as optimization problem by combining these cost values.

$$Cost\ (\Phi) = \sum_{j=1}^{m} pcost1_{ij} + \sum_{j1=1}^{m} \sum_{j2=1}^{m} \mathrm{ifdm}_{j1j2} \times \mathrm{uc}_{\Phi_{j1}\Phi_{j2}} \qquad (5.12)$$

5.3.3 CSO algorithm for DAP solution

According the literature study, it has been found that PSO has already been utilized for the solving DAP [60] due to the fact that PSO has fewer control parameters, speed convergence characteristics and lower consuming time, robustness. However, there are many optimization algorithms especially swarm optimization models that are way better than PSO. CSO has been found to be more effective for the DAP solution due to the hierarchical order of the problem formation.

Initially, each of the processors or processors is determined where the fragments are needed to be positioned. For minimizing the total cost of transaction, fragments are positioned at the best processors. The process of CSO into DAP problem is given in algorithm 5.1.

Algorithm 5.1: CSO based DAP solution

```
Initialize the population of N processors; data fragments frag; memory mem;
Iteration iter=0;
Compute fitness f of N processors
$f_{ij} = Cost\ (\Phi)_{ij}$                                                    (5.13)
While (iter<max_generation)
        If ($iter\%G\ == 0$)
                Rank processors fitness values;
                Sort the processor set in hierarchical order;
                Cluster the sorted processor set into groups;
                Determine processors' relationship (server, sub-server or
        client);
        End if
For $i = 1{:}N$
        If $i == server$
                Update location using Eq.(5.1);
        Else if $i == sub - server$
                Update location using Eq.(5.3);
        Else if $i == client$
                Update location using Eq.(5.6);
        Else go to initialize process;
        End if
Evaluate new solutions;
Allocate frag based on new sorted solutions;
If new solution > existing solution
```

```
        Update solution;
End for
End while
```

To resolve DAP, the fragment placement must be optimally determined. In the proposed CSO based DAP model, the processors are initialized as chickens and the parameters are defined. The capacity of each processor is analyzed prior to applying CSO to ensure there is no mistakes in the fitness value calculation. The fitness for each processor is the cost function which is computed based on Eq.5.12. Based on these fitness values, the N processors are sorted in the most dominating order. Each processor has to take either the roles of server, sub-server or clients which means the worst fitness processors are designated as clients. Based on this order, the clusters are formed with only one server, more than one sub-server and many clients. This clustering process act as determinant in the locations for placing the data fragments. At each iteration, the best location is determined by the food location expressions in CSO. Based on this, the location for placing these fragment allocations are finalized and the order is updated. This solution acts as the DAP resolution until the next iteration. At the maximum iterations, the best location for each fragment is finalized with minimum transaction cost.

5.4 EVALUATION

For the evaluation of the proposed CSO-DAP approach, the original dataset is determined and then the processing is performed in Hadoop environment. The performance of the CSO-DAP approach is compared with that of those using Genetic algorithm (GA), PSO, ACO, etc. in terms of cost and time, the CSO-DAP outperformed the other models. The complete description and comparison of these results are presented in Chapter 7.

6
Data Replicatıon in DDBs

6.1 Introduction

Database replication is a very increasing research topic with the increasing volume of dynamic web content. It has been traditionally employed for increasing the availability and performance of data through the reconfigurable connections and diminishing the need to frequent access of remote servers or sites [124]. There have been numerous works for developing efficient database replication protocols which provided data availability, consistency and fault-tolerance. However these protocols were not been used as commercial products since the database designers believe most of these protocols are complex, perform poor and lower scalability for non-feasibility [125]. This has led the researchers to focus on developing replication protocols which enhances the compromise between the correctness and fault-tolerance as well as assuring availability and performance.

The gap between the replication theory and practice was wide and in addition to the performance, consistency and availability, the communication issues have overtaken the research core. Efficiency can be achieved only when the communication cost is minimal and the overhead is small [126]. Therefore it is a necessity that the database replication strategies should also include communication issues in the cost analysis and complexity analysis. This chapter focuses on developing an efficient data replication algorithm. First the need for replication is described and the types of replications are explained. Snapshot replication and merge replication strategies are presented and then the multi-objective glow-worm optimization based data replication strategy is explained in this chapter.

6.2 NEED FOR DATA REPLICATION

The main reason for requiring the data replication is the efficient availability and fault tolerance in the web services where large amount of dynamic web data are requested every time. The motivation for requiring data replication strategies can be due to its efficient performance based advantages. The major need is to improve the performance of the distributed system. In this aspect, the vital issue is the unwanted replication of resources due to the caching of resources from servers to the browsers and web proxies to increase availability. This process is not an issue when only read only data is replicated but the overhead incurred for replicating resources for write data is large. Hence only optimal resource allocation is maintained by introducing replication which is less cost and also improves the performance of the system significantly [127]. There should also contain a limit in replicating the frequently updated data and achievable benefit of replication.

The second factor influencing the replication is the availability [128]. As the expectation of the users is always higher for availability, the response time is also expected to be faster. However, in cases without the replication, the issues of concurrency, failure of servers, communication disruption and network partition due to high mobility rate of users can reduce availability. But replication is a perfect solution in these cases of failures and thus increases the availability. This is possible due to the distributed data placement and hence the servers are also more improved in terms of failure independent.

An important factor is the fault tolerance which makes the point even when replication is applied, the correctness is not ensured due to non-adaptability for updates. Hence the data needs to be correct for all users irrespective of the frequency of update and so the replication strategies must also include this feature. This is achieved by utilizing the concept of 2f+1 for byzantine failures [129].

Another important but less considered factor is the transparency [130]. Replication creates a visualization model in which the users believe they are operating on the single data available and hence the data is considered as single set of values. Though their operations are performed on several copies of the same data at different locations, the physical similarity of the data without any copyright issues creates the necessary transparency. Thus these factors have made the need for replication, more promptly a motivation for the data replication strategies.

6.3 REPLICATION TYPES

Database replication in distributed systems support three distinct types based on its purpose. The three types are Snapshot replication, Transactional replication and Merge replication.

Snapshot replication: Snapshot replication is considered to be the simplest type of replication which takes a snapshot of the data on the server and copies it to another server [131]. After the initial synchronization, the data can be refreshed in the published site tables occasionally based on a pre-defined schedule. However in snapshot replication all the data must be copied each time when the table is refreshed. The data, either partially modified or fully modified after update, the same amount of resources are spent in executing snapshot replication each time.

Transactional replication: Transactional replication contains copying data from the publisher to the subscriber once and then delivering transactions to the subscribers as they occur on the publisher. As database users insert, update, or delete records on the publisher, transactions are forwarded to the subscribers. Automatic changes of the replicated data on the publisher and on the subscribers are not needed in transactional replication [132]. Generally the transactional publishers of database servers do not modify the data ad utilize only as read-only purposes. Transactional replication is most useful in environments that have a dependable dedicated network line between database servers participating in replication.

Merge Replication: It is considered as the difficult type of replication as it combines data from multiple sources into a single central database. The initial synchronization are performed similar to the transactional replication i.e. by taking snapshot of data and copying them at subscribers but in merge replication it allows changes of the same data on publishers and subscribers [133]. Once the subscribers get network access, the replication will be detected and combine all changes from subscribers and modify the data at publishers accordingly.

6.4 SNAPSHOT REPLICATION

Snapshot is the point-in-copy stored in the similar volume of subscribers system utilized in registering the status of the complete data at specified times of data storage. The main advantage is that it utilizes only small amount of additional storage and does not have significant impact in the overall system performance. In case of accidental data modification or deletion by the users, the previous time of the snapshot can be restored very easily and quickly. Additionally, the users are also able to recover their modified or deleted data from shared folders without the aid of administrators. This process also ensures the data copy to be stored or transferred easily and also capable of performing even without dependable dedicated network line. Another aspect in the snapshot replication is that one can take manual or on-demand snapshots and schedule the snapshots while performing restore and replicating processes. The scheduling can be specified by regular time intervals to ensure automatically taking snapshots.

Snapshot replication exhibits certain constraints that are not available in other replication types [134]. If the single publisher-single subscriber and one to one business applications are utilized currently then the snapshot replication is recommended. In case the scenario presents single publisher - multiple subscribers with the need of application is for distributing any type of lists, the transactional replication is suggested as it is much convenient. Similarly, if the warehousing applications are utilized where the multiple publishers - single subscriber scenario appears, the transactional replication is suggested. While it is efficient in other cases, for online transactions which utilize the scenario of multiple publishers- multiple subscribers, the merge replication is highly recommended.

For implementing the snapshot replication, three types of servers are required which maintains the history of the replication process (i) *publisher,* (ii) *subscriber*, and (iii) *distributor*. These three servers are employed in a three step initialization process for the snapshot replication. Initially the distributer is created with required constraints followed by the creation of the publishers where the source data are generated. Upon completion of these two processes, the final step of creating the subscribers and subscribing them to the required publishers is performed. The final step is repeated as many times as needed to create all the subscribers in designated organization.

Secondly, a snapshot agent is required for generating the snapshot of the publications which will be kept in a particular folder in the distributor. Figure 6.1 shows the snapshot replication process.

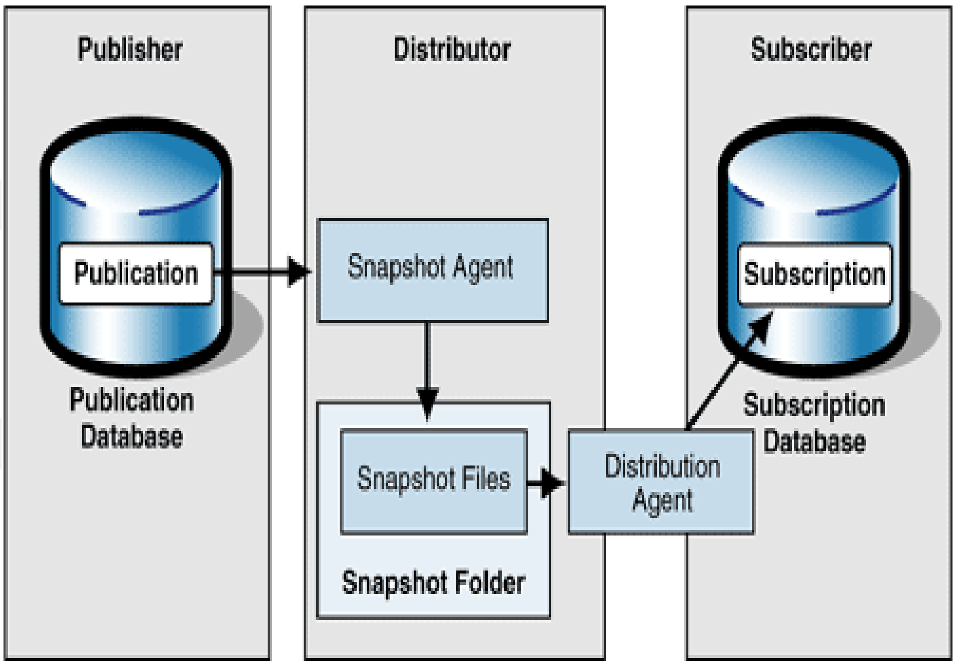


Figure 6.1 Snapshot replication agents and triggers

Snapshot replication operates by evaluation of the published database and constructing files in the working folder on the distributor [135]. These files are named snapshot files and comprise the data from the published database as well as some additional facts that will support generating the initial copy on the subscription server. SQL Server stores configuration and status data in the distribution database however stores all authentic data in the snapshot files.

If the push subscription is deployed, the distribution agent operates on the distributor and forwards the snapshot files to the subscriber. If a pull subscription is deployed, the distribution agent operates on the subscriber. The difference is that in push subscription, the maximum workload is handled by the distributor machine while in pull subscription; the maximum workload is handled by the subscribing machine. Based on the machines deployed, the subscription is selected. Snapshot replication is commonly employed in small organizations which connects using modems, but do not have to merge data. If the organization uses a modem to connect, and only needs to send the data one way, snapshot replication is most preferred.

6.5 MERGE REPLICATION

Merge replication allows changes to be sent from one primary server, called a publisher, to one or more secondary servers, called subscribers. Merge replication is used for distributing data to various servers from a primary server. One important criterion is how frequently changes are made to it. In this scenario, strictly speaking the publisher doesn't look like the primary server, because other servers can also make changes to the data. Such clashes may arise because merge replication does not require a real-time network connection between the publisher and the subscriber. It is possible that one server changes data and later on another server changes the same data to a different value [136]. A merge replication setup includes the following components: i) snapshot agent, ii) triggers, tables, and views and iii) merge agent. The merge replication agents and triggers are shown in Figure 6.2.

The Snapshot agent in merge replication plays the same role as in transactional replication. It creates a snapshot of the data in the publication database, which is used to initialize a subscriber. In merge replication, the role that the log reader agent fills in transactional replication is taken by a set of triggers, tables, and views. These objects are added to each subscription database as well as to the publication database. The merge agent applies the collected changes to the subscriber. Because merge replication is bi-directional, the merge agent also applies changes to the publisher.

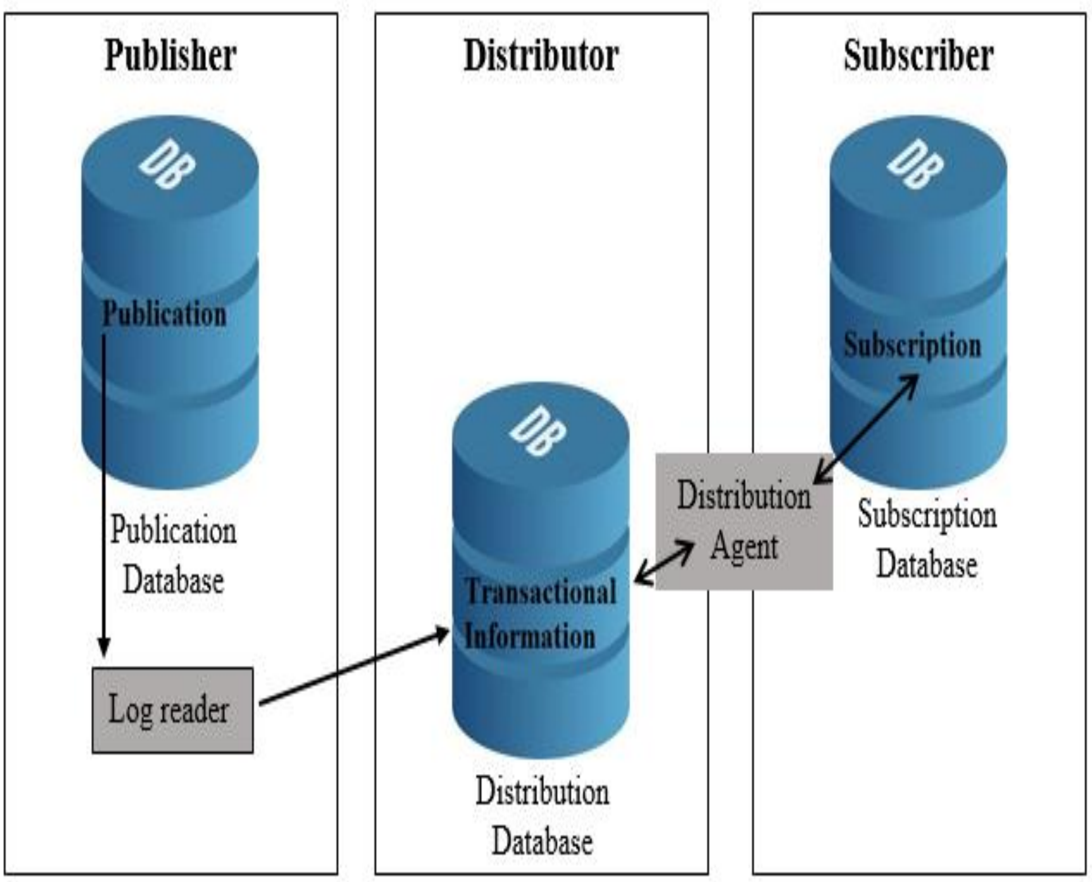


Figure 6.2 Merge replication agents and triggers

Conflicts

When the same data are modified at two different locations then i.e. if one of the global unique identifiers occurs in the list initiating at the subscriber and it also occurs in the list that originates at the publisher that a *conflict* has occurred [137]. There are several reasons for conflicts. A conflict occurs due to the fact that a row is changed on one node (publisher or subscriber) and then changed on another node. This type of conflict is called an *update-update* conflict. However, if a row is updated on one node, but deleted on another node, the resulting conflict is called an *update-delete* conflict. A conflict can also occur if a change that is applied to one node cannot be applied to another node called a *failed-change* conflict.

Handling Conflicts in Merge Replication

To handle any conflicts that have occurred during the replication synchronization, there are certain measures. A conflict occurs at a publisher and subscriber or it can occur at two different subscribers. Conflicts can be automatically resolved using the conflict resolver. However, the special tool like (i.e., conflict viewer) allows you to choose a different resolution for the conflict. In conflict viewer, the publisher is connected and the databases folder, user database and the Publications folder are expanded followed by right click of publication to view the conflicts occurred. The three main conflicts can occur in merge replications namely *Update conflict*, *Insert conflict* and *Delete conflict*.

The primary measure for handling conflicts is by setting the priority of the publication. This will defeat the primary key collisions and most other conflicts by replacing the subscribers' row with the publishers' row. Date Time resolver can also be employed in which the option of creating stored procedure and custom conflict resolvers are utilized. When conflicts are resolved during synchronization, the data from the losing row is written to a conflict table. It is necessary to review conflicts periodically to help reduce the size of the conflict table.

6.6 MGSO BASED DYNAMIC REPLICATION STRATEGY

6.6.1 CAP Theorem and its limitations

CAP theorem has been displayed in the context of a web service, presented as trade-off between consistency, availability, and partition tolerance [138]. It can be expressed as: "In a network subject to communication failures, it is impossible for any web service to implement an atomic read/write shared memory that ensures a response to each request." The three fundamental properties of CAP theorem are **C**onsistency, **A**vailability and **P**artition-tolerance [139]. CAP offers architecture improvement of distribution systems. However one can't build up a distributed database system that is continually available, sequentially-consistent and tolerant to partition pattern. It must be assembled only with two of these three properties. Furthermore, certain restrictions degrade the efficiency of CAP theorem [140], [141]. The limitations of CAP are given in [141], [142].

6.6.2 MGSO based replication model design

The limitations of the CAP theorem have been considered and this presented model of MGSO based dynamic replication algorithm is presented in order to achieve better performance. Initially, the distributed database system to be considered is described. The scheme consists of n nodes, represented as $p_1, p_2, \dots, p_n$. Each node comprises of a processor and a local memory. All the local memories are remote and available simply by local processors. Internode communication is supported by sending data through the underlying network. All requests, each with its corresponding time deadline, are presumed to reach the processors. Requests reach at a processor synchronously and there is a concurrency control mechanism to serialize them for handling. For each requested data, it is anticipated that there are at least $t(1 \leq t \leq n)$ replicas in the database system, where n is the number of network nodes. This limitation is customarily referred to as t-availability constraint.

Figure 5.3 shows the overall flow of the presented dynamic replication strategy. The overall process begins with the initialization of the LAN clusters which consists of 1 server structure with n number of clients. The data to be replicated is selected and it is loaded to the main server in the test environment. Then the dynamic window-based mechanism is utilized to access the data on each processor, where request of the same data are stocked on a temporal window until the fixed time deadline of the user request. This window mechanism determines the concurrent execution of the arriving user requests or queries. Based on this temporal window, the processor receiving more number of requests is considered for storing a replica of the original data from the main server in order make it available for certain set of users. The MGSO is applied to identify the optimal number of processers that can be allowed to replicate based on queries/requests while also in determining which processors are suitable for storing the replica data depending upon their load and other features. The output obtained from the MGSO model is employed for final replica storage which becomes invalid if the publisher or authority of the original data tends to modify the data with updated contents. At these stages, the above mentioned process is restarted to update the replica system.

