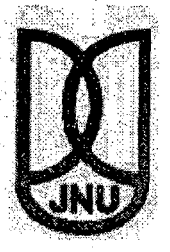
RELIABILITY
IN
DISTRIBUTED DATABASE MANEGMENT SYSTEM

A Dissertation submitted to the
School of Computer & Systems Sciences,
Jawaharlal Nehru University, New Delhi
In partial fulfillment of requirement for the award of the degree of

MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE AND TECHNOLOGY

BY
SURESH KUMAR

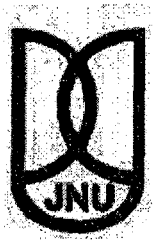
UNDER THE SUPERVISION OF
Prof P.C.SAXENA & Prof C.P.KATTI



SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI – 110067, INDIA
JULY 2009

Dedicated to

My Parents and Siblings



जवाहरलाल नेहरु विश्वविद्यालय

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES

JAWAHARLAL NEHRU UNIVERSITY

NEW DELHI – 110067, INDIA

CERTIFICATE

This is to certify that the dissertation entitled **"Reliability in Distributed Database Management System"** being submitted by Mr. SURESH KUMAR to **School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi** in partial fulfillment of requirements for the award of the degree of **Master of Technology in Computer Science and Technology,** is a record of bonafide work carried out by him under the supervision of **Prof. P.C.Saxena & Prof. C.P.Katti.**

This work has not been submitted in part or full to any university or institution for the award of any degree or diploma.

29.7.09 29.7.09.

Prof. P.C.Saxena & Prof. C.P.Katti

School of computer and system sciences

Jawaharlal Nehru University

New Delhi 110067

Dean

School of computer and system sciences

Jawaharlal Nehru University

New Delhi 110067

जवाहरलाल नेहरु विश्वविद्यालय

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES

JAWAHARLAL NEHRU UNIVERSITY

NEW DELHI – 110067, INDIA

DECLARATION

This is to certify that the dissertation entitled **"Reliability in Distributed Database Management System"** being submitted to the **School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi** in partial fulfillment of requirements for the award of the degree of **Master of Technology in Computer Science and Technology,** is a record of bonafide work carried out by me.

The matter embodied in the dissertation has not been submitted for the award of any degree or diploma in any university or institute.

Sureshkumar

Suresh Kumar

M.Tech (2007-2009)

School of computer and system sciences, JNU

New Delhi-110067

Acknowledgement

My first thanks is to the Almighty God, without his blessings I couldn't complete this dissertation.

I would like to express my heartfelt thanks to my supervisors **Prof. P.C.Saxena** and **Prof. C.P.Katti**, for giving me the guidance, encouragement, counsel throughout my dissertation. Without their valuable advice and assistance, it would not have been possible for me to complete this dissertation.

I feel it is a great privilege to have had the opportunity to work under their prestigious supervision.

I thank all the Faculties in School of Computer and Systems Sciences who have always been helpful in any of the career perspectives.

I am thankful to Dean of the school and administration of JNU for providing a congenial environment for making our work a success. Their academic support has been a real asset in completing this work.

Thanks to my class mates, friends and all others for their discussions and help at all time during the course of my dissertation work.

Finally, my greatest gratitude and thanks to my parent, siblings, who have provided endless emotional support and unending encouragement.

None of this would be possible without their help and assistance and "thanks" is too little a word to express my feeling.

Suresh Kumar

Abstract

Consistency among databases (distributed or even replicated) correctness, security, performance, etc. are the major issues leads to make a reliable distributed database system. Reliability is a probability which executes operations successfully in a given time period. Since a more available system is more reliable, replication of databases increases system availability hence reliability. Replica synchronization, replica update are the main problem in replicated environment. Due to failures, system reliability affected so system throughput and performance decreases our aim is to increase the system availability and decrease the communication cost of sites.

In this dissertation, we proposed a replication technique, HRT (Horizontal Replication Technique). We organized the sites in a two dimensional square matrix form called group structure and replicate data horizontally. We derive formulas to calculate the read/write availability and read/write communication cost by using quorum based replication. We are comparing HRT with DRG (Diagonal Replication on Grid) [7] which we used on group structure efficiently, the read/write availability increases for different probabilities and read communication cost of sites decreases while write cost increases.