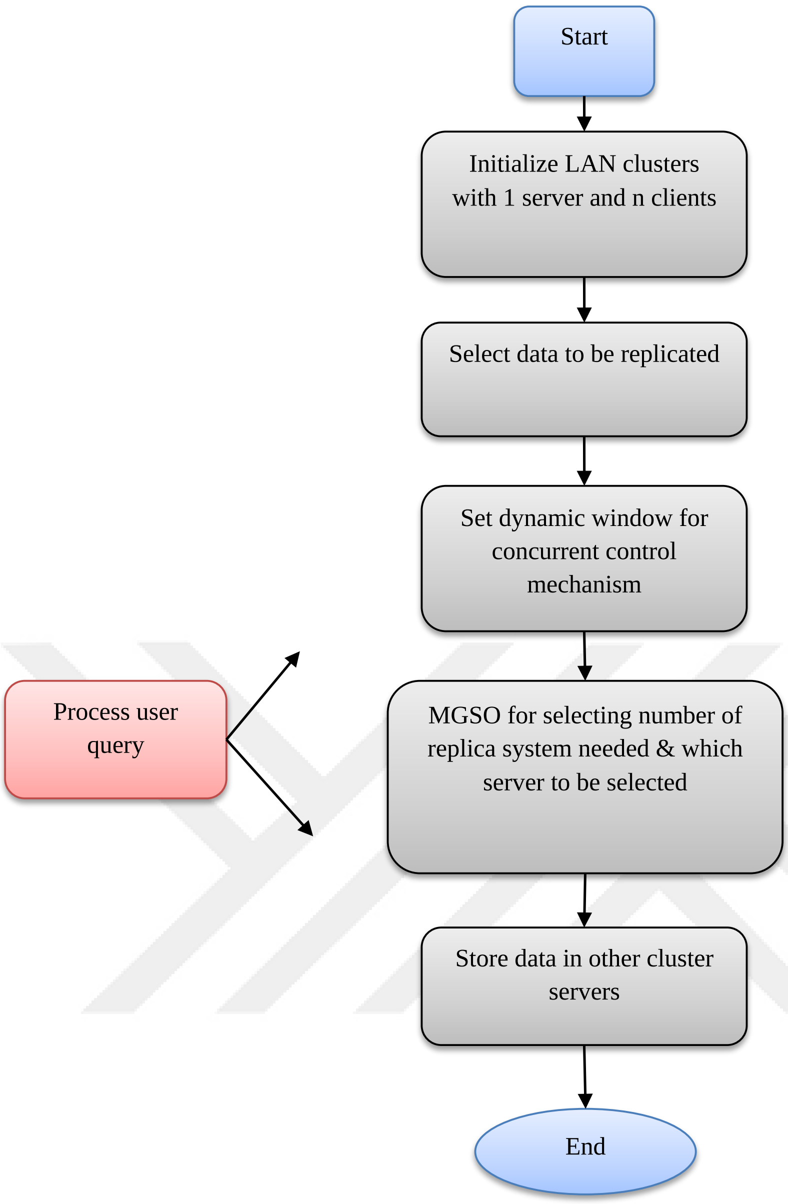

Start
Initialize LAN clusters with 1 server and n clients
Select data to be replicated
Set dynamic window for concurrent control mechanism
Process user query
MGSO for selecting number of replica system needed & which server to be selected
Store data in other cluster servers
End

Figure 6.3 MGSO based dynamic data replication strategy

In this system, all requests reach the distributed processors in the network system. For evaluation purposes, we have utilized an example of storing replica of a dataset that contains YouTube, sensor and twitter contents. This data is filtered and pre-processed to obtain complete individual set of data before storing it in the main server. Once the data is stored in the main server, the data requests from the users are analyzed. When a processor p_i wants to read a data d, if the latest version of data d is in its local memory, then data d is directly recovered from its local memory; else, since p_i knows the fixed processor set $S(d)$, p_i will direct a read request to the nearby server, p_j, in $S(d)$. This would sustain C_c units of cost. As a reply, p_j will recover data d from its local memory and send it to p_i, incurring $(C_{io} + C_d)$ units of cost. Finally, in order to minimize the total servicing cost of forthcoming requests, p_j might specify *pi* to save data d into its local memory. This read request is described as the *saving-read* request. It should be noticed that exclusive the servers of a data d can get requests from remote processors to read data *d*. Further, each processor can issue a write request for any data in a distributed database without loss of all-inclusive statement. In this manner, the request arriving at processor *q* can be anyone of these: a read request from processor *q* for a data *d* in its nearby memory, a write request from processor *q* for any data in the system, or a read request from a remote processor for a data *d'* if processor *q* is the server of data *d'.*

6.6.3 Cost Model

This cost model [143] is utilized to compute the cost of servicing a read request or a write request arriving at a processor *q*. The cost of servicing of request *REQ* by an algorithm *A* is defined as $COST_A(Req)$. Consequently, in view of servicing a read request $R_d^{p_i}(k)$, let A_d be the allocation pattern of data d identified by processor *q*, then the cost is given by

$$COST_A\left(R_d^{p_i}(k)\right) = \begin{cases} 1 & if\ p_i \in A_d \\ 1 + C_c + C_d & if\ p_i \notin A_d\ and\ R_d^{p_i}(k) is\ not\ a\ saving - read\ request \\ 2 + C_c + C_d & if\ p_i \notin A_d\ and\ R_d^{p_i}(k) is\ a\ saving - read\ request \end{cases} \quad (6.1)$$

An important feature of this model is that after getting the data d, when *pi* keeps data d into its local memory (saving-read request), then the servicing cost will be one unit greater than when p_i does not save data d (non-saving-read request). Moreover, whether this read request is a saving-read request or not is certain by processor *q* established on the dynamic request window mechanism. Once processor *q* chooses that request $R_d^{p_i}(k)$ is a saving-read request, *q* and p_i will transform their corresponding allocation patterns.

Likewise, consider servicing a write request $W_d^q(k)$, and let A_d be the allocation pattern of data d recognized by processor q. Then, the cost of servicing this request is specified as follows,

$$COST_A\left(W_d^q(k)\right) =$$

$$\begin{cases} (t-1)C_d + t + \sum_{p \in S(d)} |inv\_list(p,d)| C_c & if\ q \in S(d) \\ tC_d + (t+1) + \sum_{p \in S(d)} |inv\_list(p,d) - \{q\}| C_c & otherwise \end{cases} \tag{6.2}$$

Here $inv\_list(p,d) - \{q\}$ represents the set of processors in $inv\_list(p,d)$, excluding processor q. A write request generates a different form of a data. In order to sustain the consistency among the replicas of the data, the new form should be moved to those processors which have the available replicas of this data in their corresponding local memories. It must be noted that each relocation of this data to the corresponding processors will sustain C_d units of cost.

6.6.4 Window mechanism

A certain concurrency control mechanism in each processor expected to serialize the arriving requests so that it yields at most one request in a δ time units, with δ being imperceptibly small. Without loss of consensus, it is expected that $\delta = 1$. Each request landing at a processor creates an underlying request scheme. To service these requests, their individual time deadline, at whatever points a request is discharged from the concurrency control mechanism; window mechanism is invoked to respond to this request. Figure 6.4 shows the presented window mechanism working process which defines the addition of new write requests to the currently available read requests. It can be seen that the concurrency control is performed in time slots for each requests.

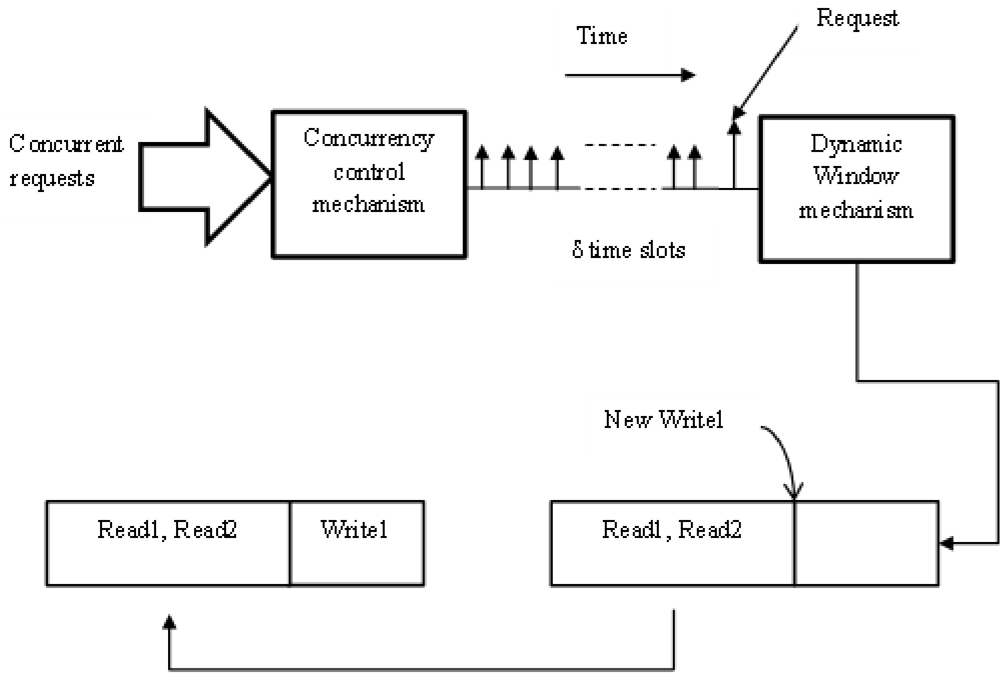


Figure 6.4 Working procedure of dynamic window mechanism

Further, the scheme of window mechanism comprises a dynamic practice. Numerous dynamic request windows are produced in each processor, one for each requested object. A request window for a data d in a processor is symbolized as $win(d)$. Each request window is of a FIFO type window with size τ to collect most τ number of requests in τ time units for the similar data. Further, two counters, TC_d^1 and TC_d^2, are connected for each $win(d)$. TC_d^1 has an preliminary value τ and the value of TC_d^1 is decremented by one per time unit till it touches 0. TC_d^2 will track the time deadline of requests in the $win(d)$. The window mechanism is described as [144]:

Algorithm 6.1: Window mechanism

```
Initialize the servers and sub-servers
For each time unit
    If there is request REQ for a data d with time deadline $\tau_{REQ}$
        If $win(d)$ does not exist
            Generate $win(d)$ and insert REQ into $win(d)$;
$TC_d^1 = \tau$; $TC_d^2 = \tau_{REQ}$;
        Else If REQ is a read request
            Insert REQ into $win(d)$; $TC_d^2 = \min(TC_d^2, \tau_{REQ})$;
        Else
            Service the requests in $win(d)$;
             Insert REQ into $win(d)$; $TC_d^1 = \tau$; $TC_d^2 = \tau_{REQ}$;
        End if
End if
End for
For each presently existing request window $win(d')$ in processor q
    $TC_{d'}^1 = TC_{d'}^1 - 1$; $TC_{d'}^2 = TC_{d'}^2 - 1$;
    If $(TC_{d'}^1 == 0)$
        Service the requests in $win(d')$; Delete $win(d')$;
        If (q is a server of data $d'$)
            Invalid the copies of $d'$ in processors that fit to
$inv\_list(q, d')$;
            Empty $inv\_list(q, d')$;
        $A_{d'} = S(d')$
        End if
Else if $(TC_{d'}^2 == 0)$
        Service the requests in $win(d')$; $TC_{d'}^1 = \tau$; $TC_{d'}^2 = \infty$;
    End if
End for
```

It can be witnessed that the handling of requests in a request window can be determined by the deadline obligation of a specific request and is not utterly constrained by the realization of the request window. In fact, three conditions can activate instant servicing of the requests in a $win(d)$. At any time TC_d^1 of a *win(o)* in processor q ranges 0, the $win(d)$ will be removed from q and window mechanism will reset A_d in q to $S(d)$. Likewise, if q is a server of data d, then q will direct invalidate control-messages to the processors in $inv\_list(q, d)$ to invalidate the replicas of data d and void $inv\_list(q, d)$. The succeeding request for data d received at processor q will be deliberated as the first request for data d and $win(d)$ will be restarted by the window mechanism. Thus, the request structure introduced into $win(d)$ is considered through its distinct lifetime as σ_d it is obvious that be using the window mechanism, a σ_d will be principally subdivided into numerous phases $P(1)$, $P(2)$, . . . , $P(r)$. Each phase fits either into Type I or into Type II. While the Type I phase comprises of a number of read requests, a Type II phase comprises of a write request monitored by numerous read requests. Whenever a phase originates into a reality, the window mechanism will prove the request sequence in that phase without any understanding of the upcoming phases [144].

It must be noticed that the extent of a request window τ is a key parameter in our system. Different estimations of τ bring about various execution of the system. The estimation of τ ought to be resolved based on the node ability, (for example, CPU power, memory limit, and network data transmission), system request arriving rate, and deadlines forced by the application requests. It is clear that, the estimation of τ ought not be too small. If τ = 1, at that point each request will frame a request sequence, and each request window would be produced and erased in one time unit. This will suddenly consume the vast majority of the computing capacity of the processor. It must also be noticed that if the deadlines imposed by the requests are too short for the system to process, certain requests might be dropped by the system. Without loss of generality, the deadline imposed by a request that can be effectively handled by the system is at least equivalent to 1. Such dropped requests known as the *blocked* requests would either leave the system or are resubmitted, depending on the underlying application. It can be managed by the admission control mechanism of this model.

6.6.5 Replica selection and placement problem

The selection of replica and the problem of placement in the distributed systems can be defined by considering a distributed system containing K data objects that would be replicated onto N servers [145]. Let $S(n)$ and $O(k)$ be the names of server n and data object k, correspondingly. $C(n)$ and $V(k)$ denote the capacity of server n and the volume of data object k, respectively, where $1 \leq n \leq N$ and $1 \leq k \leq K$. A link amongst two servers $S(n)$ and $S(m)$ (if it occurs) has an integer number $l(n, m)$, and this provides the communication cost for transmitting a data unit amongst servers $S(n)$ and $S(m)$. It is presumed that $l(n, m) = l(m, n)$. Let $read(n, k)$ and $write(n, k)$ be the number of read and write demands requested from server n for data object k. Each data object $O(k)$ has a principal server $P(k)$, which holds the main copy of $O(k)$. The main copy of data objects cannot be de-allocated. Figure 5.5 shows the replica placement strategy in which the replica has been selected after determining the constraints and then the placement is carried out for practical use.

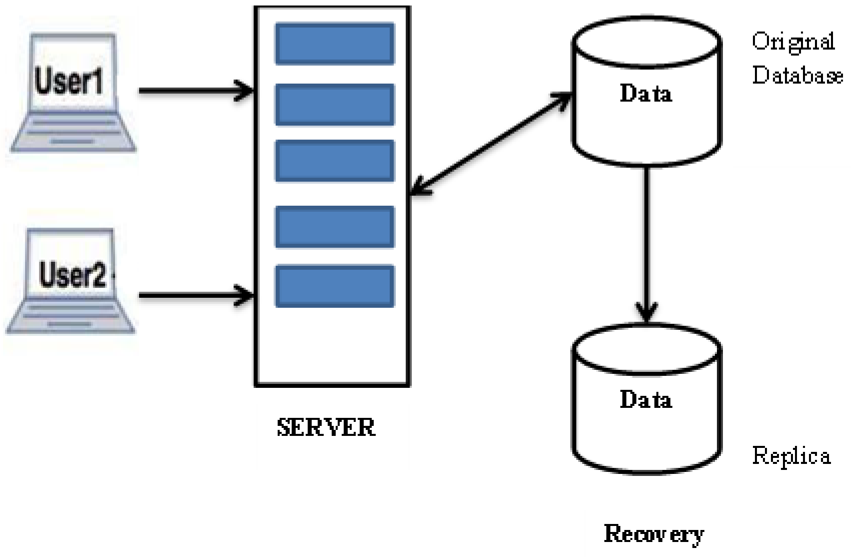


Figure 6.5 Replica placement strategy

It is expected that each primary server has the replication representation of k-th data object, $RS(k)$ which comprises a list of servers where $RS(k)$ is replicated. In order to achieve a write request, the $P(k)$ obtains the update request from source server which requests to update the data object $O(k)$ and broadcasts it to all servers in its replication representation $RS(k)$. The main objective of replica placement problem is allocating replicas over all the servers in order to reduce total operation cost $TOC(k)$, which is induced by two modules, $access_R(k)$ and $access_w(k)$. Hence $TOC(k)$ is obtained as follows:

$$TOC(k) = access_R(k) + access_w(k) \qquad (6.3)$$

Here $access_R(k)$ is the total operation cost due to all servers reading requests for $O(k)$ and $access_w(k)$ the total operation cost due to all servers writing requests for $O(k)$.

$access_R(k)$ and $access_w(k)$ are given as follows:

$$access_R(k) = V(k) \times (\sum_{n=1}^{N} read(n,k) \times l(n,..,NS(n,k))) \qquad (6.4)$$

$$access_w(k) = V(k) \times \sum_{n=1}^{N} \left(write(n,k) \times \left[l\big(n,P(k)\big) + \sum_{(\forall j \in RS(k)),j\neq n} l(P(k),j)\right]\right) \qquad (6.5)$$

Here $NS(n,k)$ is the nearby server to $S(n)$ that encloses a replica of $O(k)$. Consequently, the concise TOC based on total read and write requests' cost for all data objects is obtained as follows:

$$TOC = \sum_{k=1}^{K} TOC(k) \qquad (6.6)$$

Reducing this equation is the solution to the replica selection and placement problem in distributed systems. This is achieved in this presented model by using a multi-objective glow-worm swarm optimization algorithm for optimally selecting the number of replica and also with the server location for placement of replica.

6.6.6 MGSO based replica selection and placement

In this presented model, each glow-worm agent (gwa) is a $N \times K$ matrix with Boolean components [146]. The replication problem is modelled into each glow-worm and it moves towards the brighter glow-worm i.e. determining which processor is better suited to store the replica data. As the utilized glow worm and agents are of $N \times K$ Boolean matrix, the logical operations are employed. The logical OR operations are performed by the in-built processing unit to move the selected glow-worm towards the better one in terms of brightness (i.e. better replica conditions). Let element $gwa_{n,k} = 1$ if $S(n)$ comprises a replica of $O(k)$ and $gwa_{n,k} = 0$ otherwise. The fitness value of each agent is calculated in terms of TOC percentage, which is saved using the replication strategy of the algorithms, compared to the initial one, i.e., when only primary copies exist. This value indicates the solution quality of replication schema associated with the glow-worm agent. Fitness function is computed as follows:

$$f_{gwa(i)} = 1 - \frac{TOC_{gwa(i)}}{TOC_{initial}} \tag{6.7}$$

Here $TOC_{gwa(i)}$ is the TOC for replication scheme for the i-th agent and $TOC_{initial}$, the TOC achieved in the initial allocation, which is calculated only when the primary copy of the data objects exist while replica has not been produced.

In addition to the Boolean logical operations, the update and other processes of the MGSO are also needed to be processed. The luciferin update process is performed as follows [147]:

$$l_i(t) = (1-\rho)l_i(t-1) + \gamma J(x_i(t)) \tag{6.8}$$

Here, $l_i(t)$ represents the luciferin level associated with glow-worm *i* at time *t*, ρ is the luciferin decay constant ($0 < \rho < 1$), γ is the luciferin enhancement constant and $J(x_i(t))$ represents the value of the objective function at glow-worm *i*'s location at time *t*.

During the movement phase, each glowworm decides, using a probabilistic mechanism, to move toward a neighbor that has a luciferin value higher than its own. However, in order to increase population diversity and the convergence speed, a disturbance term is added to the location updating formula:

$$X_i(t+1) = X_i(t) + s\left(\frac{X_j(t)-X_i(t)}{\|X_j(t)-X_i(t)\|}\right) + u \tag{6.9}$$

Here $u = \alpha * rand * (iter\_max - t)/iter\_max$, s step, α which can control the range of disturbance, $rand$, the random number that meets normal distribution. α is the algorithm parameter set in advance, which is relevant to the problem. The range of α values 0 to 1, but in general, the value of α is set to 0.001.

When the glowworms are determined only by the local information to select their movements, it is anticipated that the number of peaks caught would be a function of the radial sensor range. Consequently, GSO utilizes an adaptive neighborhood range in order to detect the existence of multiple peaks in a multimodal utility landscape. It is represented as follows:

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(x_t - |X_i(t)|)\}\} \tag{6.10}$$

Here r_d^i is the neighbourhood range, r_s is the neighbourhood range with respect to steps size, β is a constant parameter and x_t is a parameter used to control the number of neighbors. The complete process of MGSO for replication process in distributed systems is given as follows:

Algorithm 6.2: MGSO based replica placement

```
Initialize the population N, step size s
Set servers (solutions) as $gwa$
A=0; All dominated solution will be placed in A
Let t=0;
Compute $f_{gwa(i)}$;
Update $l_i(t)$;
Determine direction of movement of each $gwa$;
Update $gwa$ location;
Non-inferior sorting process initiated;
Update A;
t=t+1;
Until ($t = iter$_max)
Place replica at S(n) in A;
```

```
End
```

The most time-consuming parts of the algorithms are associated with population initialization and updating best positions is achieved by all glowworms in each iteration. The time complexity of initializing $N\, glowworms$ is $O(N_p(K^2N + KN^2))$. In order to update best neighbourhood locations in each iteration, fitness value of all gwa must be considered. The time complexity of fitness value calculation $O(K^2N)$. Thus, the time complexity of solving replica placement in distributed systems via MGSO is $O(N_p(K^2N + KN^2) + N_g(K^2N)))$ [148].

3.6 CHAPTER SUMMARY

This chapter has aimed and developed an efficient replication strategy for distributed databases. Initially the need for replication and other types of replication strategies are explained in this chapter. Then the MGSO based dynamic data replication strategy is developed with better efficiency. The explanation about this process is provided in this chapter with subsequent evaluation results explained in the Chapter 8

7
Conclusion

7.1 General

Distributed database management systems have become a greater research topic due to the increasing need for retrieving and modifying the large volume of data created in this big data era. Designing a DDB faces many practical difficulties and hence the DDBs are considered to be highly challenging yet worth system for processing in organizations and related domains. The three major issues of DDB design, fragmentation, allocation and replication are vital in executing the query processing tasks and ensuring best servicers to the users. Recent trend has seen the organizations and industries aim to utilize cloud Platform as a Service for their benefit. Many researches have been undertaken around the world in providing better processing speed of the data interaction between the manipulator and benefactor. As the need for improvement in the DDB design is still not effectively achieved, this research thesis contributes to provide versatile techniques.

The first contribution is the Improved Hierarchical Agglomerative Clustering based fragmentation approach that employs enhanced clustering approach for data fragmentation. The second contribution is the Chicken Swarm Optimization based data allocation technique in which the DAP is modelled as optimization problem for solution. The third contribution is the development of efficient data replication strategies using Multi-Objective Glowworm Swarm Optimization for distributed database systems. This chapter presents the summary of the research work undertaken in this thesis followed by the conclusions drawn from the experimental contributions and also highlights the recommendations for the outgrowth of this study.

7.2 MAJOR FINDINGS

This section outlines the major contributions and findings of the research work undertaken in this thesis so as to highlight interpretations, generalizations and elucidations. The major outcome of this work is the solution to basic issues of DDB design process. The development of heuristic and metaheuristic algorithms-based approaches for fragmentation; allocation and replication have been the novel contributions.

The first approach of Improved Hierarchical Agglomerative Clustering based data fragmentation has been developed by investigating various clustering algorithms. Based on the investigations, it has been found that the Hierarchical Agglomerative Clustering model is highly suitable for data fragmentation but there are limitations in applying them practically. In order to provide better data fragmentation than other models, the HAC has been improved by modifying its functionality of considering the data counts for matrix formation. This has significantly impacted the data fragmentation process positively. The experimental results have been conducted to verify this claim and it has been positive that the IHAC outperformed the other models in providing efficient data fragmentation.

The second approach of Chicken Swarm Optimization based data allocation models the DAP into optimization problem in the lines of QAP. This optimization function is reduced by using CSO based on its hierarchy social ordering and food searching process. This CSO based data fragment allocation has significantly improved the DDB design in terms of efficient allocation. The experimental results have been conducted to verify this claim and it shows the CSO based allocation technique has greater performance than other compared models with highly accurate data allocation.

The third approach of Multi-Objective Glowworm Swarm Optimization based data replication utilizes the results of CSO data allocation to efficiently replicate the data to most appropriate processors. Initially, the snapshot replication and merge replication process for suitable databases are illustrated. Then the MGSO based approach is utilized to select the number of replica and best location for placing the replica. It is evident from the experimental results that the MGSO based approach has higher performance than the other compared models. From the evaluation results presented in chapter 6, it can be seen that the proposed techniques for fragmentation, allocation and replication significantly enhance the respective process and thus improves the design of DDBs with high QoS and assuring basic properties of CAP theorem.

7.3 FUTURE WORK

Although the developed techniques for fragmentation, allocation and replication have performed significantly greater than the existing approaches, there is room for greater improvements. Several lines of research may arise from this work based on the smaller limitations or unexplored concepts.

Firstly, the IHAC utilized for data fragmentation performs better than other models but the efficiency of HAC can be enhanced in many other ways in future. Also there are many new clustering algorithms especially from the optimization based clustering algorithms and hybrid models that have the ability to outperform the HAC given that it adapts easily with the DDBs design. Combining fuzzy algorithms, partitional algorithms and optimization algorithms like genetic algorithms, particle swarm optimization, etc. can provide new insights in this field of research.

Secondly, a similar insight can be pretended for CSO and MGSO algorithms. There are many variants of CSO and MGSO that can be tested for the problems considered in this research. Also hybrid algorithms can also be employed for the same purpose. A major area of concern in the optimization algorithms is the low convergence speed when the problem becomes global. This can be improved by developing approaches that avoid the processes to lock in local optima while ensuring better global optima solution. This can also be an idea for future researches.

Thirdly, the introduction of classification algorithms can be examined for the specified problems. Many researchers have developed combined fragmentation and allocation models; however, there are limitations in performing the two processes using the same technique. This issue can be considered for research since the concept of utilizing a single algorithm for multiple processes is highly efficient and less time consuming. Developing multi-objective and multi-processing models can be a viable option for future researches.

8
Results and Discussion

8.1 Experimental Setup

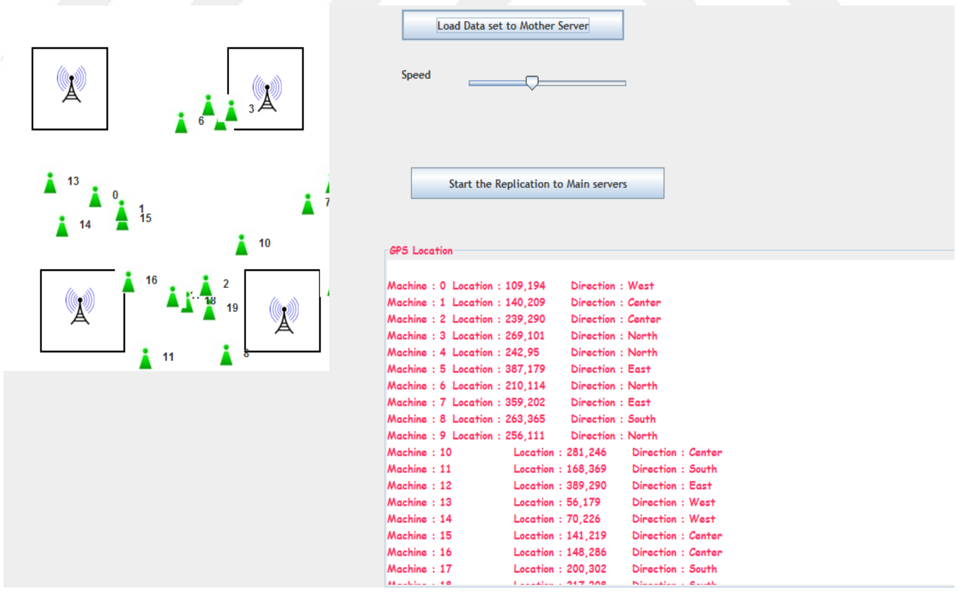
The experimental setup of the proposed research work is a mixture of both the hardware and software configurations. The experiments are conducted using the Hadoop cluster consisting of 16 dedicated machines (1 master and 15 slaves). This structure is further assigned into 3 sub-servers and 12 clients among the slave nodes. The master node (mother server) acts as both the name node as well as data node. All the clients are identical with Intel Core i5 (5th generation) processors, 8 GB of RAM and 1 TB hard drive. The master node is initiated by a specification of Intel Core i7 (6th generation) processors, 16 GB of RAM and 1 TB hard drive. All tests were run on Hadoop version 2.7.3 with the java version 1.8.0 131 installed. Default Hadoop and YARN Resource manager configurations were used. MySQL and Net beans are also installed for log files and visualizing the data replication between the mother server and sub-servers initiated among the slaves.

The files to be replicated are copied into the HDFS from the local system. These files are extracted from three major sources, Twitter, Facebook and YouTube containing various types of files namely text, audio and video files with varying sizes from 500KB to 500MB. The HDFS divided these files into blocks of either 128KB or 128MB based on file size and the access paths are through random nodes at different time intervals. The default storage of HDFS is utilized for the data fragmentation, allocation and replication. The performance of the proposed algorithms for fragmentation, allocation and replication are evaluated to ensure its efficiency in this specified experimental scenario.

8.2 IMPLEMENTATION AND EVALUATION OF PRESENTED RESEARCH MODELS

Implementation of the presented models of algorithms for data fragmentation, allocation and replication are performed in Hadoop environment with efficient query processing system. The query files were randomly generated. The trial query files encompass the queries run on a site and the attribute names equivalent to a precise dataset. A total workload of 275 queries that includes a set of 150 simple elements selection of nine attributes in the utilized dataset of Facebook, Twitter and YouTube is established.

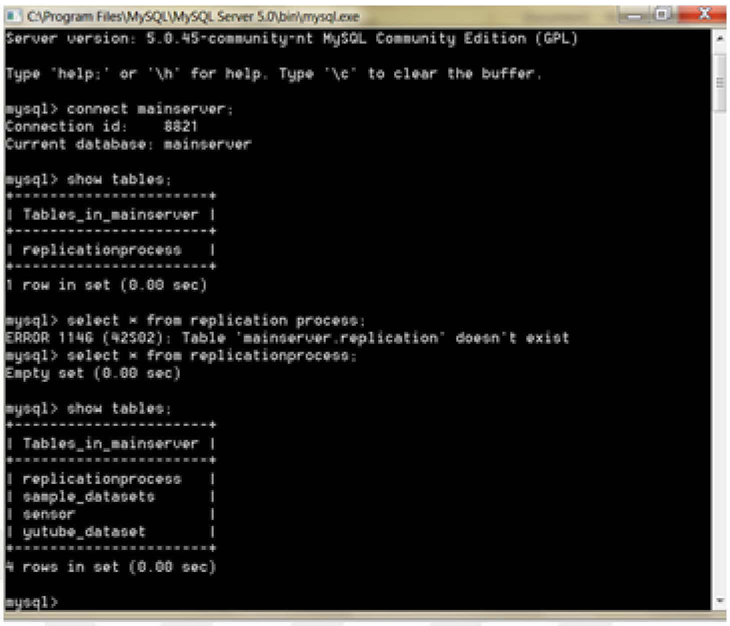
Initially the data are extracted and the queries are generated. These queries are analyzed and based on these queries the processing of the data is ensured especially the data replication process. The data fragmentation technique based on the improved hierarchical agglomerative clustering algorithm is evaluated based on its efficiency in fragmenting the dataset. Similarly, the data allocation and replication strategies are also evaluated. The performance of the developed techniques are evaluated and compared with that of the existing and state-of-the-art techniques to determine their superiority.


Load Data set to Mother Server
Speed
Start the Replication to Main servers
GPS Location

```
Machine : 0  Location : 109,194      Direction : West
Machine : 1  Location : 140,209      Direction : Center
Machine : 2  Location : 239,290      Direction : Center
Machine : 3  Location : 269,101      Direction : North
Machine : 4  Location : 242,95       Direction : North
Machine : 5  Location : 387,179      Direction : East
Machine : 6  Location : 210,114      Direction : North
Machine : 7  Location : 359,202      Direction : East
Machine : 8  Location : 263,365      Direction : South
Machine : 9  Location : 256,111      Direction : North
Machine : 10            Location : 281,246      Direction : Center
Machine : 11            Location : 168,369      Direction : South
Machine : 12            Location : 389,290      Direction : East
Machine : 13            Location : 56,179       Direction : West
Machine : 14            Location : 70,226       Direction : West
Machine : 15            Location : 141,219      Direction : Center
Machine : 16            Location : 148,286      Direction : Center
Machine : 17            Location : 200,302      Direction : South
```

Figure 8.1 Initial server set up for primary data storage

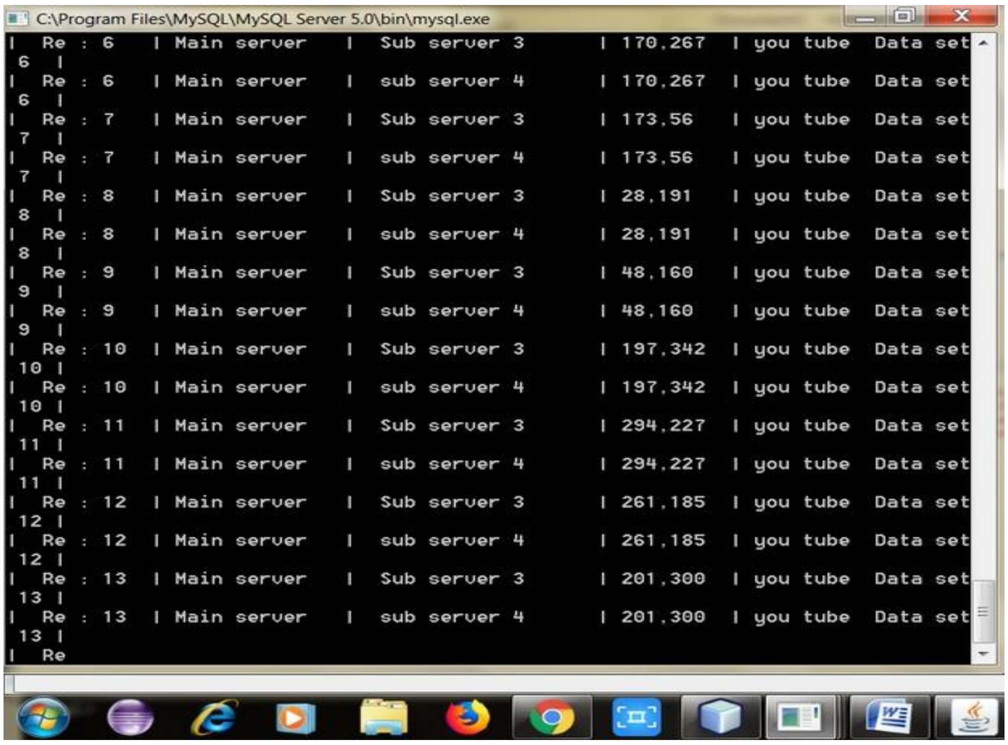
Figure 8.1 show the preliminary procedures carried out to setup primary data copy in DDB for replication. The dataset contain all the three types of gathered data. This process leads to the next possible storage process initialization using MySQL.



```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> connect mainserver;
Connection id:    8821
Current database: mainserver

mysql> show tables;
+----------------------+
| Tables_in_mainserver |
+----------------------+
| replicationprocess   |
+----------------------+
1 row in set (0.00 sec)

mysql> select * from replication process;
ERROR 1146 (42S02): Table 'mainserver.replication' doesn't exist
mysql> select * from replicationprocess;
Empty set (0.00 sec)

mysql> show tables;
+----------------------+
| Tables_in_mainserver |
+----------------------+
| replicationprocess   |
| sample_datasets      |
| sensor               |
| yutube_dataset       |
+----------------------+
4 rows in set (0.00 sec)

mysql>
```

Figure 8.2 Primary data storage process using MySQL

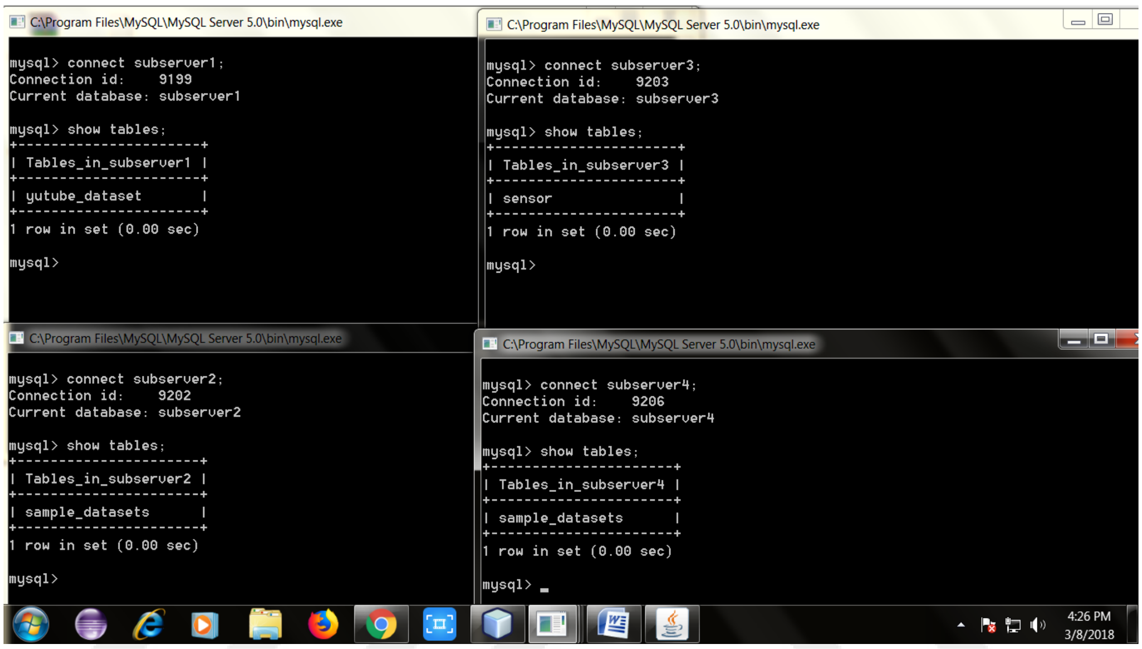
Figure 8.2 shows the initialization of the mother server using the MySQL to begin the storage of primary data copy. Completion of this process means that the data has been saved in the mother server or main server whichever described.



```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe
|  Re : 6   | Main server   | Sub server 3     | 170,267  | you tube  Data set
 6  |
|  Re : 6   | Main server   | sub server 4     | 170,267  | you tube  Data set
 6  |
|  Re : 7   | Main server   | Sub server 3     | 173,56   | you tube  Data set
 7  |
|  Re : 7   | Main server   | sub server 4     | 173,56   | you tube  Data set
 7  |
|  Re : 8   | Main server   | Sub server 3     | 28,191   | you tube  Data set
 8  |
|  Re : 8   | Main server   | sub server 4     | 28,191   | you tube  Data set
 8  |
|  Re : 9   | Main server   | Sub server 3     | 48,160   | you tube  Data set
 9  |
|  Re : 9   | Main server   | sub server 4     | 48,160   | you tube  Data set
 9  |
|  Re : 10  | Main server   | Sub server 3     | 197,342  | you tube  Data set
 10 |
|  Re : 10  | Main server   | sub server 4     | 197,342  | you tube  Data set
 10 |
|  Re : 11  | Main server   | Sub server 3     | 294,227  | you tube  Data set
 11 |
|  Re : 11  | Main server   | sub server 4     | 294,227  | you tube  Data set
 11 |
|  Re : 12  | Main server   | Sub server 3     | 261,185  | you tube  Data set
 12 |
|  Re : 12  | Main server   | sub server 4     | 261,185  | you tube  Data set
 12 |
|  Re : 13  | Main server   | Sub server 3     | 201,300  | you tube  Data set
 13 |
|  Re : 13  | Main server   | sub server 4     | 201,300  | you tube  Data set
 13 |
|  Re
```

Figure 8.3 Data fragmentation using IHAC initial stage

Figure 8.3 shows the initial stage processing of the data in the mother server and beginning of fragmentation using IHAC model. The data are fragmented into specified numbers based on the most common and hierarchical similarity between them without violating the data connectivity. Once fragmented, the final output of this data are represented in tables as shown in Figure 8.4.



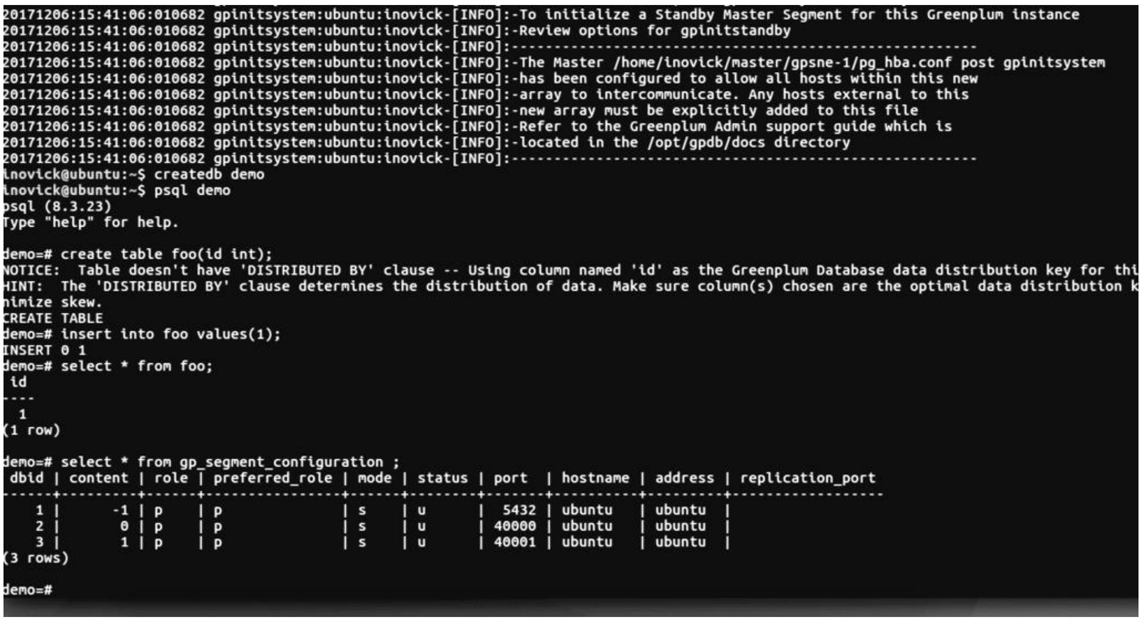
```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe

mysql> connect subserver1;
Connection id:    9199
Current database: subserver1

mysql> show tables;
+----------------------+
| Tables_in_subserver1 |
+----------------------+
| yutube_dataset       |
+----------------------+
1 row in set (0.00 sec)

mysql>
```

```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe

mysql> connect subserver3;
Connection id:    9203
Current database: subserver3

mysql> show tables;
+----------------------+
| Tables_in_subserver3 |
+----------------------+
| sensor               |
+----------------------+
1 row in set (0.00 sec)

mysql>
```

```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe

mysql> connect subserver2;
Connection id:    9202
Current database: subserver2

mysql> show tables;
+----------------------+
| Tables_in_subserver2 |
+----------------------+
| sample_datasets      |
+----------------------+
1 row in set (0.00 sec)

mysql>
```

```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe

mysql> connect subserver4;
Connection id:    9206
Current database: subserver4

mysql> show tables;
+----------------------+
| Tables_in_subserver4 |
+----------------------+
| sample_datasets      |
+----------------------+
1 row in set (0.00 sec)

mysql> _
```