The work presented in this dissertation is an outcome of calculation and simulation conducted by using MATLAB 7.0.1.

From the results of various calculation and simulations, we observed that the performance of HRT is better than the DRG in terms of system read/write availability, and communication cost.

Contents

Certificate---------- (i)

Declaration---------- (ii)

Acknowledgement---------- (iii)

Abstract---------- (iv)

Contents---------- (v)

List of Figures---------- (xi)

List of Tables---------- (xii)

Chapter 1

Introduction Distributed Database Management System

1.1 Distributed Processing---------- (1)

1.2 What are distributed in a distributed system? ---------- (1)

1.2.1 Processing Logic---------- (1)

1.2.2 Function---------- (1)

1.2.3 Data---------- (1)

1.2.4 Control---------- (1)

1.3 Distributed Database---------- (1)

1.4 Distributed Database Management System---------- (2)

1.5 Distributed Database Environment---------- (2)

1.6 Transparency---------- (3)

1.6.1 Data Independence---------- (3)

1.6.2 Distribution Transparency---------- (3)

1.6.2.1 Fragmentation Transparency---------- (3)

1.6.2.2 Location Transparency---------- (3)

1.6.2.3 Local Mapping Transparency---------- (4)

1.6.2.4 Naming Transparency---------- (4)

1.6.3 Transaction Transparency---------- (4)

1.6.4 Replication Transparency---------- (4)

1.7 advantages of distributed database management system---------- (4)

1.8 shortcomings of distributed database management system---------- (5)