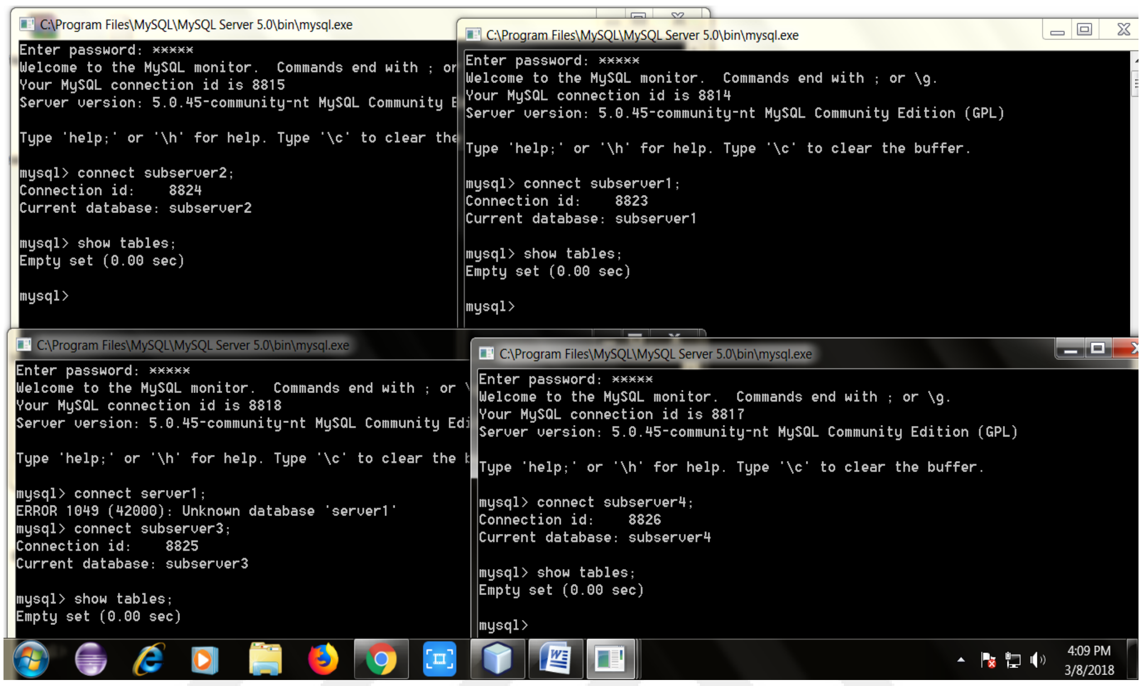
Figure 8.4 After fragmentation using IHAC



```
20171206:15:41:06:010682 gpinitsystem:ubuntu:inovick-[INFO]:-To initialize a Standby Master Segment for this Greenplum instance
20171206:15:41:06:010682 gpinitsystem:ubuntu:inovick-[INFO]:-Review options for gpinitstandby
20171206:15:41:06:010682 gpinitsystem:ubuntu:inovick-[INFO]:----------------------------------------------------------
20171206:15:41:06:010682 gpinitsystem:ubuntu:inovick-[INFO]:-The Master /home/inovick/master/gpsne-1/pg_hba.conf post gpinitsystem
20171206:15:41:06:010682 gpinitsystem:ubuntu:inovick-[INFO]:-has been configured to allow all hosts within this new
20171206:15:41:06:010682 gpinitsystem:ubuntu:inovick-[INFO]:-array to intercommunicate. Any hosts external to this
20171206:15:41:06:010682 gpinitsystem:ubuntu:inovick-[INFO]:-new array must be explicitly added to this file
20171206:15:41:06:010682 gpinitsystem:ubuntu:inovick-[INFO]:-Refer to the Greenplum Admin support guide which is
20171206:15:41:06:010682 gpinitsystem:ubuntu:inovick-[INFO]:-located in the /opt/gpdb/docs directory
20171206:15:41:06:010682 gpinitsystem:ubuntu:inovick-[INFO]:----------------------------------------------------------
inovick@ubuntu:~$ createdb demo
inovick@ubuntu:~$ psql demo
psql (8.3.23)
Type "help" for help.

demo=# create table foo(id int);
NOTICE:  Table doesn't have 'DISTRIBUTED BY' clause -- Using column named 'id' as the Greenplum Database data distribution key for thi
HINT:  The 'DISTRIBUTED BY' clause determines the distribution of data. Make sure column(s) chosen are the optimal data distribution k
nimize skew.
CREATE TABLE
demo=# insert into foo values(1);
INSERT 0 1
demo=# select * from foo;
 id
----
  1
(1 row)

demo=# select * from gp_segment_configuration ;
 dbid | content | role | preferred_role | mode | status | port  | hostname | address | replication_port
------+---------+------+----------------+------+--------+-------+----------+---------+------------------
    1 |      -1 | p    | p              | s    | u      |  5432 | ubuntu   | ubuntu  |
    2 |       0 | p    | p              | s    | u      | 40000 | ubuntu   | ubuntu  |
    3 |       1 | p    | p              | s    | u      | 40001 | ubuntu   | ubuntu  |
(3 rows)

demo=#
```

Figure 8.5 Data fragment allocation process using CSO

Figure 8.5 shows the initial data fragment allocation process initiated through MySQL. In this process, the fragmented data using IHAC are fed as input to the data allocation model where the CSO allocates them to the suitable locations for effective query processing.



```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or
Your MySQL connection id is 8815
Server version: 5.0.45-community-nt MySQL Community E

Type 'help;' or '\h' for help. Type '\c' to clear the

mysql> connect subserver2;
Connection id:    8824
Current database: subserver2

mysql> show tables;
Empty set (0.00 sec)

mysql>
```

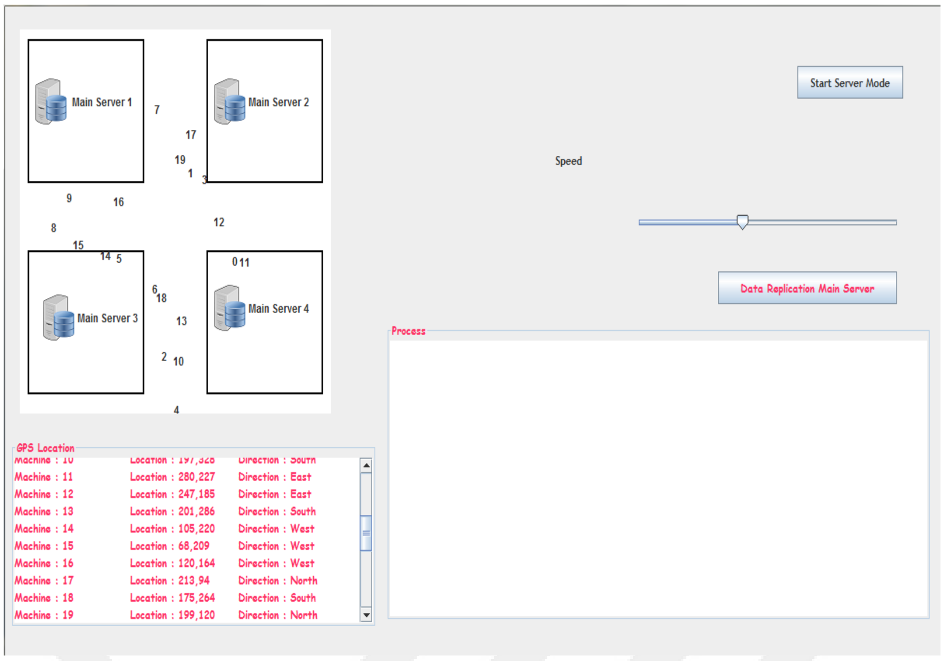
```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8814
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> connect subserver1;
Connection id:    8823
Current database: subserver1

mysql> show tables;
Empty set (0.00 sec)

mysql>
```

```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \
Your MySQL connection id is 8818
Server version: 5.0.45-community-nt MySQL Community Edi

Type 'help;' or '\h' for help. Type '\c' to clear the b

mysql> connect server1;
ERROR 1049 (42000): Unknown database 'server1'
mysql> connect subserver3;
Connection id:    8825
Current database: subserver3

mysql> show tables;
Empty set (0.00 sec)
```

```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8817
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> connect subserver4;
Connection id:    8826
Current database: subserver4

mysql> show tables;
Empty set (0.00 sec)

mysql>
```

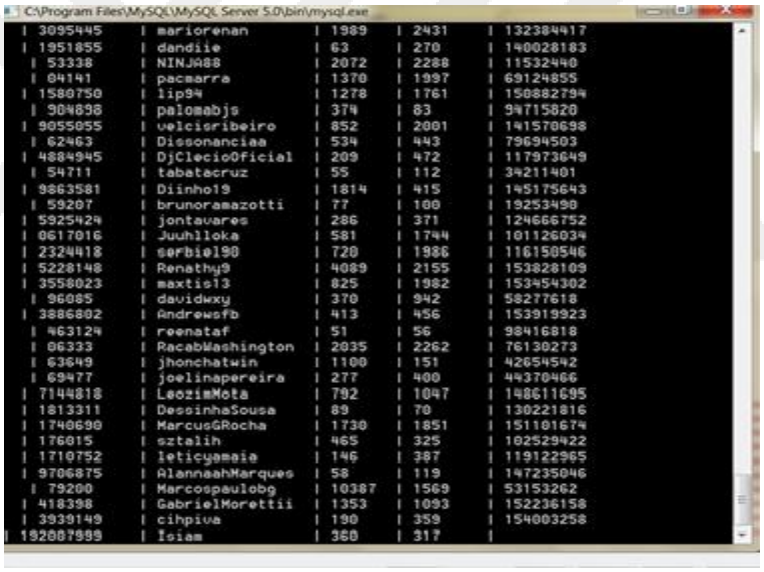
Figure 8.6 Final Data allocation result

Figure 8.6 shows the final data fragments allocated lists obtained using the CSO model. These results are ready to be replicated in the subsequent query processing stage. This replication is initiated and completed using MGSO based replication strategy.



```
Machine : 10        Location : 197,328      Direction : South
Machine : 11        Location : 280,227      Direction : East
Machine : 12        Location : 247,185      Direction : East
Machine : 13        Location : 201,286      Direction : South
Machine : 14        Location : 105,220      Direction : West
Machine : 15        Location : 68,209       Direction : West
Machine : 16        Location : 120,164      Direction : West
Machine : 17        Location : 213,94       Direction : North
Machine : 18        Location : 175,264      Direction : South
Machine : 19        Location : 199,120      Direction : North
```

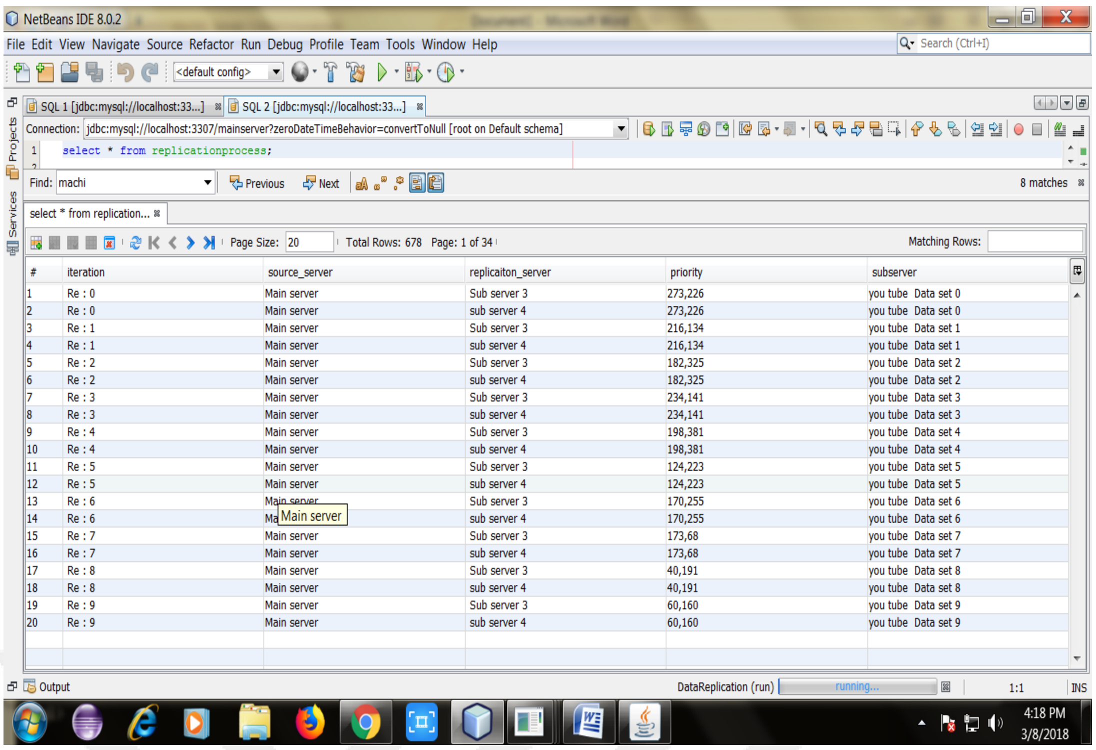
Figure 8.7 Window initialization before replication



```
| 3095445    | mariorenan        | 1989  | 2431  | 132384417
| 1951855    | dandiie           | 63    | 270   | 140028183
 | 53338     | NINJA88           | 2072  | 2288  | 11532440
 | 84141     | pacmarra          | 1370  | 1997  | 69124855
| 1580750    | lip94             | 1278  | 1761  | 150882794
 | 304838    | palomabjs         | 374   | 83    | 94715820
| 9055055    | velcisribeiro     | 852   | 2001  | 141570698
 | 62463     | Dissonanciaa      | 534   | 443   | 79694503
| 4884945    | DjClecioOficial   | 209   | 472   | 117973649
 | 54711     | tabatacruz        | 55    | 112   | 34211401
| 9863581    | Diinho19          | 1814  | 415   | 145175643
 | 59207     | brunoramazotti    | 77    | 100   | 19253490
| 5925424    | jontavares        | 286   | 371   | 124666752
| 0617016    | Juuhlloka         | 581   | 1744  | 101126034
| 2324418    | serbiel90         | 720   | 1986  | 116150546
| 5228148    | Renathy9          | 4089  | 2155  | 153828109
| 3558023    | maxtis13          | 825   | 1982  | 153454302
 | 96085     | davidwxy          | 370   | 942   | 58277618
| 3886802    | Andrewsfb         | 413   | 456   | 153919923
 | 463124    | reenataf          | 51    | 56    | 98416818
 | 86333     | RacabWashington   | 2035  | 2262  | 76130273
 | 63649     | jhonchatwin       | 1100  | 151   | 42654542
 | 69477     | joelinapereira    | 277   | 400   | 44370466
| 7144818    | LeozimMota        | 792   | 1047  | 148611695
| 1813311    | DessinhaSousa     | 89    | 70    | 130221816
| 1740690    | MarcusGRocha      | 1730  | 1851  | 151101674
| 176015     | sztalih           | 465   | 325   | 102529422
| 1710752    | leticyamaia       | 146   | 387   | 119122965
| 9706875    | AlannaahMarques   | 58    | 119   | 147235046
 | 79200     | Marcospaulobg     | 10387 | 1569  | 53153262
| 418398     | GabrielMorettii   | 1353  | 1093  | 152236158
| 3939149    | cihpiva           | 190   | 359   | 154003258
| 192087999  | İsiam             | 360   | 317   |
```

Figure 8.8 Data categorization using window mechanism

The influence of window mechanism used in the MGSO model is shown in the figures 8.7 and 8.8. Figure 8.7 demonstrates the real-world settings mandatory for setting the window mechanism while the Figure 8.8 demonstrates the query/request exploration centered on the data grouping of those requests.

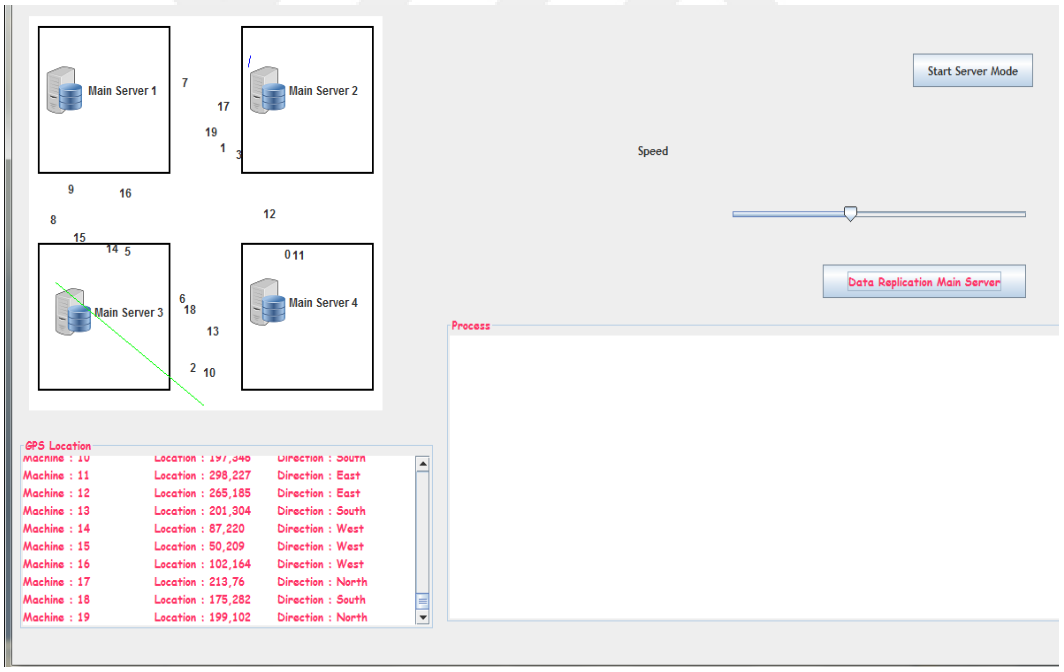

NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
SQL 1 [jdbc:mysql://localhost:33...]
SQL 2 [jdbc:mysql://localhost:33...]
Find: machi
Previous
Next
8 matches
select * from replication...
Output
DataReplication (run)
running...
4:18 PM
3/8/2018

```
Connection: jdbc:mysql://localhost:3307/mainserver?zeroDateTimeBehavior=convertToNull [root on Default schema]
select * from replicationprocess;
```

Page Size: 20 Total Rows: 678 Page: 1 of 34 Matching Rows:

#	iteration	source_server	replicaiton_server	priority	subserver
1	Re : 0	Main server	Sub server 3	273,226	you tube Data set 0
2	Re : 0	Main server	sub server 4	273,226	you tube Data set 0
3	Re : 1	Main server	Sub server 3	216,134	you tube Data set 1
4	Re : 1	Main server	sub server 4	216,134	you tube Data set 1
5	Re : 2	Main server	Sub server 3	182,325	you tube Data set 2
6	Re : 2	Main server	sub server 4	182,325	you tube Data set 2
7	Re : 3	Main server	Sub server 3	234,141	you tube Data set 3
8	Re : 3	Main server	sub server 4	234,141	you tube Data set 3
9	Re : 4	Main server	Sub server 3	198,381	you tube Data set 4
10	Re : 4	Main server	sub server 4	198,381	you tube Data set 4
11	Re : 5	Main server	Sub server 3	124,223	you tube Data set 5
12	Re : 5	Main server	sub server 4	124,223	you tube Data set 5
13	Re : 6	Main server	Sub server 3	170,255	you tube Data set 6
14	Re : 6	Main server	sub server 4	170,255	you tube Data set 6
15	Re : 7	Main server	Sub server 3	173,68	you tube Data set 7
16	Re : 7	Main server	sub server 4	173,68	you tube Data set 7
17	Re : 8	Main server	Sub server 3	40,191	you tube Data set 8
18	Re : 8	Main server	sub server 4	40,191	you tube Data set 8
19	Re : 9	Main server	Sub server 3	60,160	you tube Data set 9
20	Re : 9	Main server	sub server 4	60,160	you tube Data set 9

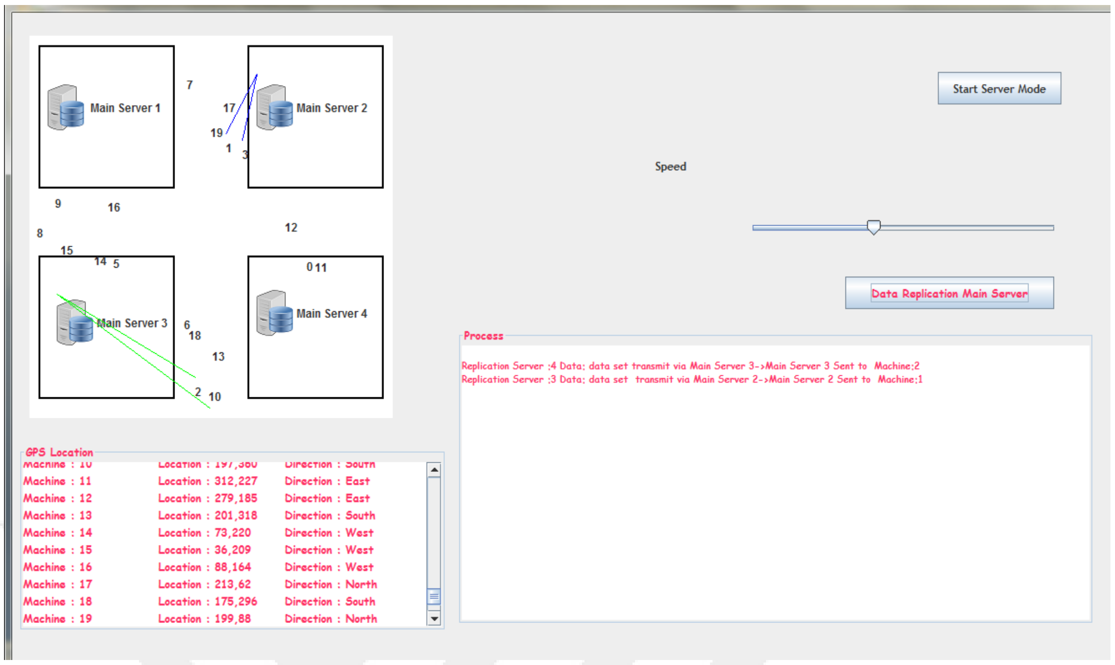
Figure 8.9 Backend server memory table



```
Machine : 10        Location : 197,346     Direction : South
Machine : 11        Location : 298,227     Direction : East
Machine : 12        Location : 265,185     Direction : East
Machine : 13        Location : 201,304     Direction : South
Machine : 14        Location : 87,220      Direction : West
Machine : 15        Location : 50,209      Direction : West
Machine : 16        Location : 102,164     Direction : West
Machine : 17        Location : 213,76      Direction : North
Machine : 18        Location : 175,282     Direction : South
Machine : 19        Location : 199,102     Direction : North
```

Figure 8.10 Replica selection and placement process

Figure 8.9 displays the backend server table after admission control process to initiate the storage capabilities of the server. These details will be used in choosing the location of replica placement. Figure 8.10 illustrates the choice of replica and assignment method as achieved centered on the theory of Figure 3 exploiting the MGSO algorithm.

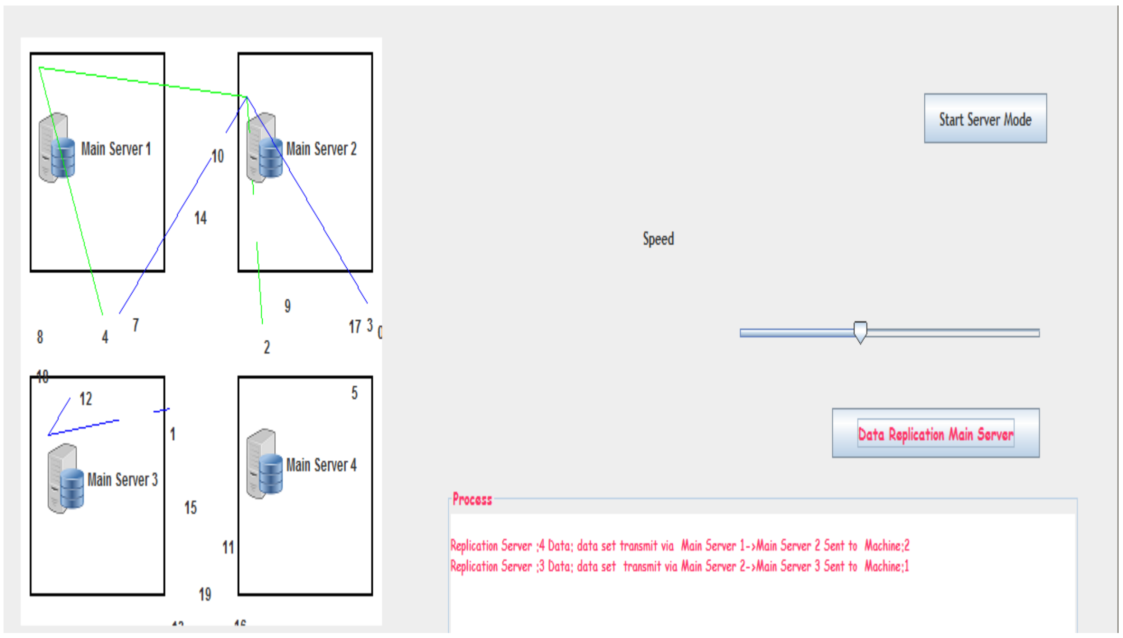


```
Machine : 10        Location : 197,360    Direction : South
Machine : 11        Location : 312,227    Direction : East
Machine : 12        Location : 279,185    Direction : East
Machine : 13        Location : 201,318    Direction : South
Machine : 14        Location : 73,220     Direction : West
Machine : 15        Location : 36,209     Direction : West
Machine : 16        Location : 88,164     Direction : West
Machine : 17        Location : 213,62     Direction : North
Machine : 18        Location : 175,296    Direction : South
Machine : 19        Location : 199,88     Direction : North
```

```
Replication Server :4 Data; data set transmit via Main Server 3->Main Server 3 Sent to  Machine;2
Replication Server :3 Data; data set  transmit via Main Server 2->Main Server 2 Sent to  Machine;1
```

Figure 8.11 Replica selection using MGSO

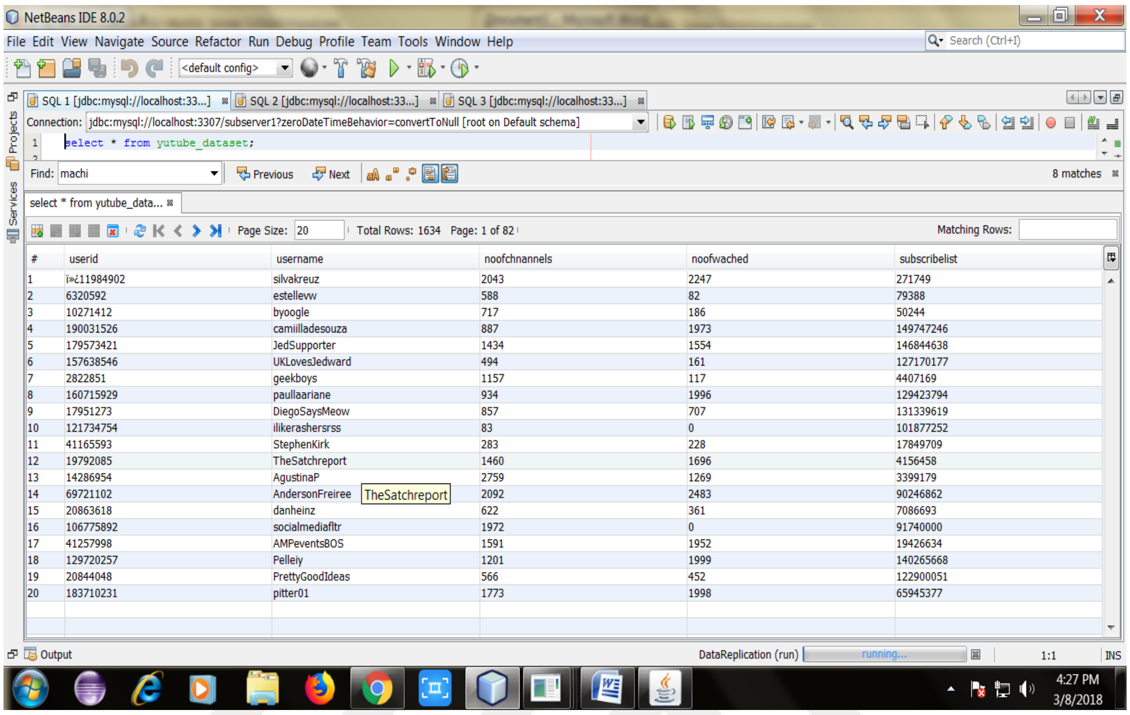
Figure 8.11 displays the choosing of replica using MGSO which is realized using the window mechanism. After the categorization of raw data into three unique classes, the most suitable sub-server is selected using MGSO.



```
Replication Server :4 Data; data set transmit via  Main Server 1->Main Server 2 Sent to  Machine;2
Replication Server :3 Data; data set  transmit via Main Server 2->Main Server 3 Sent to  Machine;1
```

Figure 8.12 MGSO based replica placement

Figure 8.12 displays the MGSO based assignment of replica. The replica is assigned to a site relevant to the cluster properties from where the data can be retrieved by the users without any difficulty whenever needed.



```
select * from yutube_dataset;
```

Find: machi — 8 matches

select * from yutube_data...

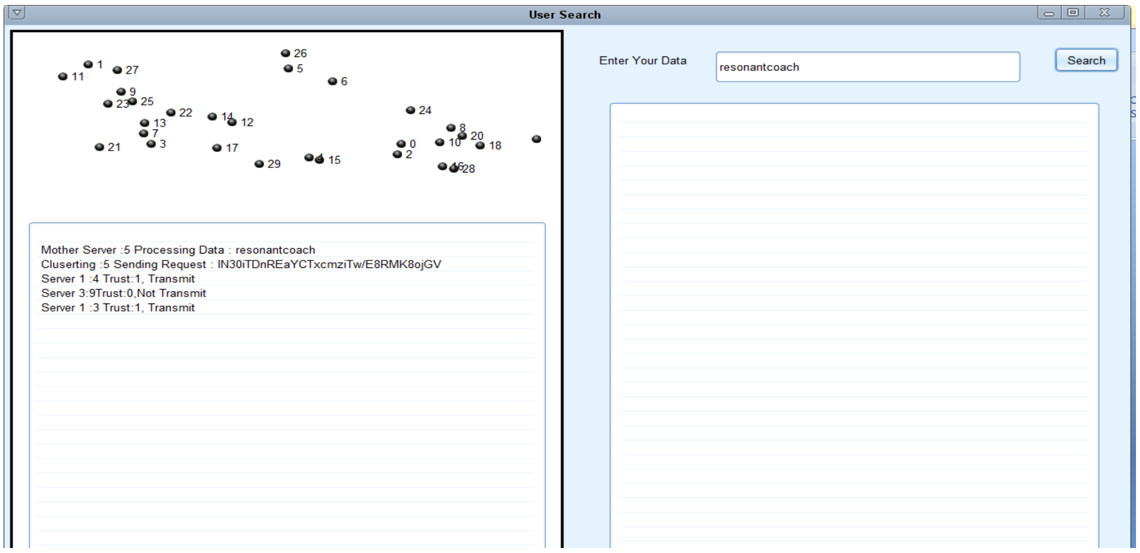
Page Size: 20 Total Rows: 1634 Page: 1 of 82 Matching Rows:

#	userid	username	noofchnannels	noofwached	subscribelist
1	ï»¿11984902	silvakreuz	2043	2247	271749
2	6320592	estellevw	588	82	79388
3	10271412	byoogle	717	186	50244
4	190031526	camiilladesouza	887	1973	149747246
5	179573421	JedSupporter	1434	1554	146844638
6	157638546	UKLovesJedward	494	161	127170177
7	2822851	geekboys	1157	117	4407169
8	160715929	paullaariane	934	1996	129423794
9	17951273	DiegoSaysMeow	857	707	131339619
10	121734754	ilikerashersrss	83	0	101877252
11	41165593	StephenKirk	283	228	17849709
12	19792085	TheSatchreport	1460	1696	4156458
13	14286954	AgustinaP	2759	1269	3399179
14	69721102	AndersonFreiree	2092	2483	90246862
15	20863618	danheinz	622	361	7086693
16	106775892	socialmediafltr	1972	0	91740000
17	41257998	AMPeventsBOS	1591	1952	19426634
18	129720257	Pelleiy	1201	1999	140265668
19	20844048	PrettyGoodIdeas	566	452	122900051
20	183710231	pitter01	1773	1998	65945377

NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
SQL 1 [jdbc:mysql://localhost:33...]
SQL 2 [jdbc:mysql://localhost:33...]
SQL 3 [jdbc:mysql://localhost:33...]
Connection: jdbc:mysql://localhost:3307/subserver1?zeroDateTimeBehavior=convertToNull [root on Default schema]
TheSatchreport
Output
DataReplication (run) running...
4:27 PM
3/8/2018

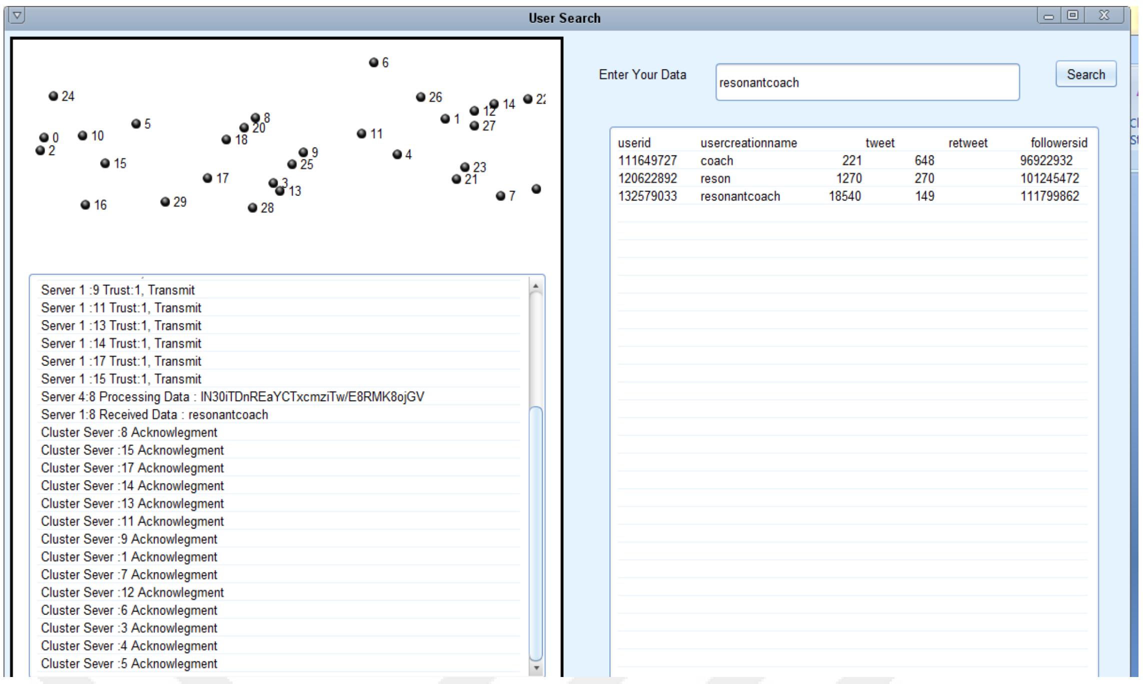
Figure 8.13 Data Replication placement table

Figure 8.13 displays the replication placement table maintained at the servers in each cluster as a routing chart for primacy choice of user query. The servers select the best location to retrieve the data using these tables for each of the user query.



```
Mother Server :5 Processing Data : resonantcoach
Cluserting :5 Sending Request : IN30iTDnREaYCTxcmziTw/E8RMK8ojGV
Server 1 :4 Trust:1, Transmit
Server 3:9Trust:0,Not Transmit
Server 1 :3 Trust:1, Transmit
```

Figure 8.14 User Query process- Input Query



```
Server 1 :9 Trust:1, Transmit
Server 1 :11 Trust:1, Transmit
Server 1 :13 Trust:1, Transmit
Server 1 :14 Trust:1, Transmit
Server 1 :17 Trust:1, Transmit
Server 1 :15 Trust:1, Transmit
Server 4:8 Processing Data : IN30iTDnREaYCTxcmziTw/E8RMK8ojGV
Server 1:8 Received Data : resonantcoach
Cluster Sever :8 Acknowlegment
Cluster Sever :15 Acknowlegment
Cluster Sever :17 Acknowlegment
Cluster Sever :14 Acknowlegment
Cluster Sever :13 Acknowlegment
Cluster Sever :11 Acknowlegment
Cluster Sever :9 Acknowlegment
Cluster Sever :1 Acknowlegment
Cluster Sever :7 Acknowlegment
Cluster Sever :12 Acknowlegment
Cluster Sever :6 Acknowlegment
Cluster Sever :3 Acknowlegment
Cluster Sever :4 Acknowlegment
Cluster Sever :5 Acknowlegment
```

userid	usercreationname	tweet	retweet	followersid
111649727	coach	221	648	96922932
120622892	reson	1270	270	101245472
132579033	resonantcoach	18540	149	111799862

Figure 8.15 User Query Process- Final Result

The query process described in the earlier paragraph is shown in figures 8.14 and 8.15. The intimations are forwarded to the user system when the queries are required. The server then analyses the availability of data and then allocates the replica based on priority with reduced time consumption. The request messages are then forwarded to obtain the data, as shown in figure 8.15.

8.3 COMPARISON OF DATA FRAGMENTATION TECHNIQUES

The performance of the presented data fragmentation technique based on the improved hierarchical agglomerative (IHAC) clustering algorithm is evaluated and compared with that of existing clustering algorithms that can be applied for fragmentation. The existing models considered for comparison are k-means algorithm [27], KR rough clustering [44] and hierarchical clustering [43]. The comparison parameters considered are final prediction error criterion (FPE), execution time, sum of squared errors (SSE) or residual sum of squares (RSS) and maximum memory usage.

Final Prediction Error Criterion: The Final Prediction Error Criterion (FPE) estimates the model-fitting error when the used model to predict new outputs. For the FPE, an optimal model is the one that minimizes the following equation:

$$FPE = SSE\left[1 + \frac{2p}{N-p}\right] \quad (8.1)$$

Here SSE is an index related to the prediction error, or the residual sum of squares, N is the number of fragments and p defines the number of parameters in the model.

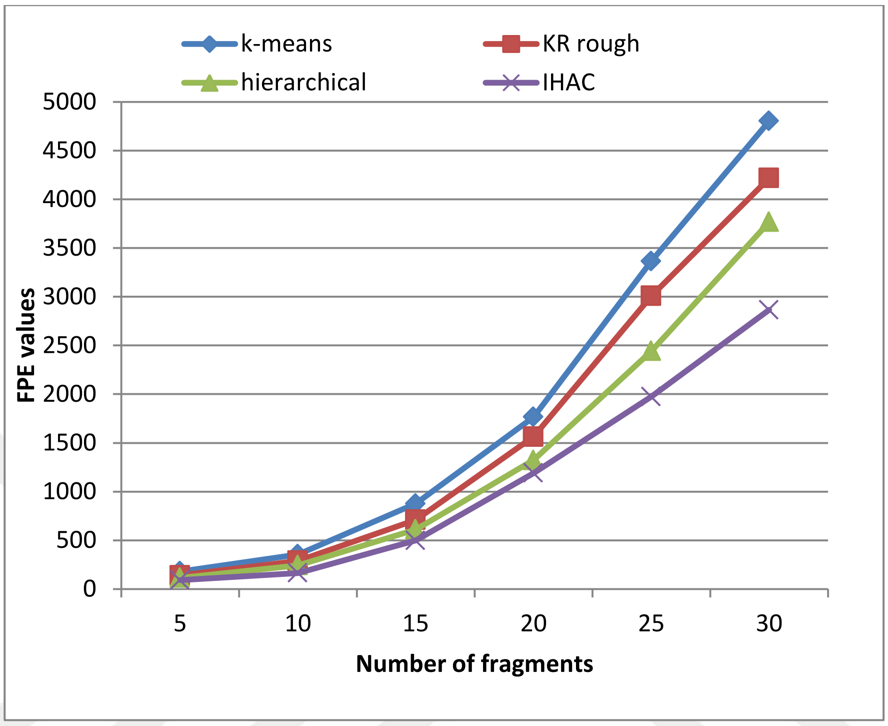


Figure 8.16 FPE criterion comparison vs. number of fragments

Figure 8.16 shows the comparison of the data fragmentation techniques in terms of their FPR values. These FPE values are plotted against the number of fragments. The FPE parameter is designated for determining the rate of error in the fragmentation model when the data are fragmented as per rules. Some set of data become meaningless when fragmented into different groups and any fragmentation model must reduce this case which can be analyzed by this FPE values. The higher the FPE values the performance is deemed not ideal. In this figure, it can be seen that the FPE values spikes with the increase in number of fragments. However the IHAC has very less values of FPE indicating it is better than the other three data fragmentation models.

Execution time: It is total time taken to complete the designated set of fragmentation using the proposed model. Execution time includes the time for loading the dataset and setting up memory for each step and subsequent supplementary processes.

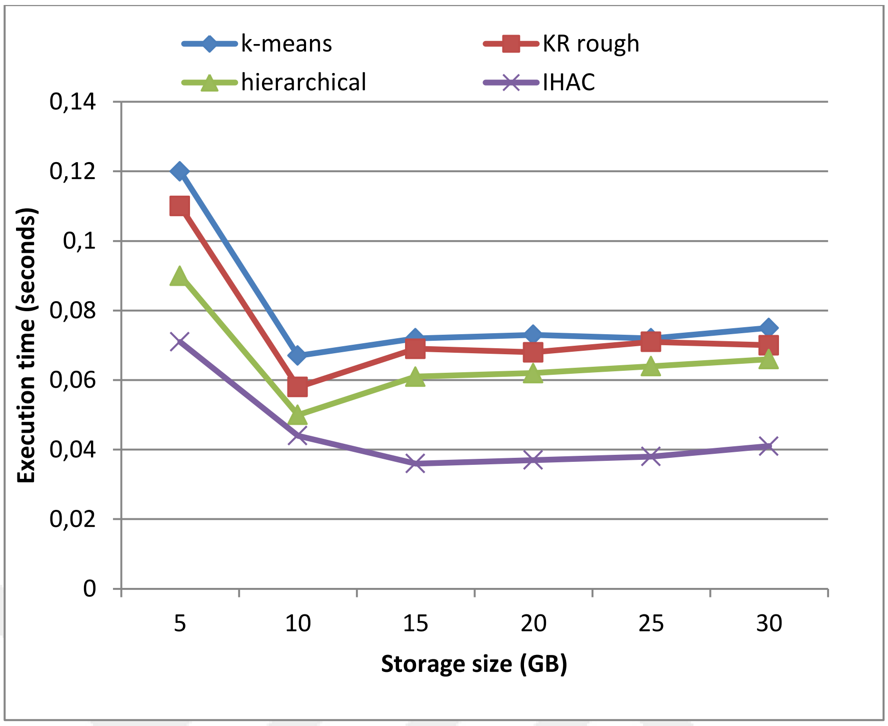


Figure 8.17 Execution time comparison vs. Storage size

Figure 8.17 shows the comparison of the data fragmentation techniques in terms of execution time of fragmentation process. It is plotted against the storage size of data measured in GB. The execution time when the storage size if 5GB is significantly higher in all the data fragmentation techniques since the initial time for processing the data are significantly higher at the first stage of fragmentation. Then after a set of initialization, the execution time reduces and finally settling under almost a linear pattern. When comparing the execution time of the considered data fragmentation techniques, the presented IHAC based model has less execution time than the other models. This is due to the fact that the presented IHAC model has significantly reduced the query processing time and hence the overall execution time drops. Thus the efficiency of IHAC based fragmentation is also proved in terms of execution time.