1.9 Architecture of Distributed Database System---------- (5)

1.9.1 Autonomy---------- (5)

1.9.2 Requirement of Autonomy---------- (5)

1.9.3 Dimension of Autonomy---------- (6)

1.9.3.1 Design autonomy---------- (6)

1.9.3.2 Communication Autonomy---------- (6)

1.9.3.3 Execution Autonomy---------- (6)

1.9.4 Distribution---------- (6)

1.9.4.1 Client/Server Distribution---------- (6)

1.9.4.2 P-2-P Distribution---------- (6)

1.9.4.3 Heterogeneity---------- (7)

1.10 Client/Server Architecture---------- (7)

1.11 Peer-to-Peer Distributed system---------- (8)

1.12 Multi-database System Architecture---------- (9)

1.13 Design of Distributed Database System---------- (11)

1.13.1 The Interrelated issue of distributed database design---------- (12)

1.13.2 Fragmentation---------- (13)

1.13.2.1 Horizontal Fragmentation---------- (13)

1.13.2.2 Vertical Fragmentation---------- (15)

1.13.2.3 Mixed Fragmentation---------- (16)

1.13.3 Allocation---------- (16)

1.13.4 DATAID-D Methodology for Distributed Database Design---------- (17)

1.13.5 Analysis of Distribution requirement---------- (18)

1.13.6 Distribution Design---------- (18)

1.14 Transaction in Distributed Database System---------- (18)

1.15 Property of Transaction in Distributed Database System---------- (19)

1.15.1 Atomicity---------- (19)

1.15.2 Consistency---------- (19)

1.15.3 Isolation---------- (19)

1.15.4 Durability---------- (19)

1.16 Problem Definition---------- (19)

1.17 Motivation ---------- (20)

1.18 The Rest of The As Follows---------- (20)

Chapter 2

Reliability in distributed database management system

2.1 Why Reliability? ---------- (22)

2.2 Some Basic Concepts---------- (22)

2.2.1 System---------- (22)

2.2.2 State ---------- (23)

2.2.2.1 External state---------- (23)

2.2.2.2 Internal state---------- (23)

2.2.3 Specification---------- (23)

2.3 Failure---------- (23)

2.3.1 Transaction Failure---------- (23)

2.3.2 Media Failure---------- (24)

2.3.3 Communication Failure---------- (24)

2.3.4 Byzantine Failure---------- (24)

2.4 Reliability---------- (25)

2.5 Availability---------- (25)

2.6 Replication---------- (25)

2.7 Types of Replication---------- (26)

2.7.1 Partial Replication---------- (26)

2.7.2 Full Replication---------- (26)

2.8 Categorization of Replication Protocols---------- (26)

2.8.1 Synchronous---------- (26)

2.8.2 Asynchronous---------- (26)

2.8.3 Group---------- (27)

2.8.4 Master---------- (27)

2.9 Quorum Based---------- (27)

2.10 ROWA Replication Protocol---------- (28)

2.11 ROWA-A replication protocol---------- (28)

Chapter 3

Related work

3.1 Literature Survey---------- (30)

3.2 Database Site Models ---------- (30)

3.3 Replication Models---------- (31)

3.3.1All Objects to All Sites---------- (31)

3.3.2 All Objects to Some Sites ---------- (32)

3.3.3 Some Objects to All Sites---------- (32)

3.3.4 Some Objects to Some Sites---------- (33)

3.4 Communication Models---------- (34)

3.5 Data Access Models---------- (35)

3.5.1 Hot spot models---------- (35)

3.5.2 Locality Models---------- (36)

3.6 Transaction Processing Models---------- (36)

3.6.1 Transaction Models---------- (36)

3.6.2 Lock conflict models---------- (37)

3.7 System Models---------- (39)

3.8 DRG Technique---------- (40)

3.9 Definition of Diagonally Replication---------- (41)

3.10 Diagonal Set---------- (42)

3.11 Vote Assignment Function---------- (42)

3.12 Definition Quorum Group---------- (43)

3.13 Communication Cost and Availability---------- (44)

Chapter 4

Proposed Work Horizontal Replication Technique (HRT)

4.1 Model Assumption---------- (45)

4.2 Group Structure---------- (45)

4.3 Horizontal Replication Technique (HRT) in Group Structure---------- (45)

4.3.1 Group Structure for N = 81---------- (46)

4.4 Row Set---------- (47)

4.5 Communication cost in HRT---------- (48)

4.6 Availability in HRT---------- (49)

4.7 Simulation Results---------- (50)

4.7.1 Results of Read Availability---------- (50)

4.7.2 Results of Read Communication Cost---------- (51)

4.7.3 Results of write Availability---------- (52)

4.7.4 Results of write Communication Cost---------- (53)

Chapter 5

Conclusion---------- (55)

Reference---------- (56)

List of figures

Fig1.1: distributed database environment---------- (2)

Fig1.2: client/server architecture---------- (8)

Fig1.3: peer-to-peer distributed system architecture---------- (9)

Fig1.4: multi-database system model using a global conceptual schema

---------- (10)

Fig1.5: multi-database architecture without a GCS---------- (11)

Fig1.6: DDBMS design architecture DATAID-D---------- (17)

Fig 3.1: Classification of Replication Models---------- (31)

Fig 3.2 A DRG organization with fixed network---------- (41)

Fig4.1: Example of group structure with N=9×9 sites---------- (46)

Fig4.2: Read availability for n=9, j=3(initiates), q=5---------- (50)

Fig4.3: Read Communication Cost---------- (51)

Fig4.4: Write availability for n=9, i, j=3and q=5---------- (52)

Fig4.5: Write Communication Cost---------- (53)

List of Tables

Table1.1: Student---------- (14)

Table1.2: Horizontally fragmented---------- (15)

Table1.3: Vertically fragmented---------- (15)

Table1.4: mixed fragmented---------- (16)

Table4.1: Comparison between DRG and HRT read availability---------- (51)

Table4.2: Comparison between DRG and HRT read cost---------- (52)

Table4.3: Comparison between DRG and HRT write availability---------- (53)

Table4.4: Comparison between DRG and HRT write cost---------- (54)

Chapter 1

Introduction to Distributed Database Management System

There are two different and apposite approaches of data processing viz. database system and computer network, distributed database is the combination of these two approaches.

1.1 Distributed Processing

Distributed processing system states that it is a number of autonomous computing devices (homogeneous/heterogeneous) that are interconnected by a computer network and they cooperate in performing their assigned task [2].

1.2 What are distributed in a distributed system?

The four things which can be assumed to be distributed in distributed system [2].