Sum of squared errors (SSE): SSE or RSS is the sum of the squares of errors predicted from the actual empirical values of the data. Lesser values of SSE denote the compact fit efficiency of the developed model.

$$SSE = \sum_{i=1}^{n}(\varepsilon_i)^2 = \sum_{i=1}^{n}(y_i - (\alpha + \beta x_i))^2 \qquad (8.2)$$

Here y_i is the i-th value of the variable to be predicted, x_i is the i-th value of the explanatory variable, α is the estimated value of the constant term a and β is the estimated value of the slope coefficient b.

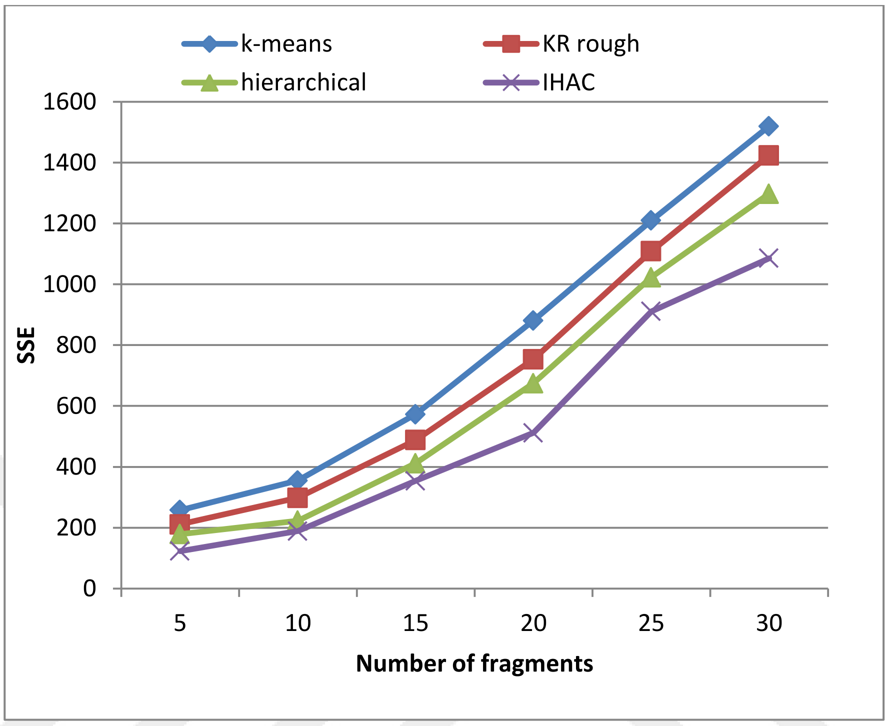


Figure 8.18 SSE comparison vs. number of fragments

Figure 8.18 shows the comparison of the data fragmentation techniques in terms of sum of squared errors. It is plotted against the number of fragments. SSE is computed based on the deviations in the data from the original data after fragmentation and the error values are squared and added to determine the overall error in data values. As the dataset used in this evaluation consists of data of three types collected from three websites, they are converted into their equitable numerical values for this evaluation. Though SSE values seems very higher, it is mainly due to the fact the dataset size utilized in this evaluation are larger data and hence the numerical values were expected to increase. However the SSE values are well within the limit compared to the size and number of fragments. The presented model of IHAC based fragmentation is highly efficient with less values of SSE and hence nominates itself as the best fit model among the compared techniques.

Maximum memory usage: This parameter is employed to determine the overall resource consumption in the system during the execution of the developed fragmentation technique. This is very essential in determining the load and capacity of the system for complete processing with the inclusion of the developed fragmentation process.

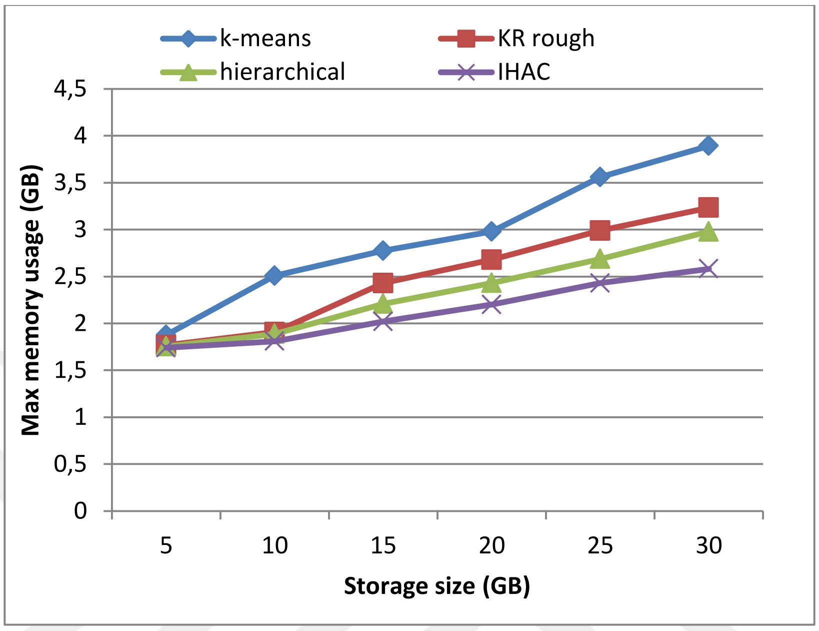


Figure 8.19 Maximum memory usage vs. Storage size

Figure 8.19 shows the comparison of the data fragmentation techniques in terms of maximum memory utilization. It is plotted against the storage size of data measured in GB. The memory utilization, in general, will be used more when any additional algorithm is implemented in the database systems. In spite of this fact, the presented data fragmentation technique consumes less memory than the other models. The major reason for this improvement is due to the fact the fragmentation process considerably reduces the memory consumption and hence the memory consumption for the IHAC algorithm is negated in this process.

8.4 COMPARISON OF DATA ALLOCATION TECHNIQUES

The presented data allocation strategy based on the chicken swarm optimization (CSO) is also evaluated in the described experimental scenario. The performance of the CSO based data allocation model is evaluated and compared with that of existing models. The existing algorithms considered for comparison are Genetic algorithm (GA) [149], Particle swarm optimization (PSO) [60] and Ant colony optimization (ACO) [47]. The metrics utilized for comparison are system cost, network transfer cost, execution time and maximum memory usage.

System Cost: System cost in any system includes the processing cost and communication cost. Cost incurred by a system can be helpful in deciding whether the system is sufficient for utilization with the presented allocation technique. Moreover the cost of a system is increased by the allocation technique but it can reduce the overall cost by effectively allocating the data fragments such that the cost in accessing data from main servers is less.

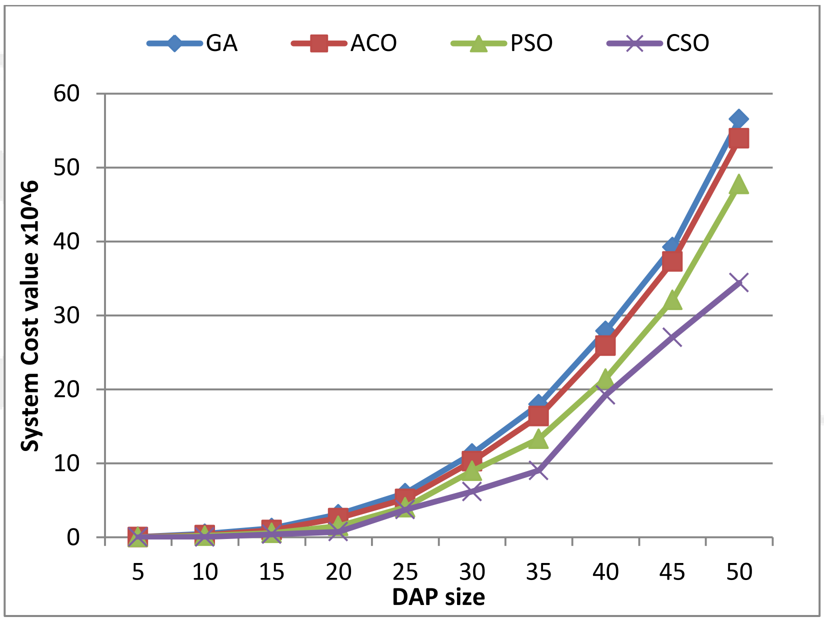


Figure 8.20 System Cost vs. DAP size

Figure 8.20 shows the comparison of the data allocation schemes in terms of system cost which is plotted against the data allocation problem (DAP) size. DAP size is the optimal number of data fragments allocated to ensure the performance of the system without any overhead. When the DAP size increases, the cost increases as the number of allocations resembles the number of users or client locations and hence it tends to increases with categorization. When compared with the existing models, CSO based data allocation is significantly better in cost due to the optimal solutions which are better than GA, ACO and PSO. Hence the cost of utilizing CSO for data allocation is also under justifiable limits.

Network transfer cost: Network transfer cost (NTC) is the cost incurred in transferring the queries to the servers and receiving the response data. NTC is considered in this model mainly because it determines the cost due to object movement in the network that has significant impact in system performance while the communication cost has little effect on the overall performance.

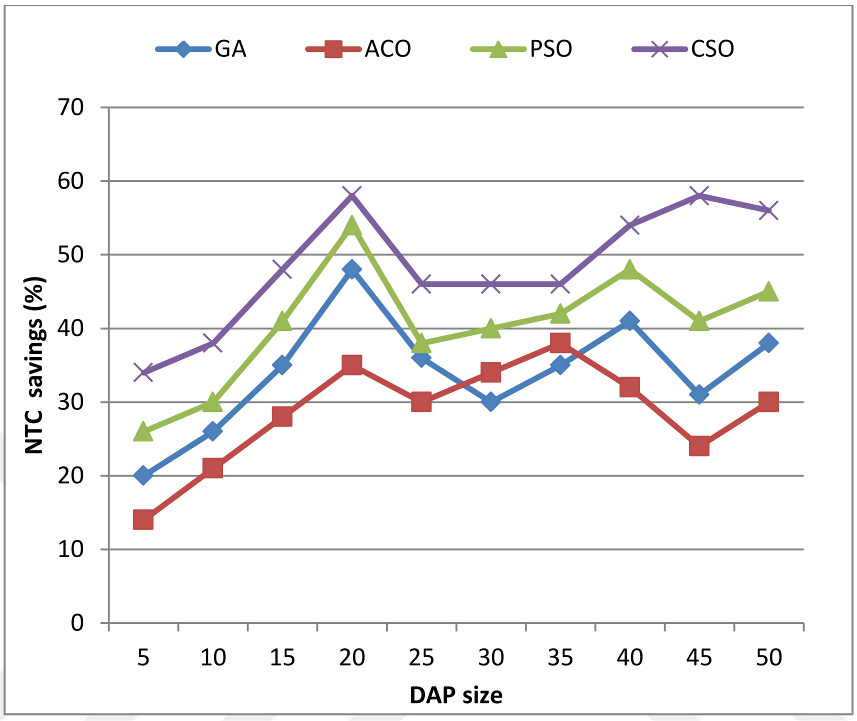


Figure 8.21 NTC savings vs. DAP size

Figure 8.21 shows the comparison of data allocation schemes in terms of NTC savings plotted against the DAP size. NTC cost is incurred mainly due to the transfer of data objects between two nodes in a network and the effective allocation of data fragments can reduce the number of fragments allocated and minimize the node data transmissions. This process can significantly reduce the cost and results in resource conservation. The main aim in considering this parameter in addition to system cost is because to identify the cost saving mechanism in data allocation. From the figure it can be seen that the CSO based data allocation has higher NTC savings which is about 9-10% greater than the second best PSO based model. This is possible most probably due to the lesser iterations of CSO and reduced communication between the nodes irrespective of best allocation process.

Execution time: It is the time taken from the output of fragmentation to the final allocation of each data fragment. It is essential in determining the system runtime and subsequent power consumption incurred during the allocation process.

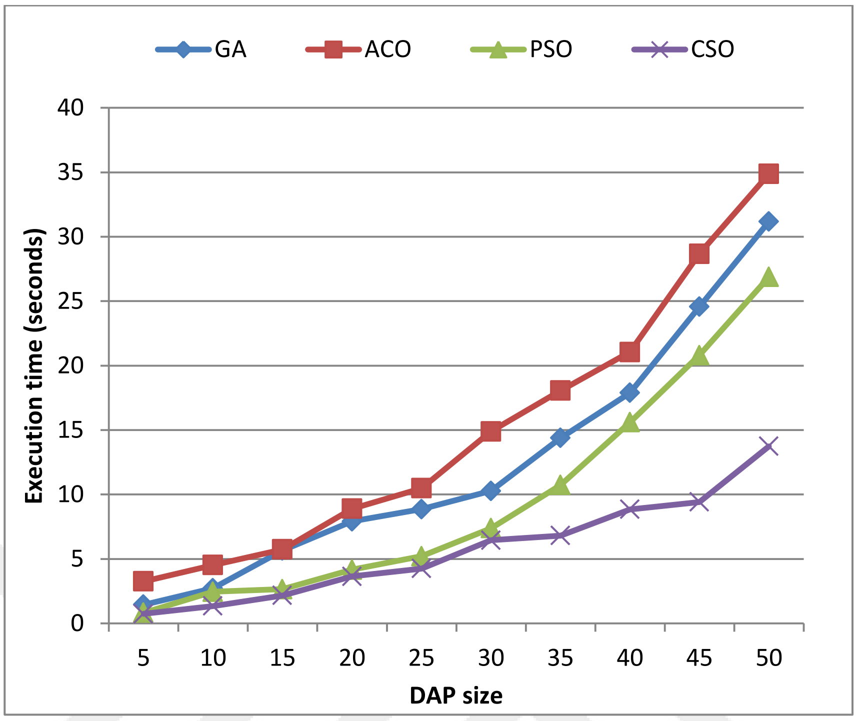


Figure 8.22 Execution time vs. DAP size

Figure 8.22 shows the comparison of the data allocation schemes in terms of execution time plotted against the DAP size. The execution time for completing this data allocation process is evaluated from the beginning of data allocation (i.e. the end of fragmentation) to the end of allocation of all fragments. However the time taken for complete allocation is affected by the traffic and delay due to query waiting. From the figure, it can be seen that the presented model of CSO based data allocation performs better with less execution time. The reduced execution time is mainly based on the minimized iterations in CSO.

Maximum memory usage: This parameter is similar to the cost value. It is pretended to be the utmost utilization of memory to the efficiency of the system without much wastage.

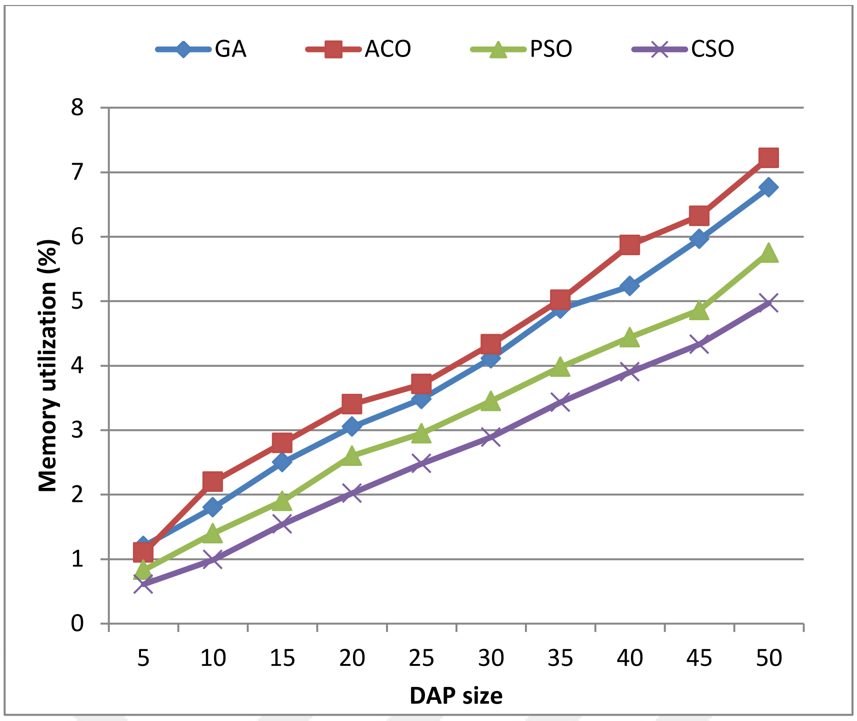


Figure 8.23 Maximum memory usage vs. DAP size

Figure 8.23 shows the comparison of data allocation schemes in terms of maximum memory usage against the DAP size. As the memory utilization and cost are similar parameters, the same ideology applies here. The increase in the DAP size tends to increase the memory utilization while the presented model of CSO tends to increase the performance of the overall system. In this figure, it can be seen that the CSO based model has less memory usage in spite of increased DAP size but does not affect the overall performance. The evaluation results prove that the presented data allocation model using CSO algorithm is highly efficient than the other models considered which has been verified through the comparison results.

8.5 COMPARISON OF DATA REPLICATION TECHNIQUES

The performance of the data replication techniques is evaluated and compared with the existing models namely Least Frequently Used (LFU), Least Recently Used (LRU), 3-Level Hierarchical Algorithm (3LHA) and Bandwidth Hierarchy based

Replication algorithm (BHR). The snapshot replication and merge replication are also evaluated and their results are displayed.

8.5.1 Snapshot replication

The case study for snapshot replication is presented. A first step, a server (DESKTOP-33FQBBL) is used with multi database. In this situation, the measures are checked on two databases as shown in Table 8.1.

Table 8.1 Calculation of transfer time

Location of publisher	Location of subscriber	Size of DB	No. of tables in DB	No. of tables	Synchronization time for publisher
DESKTOP-33FQBBL	INSTANCE3	1468.00 MB	31	31	00:10,78
DESKTOP-33FQBBL	INSTANCE3	16.00 MB	18	2	00:07,03
DESKTOP-33FQBBL	122.175.53.112	1468.00 MB	31	12	00:13,07

As the second step, the load time is checked when full database is sent from server to single client as shown in Table 8.2.

Table 8.2 Load of transfer database

Publisher name	Subscriber name	Size of DB	No. of tables	Synchronization time for publisher
SnapshotPublisherName	SnapshotRepSubscriber	1468.00 MB	31	00:10,78
Snap Rep. Pub	Snap Rep. sub.	11.13 MB	19	00:07,03

In the final step, multi rows are reorganized from publisher to subscriber using horizontal fragmentation and update multiple columns from publisher to subscriber by means of vertical fragmentation as shown in Table 8.3**.**

Table 8.3 Update of fragmentation database

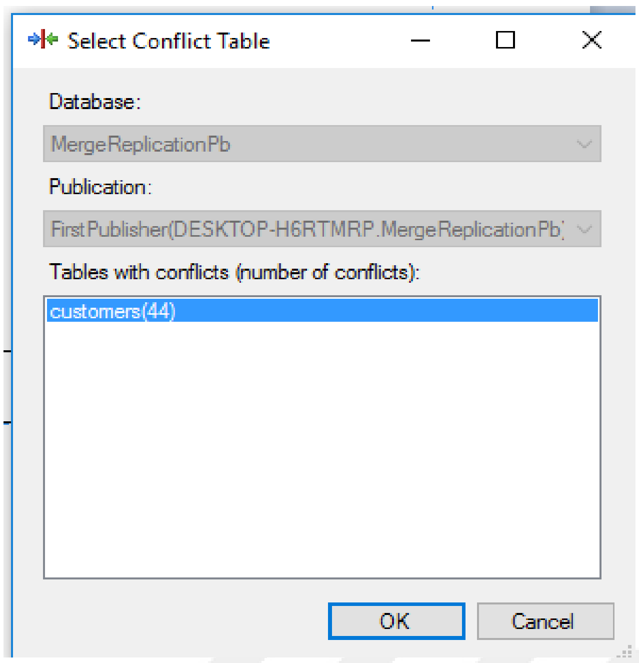
Server name	Client name	Name of DB	No. of rows updated	Synchronization time for publisher	Type of fragmentation
DESKTOP-33FQBBL	INSTANCE2	Snapshot auto transfer data subscribe	143	01:03,04	Horizontal
DESKTOP-33FQBBL	INSTANCE3	Snapshot horizontal frag. Sub.	160	00:23,02	Vertical

8.5.2 Merge Replication

Merge replication is applied to setup for using three replicas (e.i. MergeReplicationPb, MergeReplicationSB1 and MergeReplicationSB2).Each replica contains one table named*Customers*. It includes 90 rows and 7 columns (with name CustomerName, ContactName, Adress, City, PostalCode, Country and rowguid). Table 7.4 shows the resulting conflicts in the system.

Table 8.4 Testing Conflicts in Merge Replication

Subscription	No. of rows in DB	No. of rows update	No. of conflicts	No. of conflicts winner	No. of conflict loser	No. of updates rows	Synchronization time for publisher
MergeReplicationSB1	90	50	44	30	20	29	00:43,02
MergeReplicationSB2	90	70	44	20	0	70	00:50,32



Select Conflict Table

Database:

MergeReplicationPb

Publication:

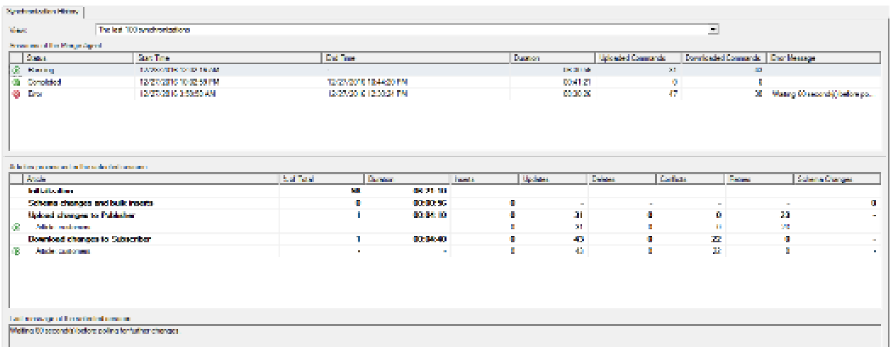
FirstPublisher(DESKTOP-H6RTMRP.MergeReplicationPb)

Tables with conflicts (number of conflicts):

customers(44)

Figure 8.24 Show Conflict table

Figure 8.24 shows the conflict table. These conflicts happened because the updates occur at the same fields almost at the same time. The merge replication accepts last arriving update request of two updates. In the test, subscriber1 (MergeReplicationSB1) has requested an update of 50 rows to publisher. Then almost at the same time subscriber2 (MergeReplicationSB2) has requested an update of 70 rows. The publisher accepts update request from MergeReplicationSB1. When it was executing that update a new update request on the same place arrives from MergeReplicationSB2. In this situation, the publisher accepts last update from MergeReplicationSB2 reports 44 conflicts in the replication systems.



Synchronization History

View: The last 100 synchronizations

Sessions of the Merge Agent

Status	Start Time	End Time	Duration	Uploaded Commands	Downloaded Commands	Error Message
Running	[illegible]	-	[illegible]	[illegible]	[illegible]	
Completed	12/27/2016 10:02:59 PM	12/27/2016 10:44:20 PM	00:41:21	0	0	
Error	12/27/2016 3:53:50 AM	12/27/2016 12:30:24 PM	00:30:28	47	36	Waiting 60 second(s) before po...

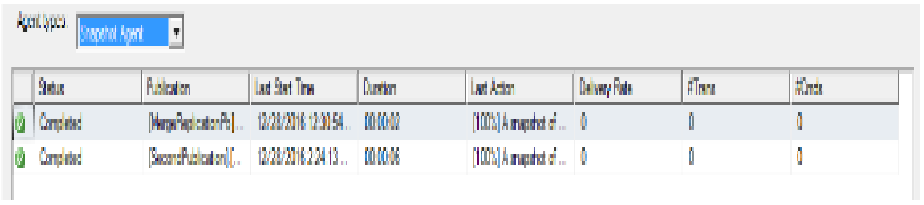
Actions in the selected session

Article	% of Total	Duration	Inserts	Updates	Deletes	Conflicts	Retries	Schema Changes
Initialization	98	[illegible]						
Schema changes and bulk inserts	0	00:00:56	0	-	-	-	-	0
Upload changes to Publisher	1	00:04:10	0	31	0	0	23	-
Article: customers			0	31	0	[illegible]	[illegible]	
Download changes to Subscriber	1	00:04:40	0	43	0	22	0	-
Article: customers	-	-	0	43	0	22	0	-

Last message of the selected session

Waiting 60 second(s) before polling for further changes.

Figure 8.25 Synchronization



Agent types: Snapshot Agent

	Status	Publication	Last Start Time	Duration	Last Action	Delivery Rate	#Trans	#Cmds
✓	Completed	[MergeReplicationPb]...	12/28/2016 12:30:54...	00:00:02	[100%] A snapshot of ...	0	0	0
✓	Completed	[SecondPublication]:[...	12/28/2016 2:24:13 ...	00:00:06	[100%] A snapshot of ...	0	0	0

Figure 8.26 Status of Snapshot Agent

After requesting the synchronizations between subscribers and publisher get the status are shown Figure 8.25. Snapshots agents are shown in Figure 8.26

8.5.3 MGSO based dynamic data replication

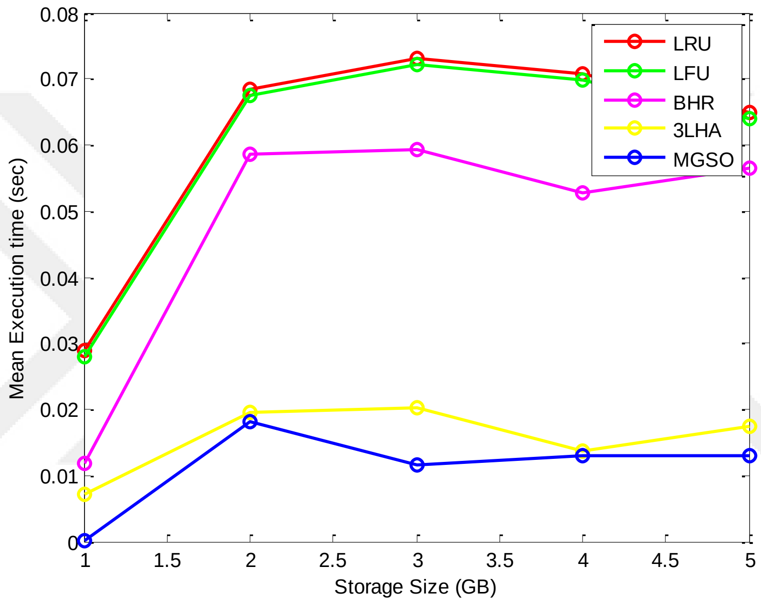


Figure 8.27 Storage Size vs. Mean Execution time

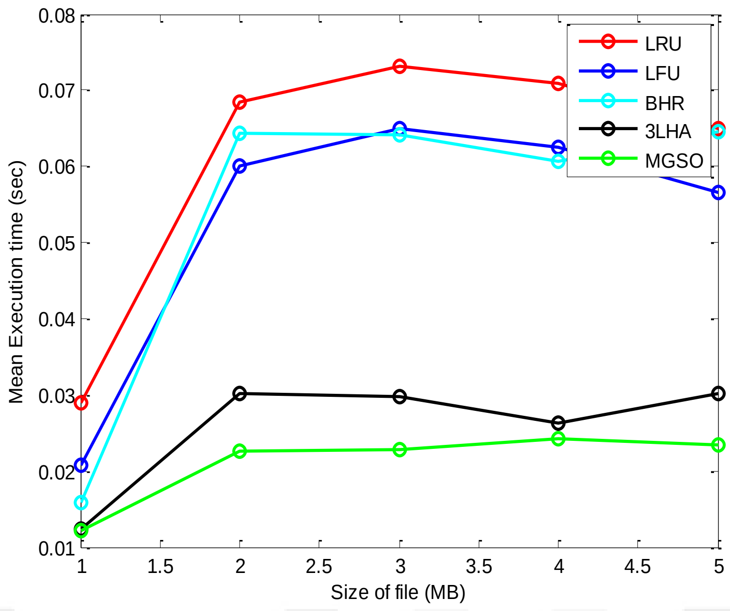


Figure 8.28 Size of file vs. Mean Execution time

Figure 8.27 demonstrates the mean execution time evaluation of the replication algorithms against the storage size, while figure 7.28 illustrates the evaluation of the algorithms in terms of mean execution time against the file size. It can be observed that the MGSO based dynamic replication algorithm has lesser execution time when compared to the other algorithms even when larger files are employed for replication. The reason that can validate this deviation is the fact that the window mechanism and optimal replica selection concept significantly reduce the execution time.

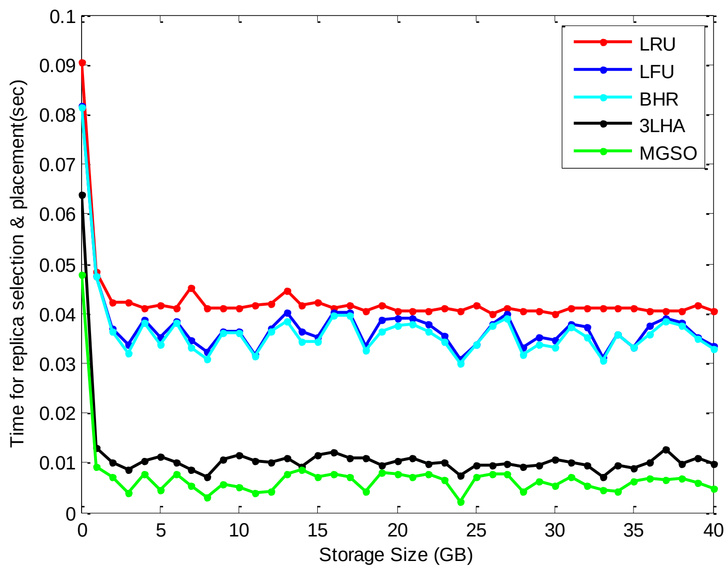


Figure 8.29 Execution time of replica selection and placement

Figure 8.29 displays the execution time for replica selection and placement. The total time of the former algorithms are significantly higher than the MGSO based model because MGSO has better optimal replica selection within less time. This reason significantly improves the performance with less execution time.

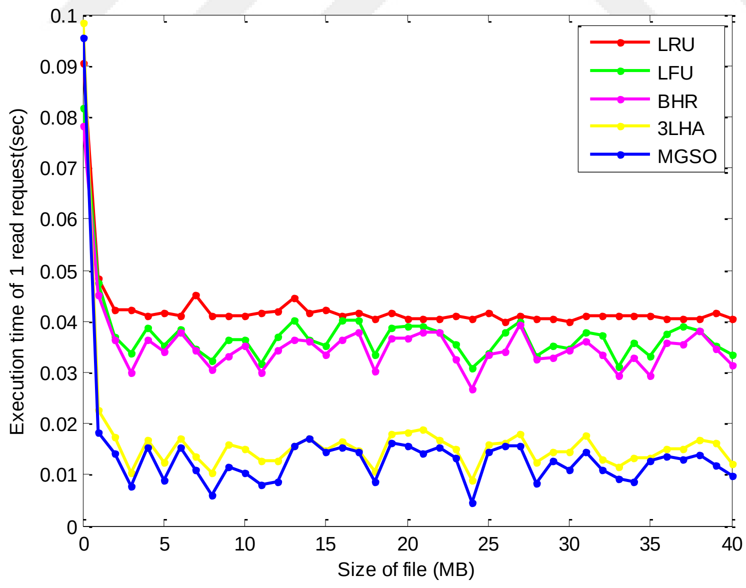


Figure 8.30 Execution time of 1 read request for different file sizes

Similarly, the Figure 8.30 shows the execution time of 1 read request for diverse file sizes in which MGSO model consumes less time. The execution time is significantly less even when the file size is increased. The main reason for this efficiency is that the MGSO has acceleration of performance in increasing the processing speed of the tasks through optimal replica placement. Even though this approach incurs some negligible overhead due to replica operations, the enhancement in execution time helps this method to overcome the former methods. Thus, from the performance evaluation, it can be concluded that the proposed MGSO based dynamic replication algorithm is the better algorithm.

8.6 CHAPTER SUMMARY

This chapter intended to provide the evaluation results of the presented research models. It also provided the comparison of the developed data fragmentation, allocation and replication techniques with the existing models. As a result, the performance improvement of the proposed models is visualized and the theoretical study of each model is justified.

REFERENCES

[1] C. J., Date, "An introduction to database systems. "*Pearson education india,* 2006*.*

[2] C. Tenopir, Allard, S., Douglass, K., Aydinoglu, A. U., Wu, L., Read, E., and Frame, M., "Data sharing by scientists: practices and perceptions," *PloS one,* vol.6, no. 6, e21101, 2011*.*

[3] O. M., Tamer, "Principles of distributed database systems," *Pearson education india,*2006.

[4] M. T., Özsu, and Valduriez, P., "Principles of distributed database systems," *Springer science and business media,*2011.

[5] R., Elmasri, "Fundamentals of database systems," *Pearson education india*,2008.

[6] S. K., Tiwari, Sharma, A. K., and Swaroop, V., "Issues in replicated data for distributed real- time DBS," *International journal of computer science and IT,* vol.2, no. 4, pp*. 1364-1371,* 2011.

[7] S., Ceri, "Distributed databases." *Tata McGraw-Hill education,* 2017.

[8] S. K., Rahimi, and Haug, F. S., "Distributed database management systems: A Practical Approach," *John Wiley and Sons,*2010.

[9] C., Kavitha, Sadasivam, G. S., and Shenoy, S. N., "Ontology based semantic integration of heterogeneous databases," *European journal of scientific research,* vol.64, no. 1, pp. 115-122, 2011.

[10] A., Bouguettaya, Benatallah, B., and Elmagarmid, A. K., "Interconnecting heterogeneous information systems,", *Springer science and business media, vol. 14,2012.*

[11] R., Elmasri, and Navathe, S., "Fundamentals of database systems," *Addison-Wesley publishing company,*2010.

[12] Ma, X., Baresi, L., Ghezzi, C., Panzica La Manna, V., and Lu, J., "Version-consistent dynamic reconfiguration of component-based distributed systems," *in proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on foundations of software engineering, pp. 245-255, ACM,* 2011.

[13] S., Ghosh, "Distributed systems: an algorithmic approach,"*chapman and Hall/CRC,*2014.

[14] C., Gao, Liu, J., and Zhong, N., "Network immunization with distributed autonomy-oriented entities," *IEEE transactions on parallel and distributed systems,* vol.22, no. 7, pp. 1222-1229, 2011.

[15] N. H., Ryeng, Hauglid, J. O., and Nørvåg, K., "Site-autonomous distributed semantic caching," *In Proceedings of the 2011 ACM symposium on applied computing, pp. 1015-1021, ACM,* 2011.

[16] C., Yang, Yen, C., Tan, C., and Madden, S. R., "Osprey: implementing MapReduce-style fault tolerance in a shared-nothing distributed database," *in IEEE 26th international conference on data engineering (ICDE), IEEE,* pp. 657-668, 2010.

[17] J., Kuhlenkamp, Klems, M., and Röss, O., "Benchmarking scalability and elasticity of distributed database systems," *Proceedings of the VLDB Endowment,* vol.7, no. 12, pp. 1219-1230, 2014.

[18] S., Ilarri, Mena, E., and Illarramendi, A., "Location-dependent query processing: Where we are and where we are heading," *ACM computing surveys (CSUR),* vol.42, no. 3, pp. 12,2010.

[19] Anderson, R. J., "Security engineering: a guide to building dependable distributed systems," *John Wiley and Sons,*2010.

[20] L., Okman, Gal-Oz, N., Gonen, Y., Gudes, E., and Abramov, J.," Security issues in NoSQL databases," In trust, security and privacy in computing and communications (TrustCom), *2011 IEEE 10th international conference on, IEEE, pp. 541-547,*2011.

[21] M. J. Franklin," Client data caching: A foundation for high performance object database systems," *Springer science and business media,* vol. 354,2012.

[22] W., Lu, M., Zhou, Guo, K., Pan, X., and Ma, J.," An effective and economical architecture for semantic-based heterogeneous multimedia big data retrieval," *Journal of systems and software,* 102, 207-216, 2015.

[23] A. E., Agrawal, D., Abbadi, Emekci, F., and Metwally, A.," Database Management as a Service: Challenges and Opportunities," *In proceedings of the 2009 IEEE international conference on data engineering, IEEE computer society,* pp. 1709-1716,2009.

[24] D. S., Kim, W., Reiner and Batory, D. (Eds.)," Query processing in database systems," *Springer science and business media*,2012.

[25] D. J., Abadi," Consistency tradeoffs in modern distributed database system design: CAP is only part of the story," *IEEE computer society press los Alamitos, CA, USA computer,* vol.45 (2), 37-42,2012.

[26] W., Cellary, Morzy, T., and Gelenbe, E.," Concurrency control in distributed database systems," *Elsevier,* vol. 3,2014.

[27] A., Cuzzocrea, Darmont, J., and Mahboubi, H.," Fragmenting very large XML data warehouses via K-means clustering algorithm," *International journal of business intelligence and data mining,* vol.4, no.3/4, pp. 301-328,2009.

[28] R, R., Pazos, Vázquez A, G., Martínez F, J., and Pérez O, J.," Vertical Fragmentation Design of Distributed Databases Considering the Nonlinear Nature of Roundtrip Response Time," *Knowledge-based and intelligent information and engineering systems,* pp.173-182,2010.

[29] J. O., Hauglid, Ryeng, N. H., and Nørvåg, K.," DYFRAM: dynamic fragmentation and replica management in distributed database systems," *distributed and parallel databases,* vol.282, no. 4, pp. 157-185,2010.

[30] A. A. C., Fauzi, Noraziah, A., and Zin, N. M.," A Novel Binary Vote Assignment

Grid Quorum Algorithm for Distributed Database Fragmentation," *in digital enterprise and information systems, Springer, berlin, heidelberg,* pp. 339-349,2011.

[31] D., Boukraâ, Boussaïd, O., and Bentayeb, F.," Vertical fragmentation of XML data warehouses using frequent path sets," *in international conference on data warehousing and knowledge discovery, Springer, berlin, heidelberg,* pp. 196-207,2011.

[32] M., Goli, and Rankoohi, S. M. T. R.," A new vertical fragmentation algorithm based on ant collective behavior in distributed database systems," *knowledge and information systems*, vol.30, no. 2, pp. 435-455,2012.

[33] R. A., Pazos, Vázquez, G., Martínez, J. A., Pérez-Ortega, J., and Martínez-Luna, G.," Minimizing roundtrip response time in distributed databases with vertical fragmentation," *journal of computational and applied mathematics,* vol.259, pp.905-913,2014.

[34] L., Wiese," Clustering-based fragmentation and data replication for flexible query answering in distributed databases," *journal of cloud computing*, vol.3, no. 1, pp. 18,2014.

[35] R., Dharavath, Kumar, V., Kumar, C., and Kumar, A.," An Apriori-Based Vertical Fragmentation Technique for Heterogeneous Distributed Database Transactions," *in intelligent computing, networking, and informatics, Springer, new delhi,* pp. 687-695,2014*.*

[36] V. N., Luong, Nguyen, H. H. C., and Le, V. S.," An improvement on fragmentation in distribution database design based on knowledge-oriented clustering techniques," *arXiv preprint arXiv:*1505.01535, 2015.

[37] N., Thakur and Ram, B.," Examining the Performance of Vertical Fragmentation using FP-MAX Algorithm," *international journal of computer applications,* vol.116, no.23, pp. 43-48, 2015.

[38] J. M., Hadden and Mahdy, A. M.," The Royal Split Paradigm: Real-Time Data Fragmentation and Distributed Networks for Data Loss

Prevention,” *international journal of computer science and security (IJCSS),* vol.10, no. 3, pp. 107, 2016.

[39] N. T., Mai,” Heuristic Algorithm for Fragmentation and Allocation in Distributed Object-Oriented Database,” *journal of computer science and cybernetics,* vol.32, no. 1, pp. 47-60, 2016.

[40] A. A., Amer, Sewisy, A. A., and Elgendy, T. M.,” An optimized approach for simultaneous horizontal data fragmentation and allocation in Distributed Database Systems (DDBSs),” *Heliyon*, vol.3, no. 12, e00487, 2017.

[41] A., Drissi, Nait-Bahloul, S., Benouaret, K., and Benslimane, D.,” Horizontal fragmentation for fuzzy querying databases, “*distributed and parallel databases,* p.1-28, 2017.

[42] G. H., Eladl,” A Modified Vertical Fragmentation Strategy for Distributed Database,” *international journal of computer science and network,* vol.6, no. 4, pp. 505-508, 2017.

[43] A. A., Sewisy, Amer, A. A., and Abdalla, H. I.,” A Novel Query-Driven Clustering-Based Technique for Vertical Fragmentation and Allocation in Distributed Database Systems,” *international journal on semantic web and information systems (IJSWIS),* vol.13, no. 2, pp. 27-54, 2017.

[44] V. N., Luong,” Fragmentation in Distributed Database Design Based on KR Rough Clustering Technique,” *in context-aware systems and applications, and nature of computation and communication, Springer, cham,* pp. 166-172, 2017.

[45] S., Mehta, Agarwal, P., Shrivastava, P., and Barlawala, J.,” Differential bond energy algorithm for optimal vertical fragmentation of distributed databases, “*journal of king saud university-computer and information sciences,* 2018.

[46] I. O., Hababeh, Ramachandran, M., and Bowring, N.,” A high-performance computing method for data allocation in distributed database systems, “*the journal of supercomputing*, vol.39, no. 1, pp. 3-18, 2007.

[47] R. K., Adl, and Rankoohi, S. M. T. R.," A new ant colony optimization-based algorithm for data allocation problem in distributed databases," *knowledge and information systems,* vol.20, no. 3, pp. 349-373, 2009.

[48] A. S., Mamaghani, Mahi, M., Meybodi, M. R., and Moghaddam, M. H.," A novel evolutionary algorithm for solving static data allocation problem in distributed database systems, "*in 2010 second international conference on network applications, protocols and services, IEEE,* pp. 14-19,2010.

[49] W. K., Mashwani, and Salhi, A.," A decomposition-based hybrid multiobjective evolutionary algorithm with dynamic resource allocation, "*applied soft computing,* vol.12, no. 9, pp. 2765-2780, 2012.