1.2.1 Processing logic: Processing logic of computing devices is distributed.

1.2.2 Function: Various function of computing devices could be delegated to various pieces of hardware and software.

1.2.3 Data: Data those are used by number of application may be distributed to a number of computing devices.

1.2.4 Control: The control of execution of various tasks might be distributed.

1.3 Distributed Database

Distributed database is considered as a subset of Distributed Computing System.

A distributed database is a collection of multiple logically interrelated databases physically distributed on different sites over a computer network [2].

1.4 Distributed database management system

A distributed database management system is a collection of multiple logically interrelated databases and the software system that permits the management of the DDBS and makes the distribution transparent to the user [2].

In distributed database systems the data is stored at multiple sites that are geographically distributed over a possibly large area, a city, a country or even the whole world. For many distributed applications like banking, telecommunications, etc. distributed databases represent a more reliable and appropriate solution than centralize database system.

1.5 Distributed Database Environment

Two types of distributed database are considered.

- Homogeneous distributed database system in which participating sites are runs on same database management system.
- Heterogeneous distributed database system in which the participating sites not necessarily runs on same database management system.

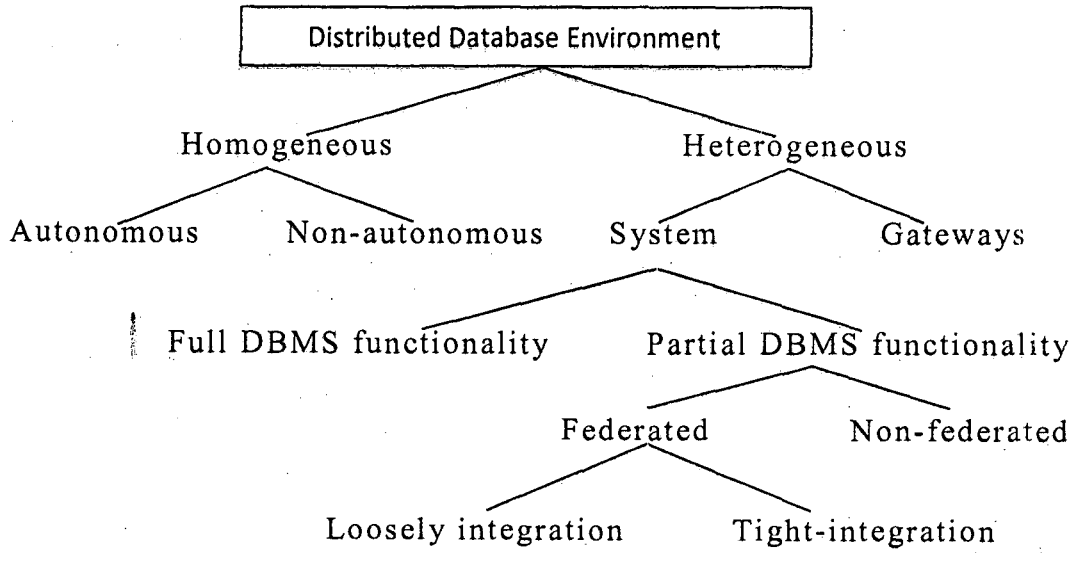


Fig1.1 Distributed database environment [1]

1.6 Transparency

Separation of higher level semantics of the system from lower level implementation issue is Transparency in Distributed Database System. Data independence is the fundamental transparency.

1.6.1 Data independence

It refers to the immunity of user applications to changes in the definition and the organization of data and vice versa. Two types of data independence are there.

- Logical data independence refers to the immunity of user application to changes in the logical structure of the database.
- Physical data independence deals with hiding the details of the storage structure from user application.

1.6.2 Distribution transparency

It treated physically dispersed database in DDBMS as single logical database the user don't need to know about the data are partitioned, data are replicated at several sites, data location.

1.6.2.1 Fragmentation transparency

The highest level of transparency, the end user or programmer does not need to know that a database is partitioned. Therefore neither fragment names nor fragment locations are specified prior to data access.

1.6.2.2 Location transparency

When the end user or programmer specifies the database fragment names but they didn't need to specify where these fragments are located.