[50] Li, S. P., and Wong, M. H.," Data allocation in scalable distributed database systems based on time series forecasting," *in 2013 IEEE international congress on big data (BigData Congress), IEEE,* pp. 17-24, 2013.

[51] K. S., Singh, A., Kahlon, and Virk, R. S.," Nonreplicated static data allocation in distributed databases using biogeography-based optimization," *Chinese journal of engineering,* 2014.

[52] R. D., More, and Nasik, A.,"Energy Efficient Task Scheduling and Data Allocation Strategy in Heterogeneous Environment with Real Time Constraints," *energy,* vol.100,no.11, 2014.

[53] A., Singh," Empirical Evaluation of Threshold and Time Constraint Algorithm for Non-replicated Dynamic Data Allocation in Distributed Database Systems," *in proceedings of the international congress on information and communication technology, Springer, Singapore*, pp. 131-138, 2016.

[54] G., Sen, Krishnamoorthy, M., Rangaraj, N., and Narayanan, V.," Mathematical models and empirical analysis of a simulated annealing approach for two variants of the static data segment allocation problem", *Networks,* vol.68, no. 1, pp. 4-22, 2016.

[55] V., Malyshkin, Perepelkin, V., and Schukin, G.," Scalable distributed data

allocation in LuNA fragmented programming system," *the journal of supercomputing,* vol.73, no. 2, pp. 726-732, 2017.

[56] R., Kumar, and Gupta, N.,"Clustered Approach to Dynamic Data Allocation in Distributed Database Systems,"*international journal of advanced research in computer science*, vol.8, no. 5, 2017.

[57] A. S., Darabant, Tambulea, L., and Varga, V." Access Patterns Optimization in Distributed Databases Using Data Reallocation," *in international conference on database and expert systems applications, Springer, Cham, pp. 178-* 186,2017.

[58] Q., Sun, Deng, B., Fu, L., and Sun, J.,"Non-redundant Distributed Database Allocation Technology Research," *in 2017 international conference on computing intelligence and information system (CIIS), IEEE, pp. 155-159,* 2017.

[59] V., Kulba, and Somov, S.," Problem of optimal placement of data files in large-scale unreliable distributed systems," *in 2017 tenth international conference management of large-scale system development (MLSD), IEEE,* pp. 1-5, 2017.

[60] M., Mahi, Baykan, O. K., and Kodaz, H.," A new approach based on particle swarm optimization algorithm for solving data allocation problem," *applied soft computing,* 62, p.571-578, 2018.

[61] N. K. Z., Lwin, and Naing, T. M.," Non-Redundant Dynamic Fragment Allocation with Horizontal Partition in Distributed Database System, "*in 2018 international conference on intelligent informatics and biomedical sciences (ICIIBMS) , IEEE,* Vol. 3, pp. 300-305,2018.

[62] C., Curino, Jones, E., Zhang, Y., and Madden, S.," Schism: a workload-driven approach to database replication and partitioning," *proceedings of the VLDB endowment,* pp. 48-57, 2010.

[63] C. L., Abad, Lu, Y., and Campbell, R. H.," DARE: Adaptive data replication for efficient cluster scheduling," *in 2011 IEEE international conference*

on cluster computing (CLUSTER), IEEE, pp. 159-168, 2011.

[64] T. M., Naing, and Win, A.,” Enforcing Scalability: An Efficient Approach for Distributed Database Partial Replication,” *international conference on advances in engineering and technology (ICAET'2014*), pp. 370-374,2014.

[65] F., Xhafa, Potlog, A. D., Spaho, E., Pop, F., Cristea, V., and Barolli, L.,” Evaluation of intra-group optimistic data replication in P2P groupware systems,” *Concurrency and Computation: Practice and Experience,* vol.27, no. 4, pp. 870-881,2015.

[66] T., Crain, and Shapiro, M.,” Designing a causally consistent protocol for geo-distributed partial replication,” *in proceedings of the first workshop on principles and practice of consistency for distributed data, ACM*, p. 6,2015.

[67] S., Varma, and Khatri, G.,” QoS-Aware Data Replication in Hadoop Distributed File System.,” *international journal of advanced networking and applications,* vol.7, no. 3, pp. 2741, 2015.

[68] G., Liu, Shen, H., and Chandler, H.,” Selective data replication for online social networks with distributed datacenters,” *IEEE transactions on parallel and distributed systems,* vol.27, no. 8, pp. 2377-2393, 2016.

[69] S., Warhade, Dahiwale, P., and Raghuwanshi, M. M.,” A dynamic data replication in grid system,”*procedia computer science*, 78, 537-543,2016.

[70] Z., Yang, Bhimani, J., Wang, J., Evans, D., and Mi, N.,” Automatic and scalable data replication manager in distributed computation and storage infrastructure of cyber-physical systems,” journal *of scalable computing, special issue on communication, computing, and networking in cyber-physical systems,* vol.18, no. 4, 2017.

[71] W., Sun, Simon, V., Monnet, S., Robert, P., and Sens, P.,” Analysis of a stochastic model of replication in large distributed storage systems: A mean-field approach,” *proceedings of the ACM on measurement and analysis of computing systems,* vol.1, no. 1, pp. 24, 2017.

[72] J., Zhu, Huang, C., Fan, X., Guo, S., and Fu, B.,” EDDA: An efficient distributed

data replication algorithm in VANETs," *sensors,* vol.18, no. 2, pp. 547,2018.

[73] V., Cardellini, Lo Presti, F., Nardelli, M., and Russo Russo, G.,"Optimal operator deployment and replication for elastic distributed data stream processing, "*concurrency and computation: practice and experience,* vol.30, no. 9, e4334, 2018*.*

[74] L., Xiong, Yang, L., Tao, Y., Xu, J., and Zhao, L.," Replication Strategy for Spatiotemporal Data Based on Distributed Caching System," *sensors,* vol.18, no. 1, pp. 222, 2018.

[75] T. V., Kumar, Singh, V., and Verma, A. K.," Distributed query processing plans generation using genetic algorithm," *international journal of computer theory and engineering*, vol.3, no. 1, pp. 38, 2011.

[76] K. U.," Sattler," Distributed Query Processing. In Encyclopedia of Database Systems", *Springer, Boston, MA,* pp. 912-917, 2009.

[77] B. B., Cambazoglu, Kayaaslan, E., Jonassen, S., and Aykanat, C.," A term-based inverted index partitioning model for efficient distributed query processing," *transactions on the web (TWEB), ACM* , vol.7, no. 3, pp. 15, 2013.

[78] S. R., Pettifer, and Attwood, T. K.," Distributed Query Processing," *in encyclopedia of systems biology, springer, new york, NY,pp.* 604-605,2013.

[79] W. H., Tok," Distributed Transaction Management," *in encyclopedia of database systems. 925-929, Springer, Boston, MA*,2009.

[80] Y., Breitbart, Garcia-Molina, H., and Silberschatz, A.," Overview of multidata base transaction management," *in cascon first decade high impact papers. 93*-126, IBM Corp,2010.

[81] T., Wang, Vonk, J., Kratz, B., and Grefen, P.,"A survey on the history of transaction management: from flat to grid transactions," *distributed and parallel databases,* vol.23, no. 3, pp. 235-270,2008.

[82] M., Kaur, and Kaur, H.," Concurrency control in distributed database system", *international journal of advanced research in computer science and*

software engineering ISSN, 2277, 2013.

[83] W., Stallings, Brown, L., Bauer, M. D., and Bhattacharjee, A. K.," *Computer security: principles and practice,", Pearson Education,* pp. 978-0 ,2012.

[84] H. F., Tipton, and Nozaki, M. K.," Information security management handbook," *CRC press,* 2007.

[85] X., Huang, Xiang, Y., Chonka, A., Zhou, J., and Deng, R. H.," A generic framework for three-factor authentication: Preserving security and privacy in distributed systems," *IEEE transactions on parallel and distributed systems,* vol.22, no. 8, pp. 1390-1397, 2011.

[86] A., Thomson, and Abadi, D. J.," CalvinFS: Consistent WAN Replication and Scalable Metadata Management for Distributed File Systems,"*In FAST ,pp. 1-*14, 2015.

[87] S., Kobbe, Bauer, L., Lohmann, D., Schröder-Preikschat, W., and Henkel, J. DistRM: distributed resource management for on-chip many-core systems. *In Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, ACM,* pp. 119-128, 2011.

[88] S., Jeon, Bang, J., Byun, K., and Lee, S.," A recovery method of deleted record for SQLite database," *personal and ubiquitous computing,* vol.16, no. 6, pp., 707-715, 2012.

[89] B.,Bryla," Oracle database 11g DBA handbook," *Tata McGraw-Hill education,* 2007.

[90] S., Angelov, Grefen, P., and Greefhorst, D.," A framework for analysis and design of software reference architectures," *information and software technology,* vol.54, no. 4, pp. 417-431, 2012.

[91] M., Alam, and Shakil, K. A.," Cloud database management system architecture," *UACEE international journal of computer science and its applications,* vol.3, no. 1, pp. 27-31, 2013.

[92] D., Agrawal, El Abbadi, A., Das, S., and Elmore, A. J.," Database scalability, elasticity, and autonomy in the cloud," *in international conference on*

database systems for advanced applications, Springer, berlin, heidelberg,pp. 2-15, 2011.

[93] T., Lin, Zhou, K., and Wang, S.," Cloudlet-screen computing: a client-server architecture with top graphics performance," *international journal of ad hoc and ubiquitous computing, vol.13, no. 2, pp. 96-108, 2013.*

[94] X., Shen, Yu, H., Buford, J., and Akon, M. (Eds.).," *Handbook of peer-to-peer networking," Springer science and business media,* vol. 34, 2010.

[95] T., Karnagel, Habich, D., Schlegel, B., and Lehner, W.," Heterogeneity-aware operator placement in column-store DBMS," *datenbank-spektrum*, vol.14, no. 3, pp. 211-221, 2014.

[96] R., Ghaemi, Fard, A. M., Tabatabaee, H., and Sadeghizadeh, M.," Evolutionary query optimization for heterogeneous distributed database systems," *world academy of science,* 43, 43-49, 2008.

[97] A. B., GadichaAlvi, A. S., Gadicha, V. B., and Zaki, S. M.," Top-Down Approach Process Built on Conceptual Design to Physical Design Using LIS, GCS Schema."*international journal of engineering sciences and emerging technologies*, 3, 90-96,2012.

[98] A., Mwombeki," Using an Interaction Model as a Requirement for Distributed Database Design to Enhance Data Access and Reliability in Higher Learning Institutions: A Study Conducted at Saint Augustine University of Tanzania," *interactions,* 2015.

[99] S. I., Khan, and Hoque, A. S. M. L.," A new technique for database fragmentation in distributed systems," *international journal of computer applications,* 2015.

[100] P. R., Bhuyar, Gawande, A. D., and Deshmukh, A. B.," Horizontal fragmentation technique in distributed database," *international journal of scientific and research publications*, vol.2, no. 5, pp. 1-7, 2012.

[101] S., Gupta, and Panda, S.," Vertical Fragmentation, Allocation and Re-Fragmentation in Distributed Object Relational Database Systems-(Update

Queries Included),” *international journal of engineering research and development,* vol.4, no. 7, pp. 45-52, 2012.

[102] N., Mukherjee,” Synthesis of non-replicated dynamic fragment allocation algorithm in distributed database systems,” *ACEEE Int. J. on Information Technology,* vol.1, no. 10, 2011.

[103] S., Jagannatha, Mrunalini, M., Kumar, T., and Kanth, K.,” Modeling of mixed fragmentation in distributed database using UML 2.0,” *IPCSIT,* v. 2, 190-194, 2011.

[104] U., Tosun, Dokeroglu, T., and Cosar, A.,” Heuristic algorithms for fragment allocation in a distributed database system,” *In Computer and Information Sciences III, Springer, London,* pp. 401-408, 2013.

[105] D. C., Gope,” Dynamic data allocation methods in distributed database system,” *American academic and scholarly research journal,* vol. 4, 1, 2012.

[106] A., Sleit, AlMobaideen, W., Al-Areqi, S., and Yahya, A.,” A dynamic object fragmentation and replication algorithm in distributed database systems,” *American journal of applied sciences,* vol.4no.8, pp. 613-618, 2007.

[107] H. I., Abdalla, and Amer, A. A.,” Dynamic horizontal fragmentation, replication and allocation model in DDBSs,” *In Information Technology and e-Services (ICITeS), 2012 International Conference on, IEEE,* pp. 1-7, 2012.

[108] C. S., Kumar, Seetha, J., and Vinotha, S. R.,” Security Implications of Distributed Database Management System Models,” *arXiv preprint* arXiv:1401.7733, 2014.

[109] M. T., Quasim,” Security Issues in Distributed Database System Model,” *Compusoft,* vol.2, no. 12, pp. 396, 2013.

[110] M. T., Quasim,” Security Issues in Distributed Database System Model,” *Compusoft,* vol.2, no. 12, pp.396, 2013.

[111] W., Li, J., Fan, and Tang, N.,” Incremental detection of inconsistencies in distributed data,” *IEEE transactions on knowledge and data engineering,* vol.26, no. 6, pp. 1367-1383, 2014.

[112] C., Ordonez," Integrating K-means clustering with a relational DBMS using SQL," *IEEE transactions on knowledge and data engineering*, vol.18, no.2, pp. 188-201, 2006.

[113] M., Kaczmarczyk, Barczynski, M., Kilian, W., and Dubnicki, C.," Reducing impact of data fragmentation caused by in-line deduplication," *in proceedings of the 5th annual international systems and storage conference, ACM,* p. 15,2012.

[114] N., Pelekis, Tampakis, P., Vodas, M., Panagiotakis, C., and Theodoridis, Y.," In-DBMS Sampling-based Sub-trajectory Clustering," *in EDBT,* pp. 632-643, 2017.

[115] A., Bouguettaya, Yu, Q., Liu, X., Zhou, X., and Song, A.," Efficient agglomerative hierarchical clustering," *expert systems with applications,* 42(5), 2785-2797, 2015.

[116] M. L., Zepeda-Mendoza, and Resendis-Antonio, O.," Hierarchical agglomerative clustering," *In Encyclopedia of Systems Biology, Springer, New York, NY*, pp. 886-887, 2013.

[117] C. M., Hwang, Yang, M. S., and Hung, W. L.," New similarity measures of intuitionistic fuzzy sets based on the Jaccard index with its application to clustering," *international journal of intelligent systems,* vol.33, no.8, pp. 1672-1688, 2018.

[118] I. A., Pagnuco, Pastore, J. I., Abras, G., Brun, M., and Ballarin, V. L.," Analysis of genetic association in Listeria and Diabetes using Hierarchical Clustering and Silhouette Index," *in journal of physics: conference series, IOP Publishing,* vol. 705, no. 1, p. 102,2016.

[119] E., Cela," The quadratic assignment problem: theory and algorithms." *Springer science and business media* vol. 1,2013.

[120] T., Xie, and Sun, Y.," A file assignment strategy independent of workload characteristic assumptions," *ACM transactions on storage (TOS),* vol.5, no. 3, pp. 10, 2009.

[121] S. M., Lim, and Leong, K. Y.," A Brief Survey on Intelligent Swarm-Based Algorithms for Solving Optimization Problems," *in nature-inspired methods for stochastic, robust and dynamic optimization. Intech open,* 2018.

[122] X., Meng, Liu, Y., Gao, X., and Zhang, H.," A new bio-inspired algorithm: chicken swarm optimization," *in international conference in swarm intelligence, Springer, Cham,* 2014.

[123] R., Grillo," Chicken Behavior: An Overview of Recent Science," *2014*.

[124] C.L., Smith, and Zielinski, S.L.," The Startling Intelligence of the Common Chicken," *scientific American* vol.310, no. 2, 2014.

[125] M. C., Mazilu," Database replication," *database systems journal,* vol.1, no. 2, pp. 33-38, 2010.

[126] B., Kemme, Jiménez-Peris, R., Patiño-Martínez, M., and Alonso, G. ," Database replication: A tutorial. In Replication," *Springer, berlin, heidelberg,* pp. 219-252, 2010.

[127] A., Thomson, and Abadi, D. J.," The case for determinism in database systems,"*proceedings of the VLDB endowment*, vol.3,no.1-2,pp. 70-80, 2010.

[128] F. R., Sousa, and Machado, J. C.," Towards elastic multi-tenant database replication with quality of service," *in proceedings of the 2012 IEEE/ACM fifth international conference on utility and cloud computing, IEEE computer* society, pp. 168-175, 2012.

[129] U. F., Minhas, Rajagopalan, S., Cully, B., Aboulnaga, A., Salem, K., and Warfield, A.," Remusdb: Transparent high availability for database systems," *the VLDB journal—the international journal on very large data bases,* vol.22, no. 1, pp. 29-45, 2013.

[130] R., Garcia, Rodrigues, R., and Preguiça, N.," Efficient middleware for byzantine fault tolerant database replication," in *proceedings of the sixth conference on computer systems, ACM,* pp. 107-122, 2011.

[131] S., Depaoli, and van de Schoot, R.," Improving transparency and replication

in Bayesian statistics: The WAMBS-Checklist," *psychological methods,* vol.22, no. 2, pp. 240, 2017.

[132] H., Chan, and Chieu, T.," An approach to high availability for cloud servers with snapshot mechanism," *in proceedings of the industrial track of the 13th ACM/IFIP/USENIX international middleware conference, ACM,* p. 6, 2012.

[133] R., Palmieri, Quaglia, F., Romano, P., and Carvalho, N.," Evaluating database-oriented replication schemes in software transactional memory systems," 2010.

[134] S. K., Yadav, Singh, G., and Yadav, D. S.," Mathematical framework for a novel database replication algorithm," *international journal of modern education and computer science,* vol.5, no.9, pp.1, 2013.

[135] G., Navarro, and Manic, M.," FuSnap: Fuzzy control of logical volume snapshot replication for disk arrays," *IEEE transactions on industrial electronics,* vol58, no.9, pp. 4436-4444, 2011.

[136] S., Sapate, and Ramteke, M.," Survey on comparative analysis of database replication techniques," *international journal of IT, engineering and applied sciences research (IJIEASR),* vol2, no.3, pp.72-80, 2013.

[137] Y., Wang, Hui, M., Liu, M., Dong, L., Liu, X., and Zhao, Y.," Edge extraction of optical subaperture based on differential box-counting dimension method with window merge replication,"*optical engineering,* vol.55,no.9, 2016.

[138] A. L. P. N., Alonso," Database replication for enterprise applications," 2017.

[139] E., Brewer," A certain freedom: thoughts on the CAP theorem," *in proceedings of the 29th ACM SIGACT-SIGOPS symposium on principles of distributed computing, ACM,* pp. 335-335,2010.

[140] S., Gilbert, and Lynch, N.," Perspectives on the CAP Theorem," *computer,* vol.45, no. 2, pp.30-36, 2012.

[141] B. W., Diack, Ndiaye, S., and Slimani, Y.," CAP Theorem between Claims and Misunderstandings: What is to be Sacrificed," *international journal of advanced science and technology,* vol.56, pp. 1-12, 2013.

[142] M., Kleppmann,” A Critique of the CAP Theorem,” *arXiv preprint arXiv*:1509.05393, 2015.

[143] O., Patinge, Karkhanis, V., and Barapatre, A.,” Inadequacies of CAP Theorem,” *international journal of computer applications,* vol.151no.10, 2016.

[144] N. K., Gill, and Singh, S.,” A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers,” *future generation computer systems,* vol.65, pp. 10-32, 2016.

[145] L., Wujuan, and Veeravalli, B.,” An object replication algorithm for real-time distributed databases,” *distributed and parallel databases,* vol.19, pp. 125-146, 2006.

[146] Z., Shamsa, and Dehghan, M.,” Placement of replicas in distributed systems using particle swarm optimization algorithm and its fuzzy generalization,” *in 2013 13th iranian conference on fuzzy systems (IFSC), IEEE,* pp. 1-6, 2013.

[147] B. K., Panigrahi, Shi, Y., and Lim, M. H. (Eds.).,” Handbook of swarm intelligence: concepts, principles and applications,” *Springer science and business media,* vol. 8 ,2011.

[148] K. N., Kaipa, and Ghose, D.,” Glowworm Swarm Optimization: Algorithm Development,” *in glowworm swarm optimization, Springer, cham,* pp. 21-56, 2017.

[149] U., Tosun,” Distributed database design using evolutionary algorithms.” *journal of communications and networks*, vol.16, no. 4, pp. 430-435, 2014.

PUBLICATIONS FROM THE THESIS

Contact Information: sadi_aluhabi@yahoo.com

Papers

1- S., H., THALIJ and V. HAKKOYMAZ,” Multiobjective Glowworm Swarm Optimization-Based Dynamic Replication Algorithm for Real-Time Distributed Databases”, *Hindawi, Scientific Programming*, Vol. 2018, Article ID 2724692,2018.

2- S., H., THALIJ and V. HAKKOYMAZ,” Dynamic Data Distribution for Merge Replication in Databases”, *IOSR Journal of Computer Engineering (IOSR-JCE)*, Vol. 19, Issue 6, Ver. I (Nov.- Dec. 2017), PP 41-46 ,2017.

3- S., H., THALIJ and V. HAKKOYMAZ,” A Case Study of Snapshot Replication and Transfer of Data in Distributed Databases”, *Tikrit University, Iraq, Tikrit Journal of Pure Science*, vol. 23, no. 10, p. 87-101, Jan. 2019. ISSN 2415-1726. ,2018.