1.6.2.3 Local mapping transparency

When the end user or programmer must specify the fragment names and where fragments are located.

1.6.2.4 Naming transparency

Each object in the database is given a unique name.

Distributed data dictionary (DDD) or distributed data catalog (DDC) supports distribution transparency.

1.6.3 Transaction transparency

Allows a transaction to access or update data at several network sites without knowing the location of sites.

- Ensures transactions maintain integrity and consistency
- Completed only if all involved database sites complete their of the transaction

1.6.4 Replication transparency

We don't bother where we accessing the main copy or the replica of the database from various sites in distributed database system.

1.7 Advantages of distributed database management system

In comparison of centralize database systems, DDBS is a better option for organizational point of view because data are distributed or even replicated over multiple sites. Every sites participating in DDBS provides local autonomy, i.e. they can execute and modify their data locally so user can access data locally with reduced cost and response time. Each site can access data from other remote sites via computer network. Due to data replication if one of the servers fails in a site it doesn't mean that DDBS is inaccessible.

one can access the data from another site. Replication provides the data protection and improves performance of DDBS. Transactions are more reliable, if centralize database system fails transaction operation can't be performed while in DDBS read or write operation can be performed on replicated data at another sites [2, 15].

1.8 Shortcomings of distributed database management system

Distributed database system is more complex (in designing, architecture, and modeling) and expensive due to maintain the extra overheads (hardware, software, communication network etc.) for reliability, in compare to centralize database system. Securing data at all the sites and integrity ate the major disadvantage of distributed database system [2, 15].

1.9 Architecture of distributed database system

Architecture of any system implies that component of the system are identified, functionality of each component is specified, and interrelationship and interaction among these components are specified. We will use a classification that organizes the system.

1.9.1 Autonomy

Individual system in DDBMS should be autonomous in terms of whether the component of system exchange the information, whether they can independently execute transaction and whether one is allowed to modify the transaction. Autonomy refers to the degree to which the system can operate independently and it talks about distribution of the control not about data.

1.9.2 Requirement of autonomy

- If individual DBMSs participate in the multiple database system for the local execution of operation are affected.
- The consistency of the system should not be compromise when individual DBMSs leave or join the multi database confederation.
- The manner in which individual DBMSs execute queries & optimize them should not be affected by the execution of global queries.

1.9.3 Dimension of autonomy

1.9.3.1 Design autonomy

On the preference basis individual distributed database system are free to use the data models and transaction technique.

1.9.3.2 Communication autonomy

Each of the individual distributed database system independently takes decision as to what type of information it wants to provide to the other distributed database systems.

1.9.3.3 Execution autonomy

Individual distributed database system can execute transaction independently.

1.9.4 Distribution

Distribution refers to the data distribution (physically) over multiple sites. There are many ways to distribute the data in DBMSs we consider only two of them.

1.9.4.1 Client/server distribution

In this distribution server provides the data management while client provides the application environment including the user interface. Client and server communicate via a high communication link.

1.9.4.2 Peer-to-peer distribution

This distribution system having no client server difference, each system has full DBMS functionality and can communicate with other system to execute queries and transaction.

1.9.4.3 Heterogeneity

In distributed system various forms of heterogeneity may occur like hardware heterogeneity, networking protocols may differ to variations in data managers. Representing data with different modeling tool creates heterogeneity because of the inherent expressive power and limitations of individual data models.

According to the classification autonomy, distribution and heterogeneity of system we will consider in detail the architecture of distributed database management system.

1.10 Client/server architecture

This provides a two-level architecture which makes it easier to manage the complexity of DBMSs and the complexity of the distribution. We can distinguish the functionality of Client/server architecture into two classes, server function and client function. Server performs most of the data management work like query processing and optimization, transaction management and storage management etc. Clients provides application and user interface and has a DBMS client module that is responsible for managing data that is cached to the client and sometimes managing transaction locks. The consistency checking of user query takes place at client side, it's not frequently happen since it requires the replication of the system catalog at client. The client sends SQL queries to the servers and server does most of the work (like query understanding, executing etc.) and returns result to the client. The client and server communicate via high communication link. Various client/server architecture exits like multiple-client-single-server and multiple client-multiple-server etc.

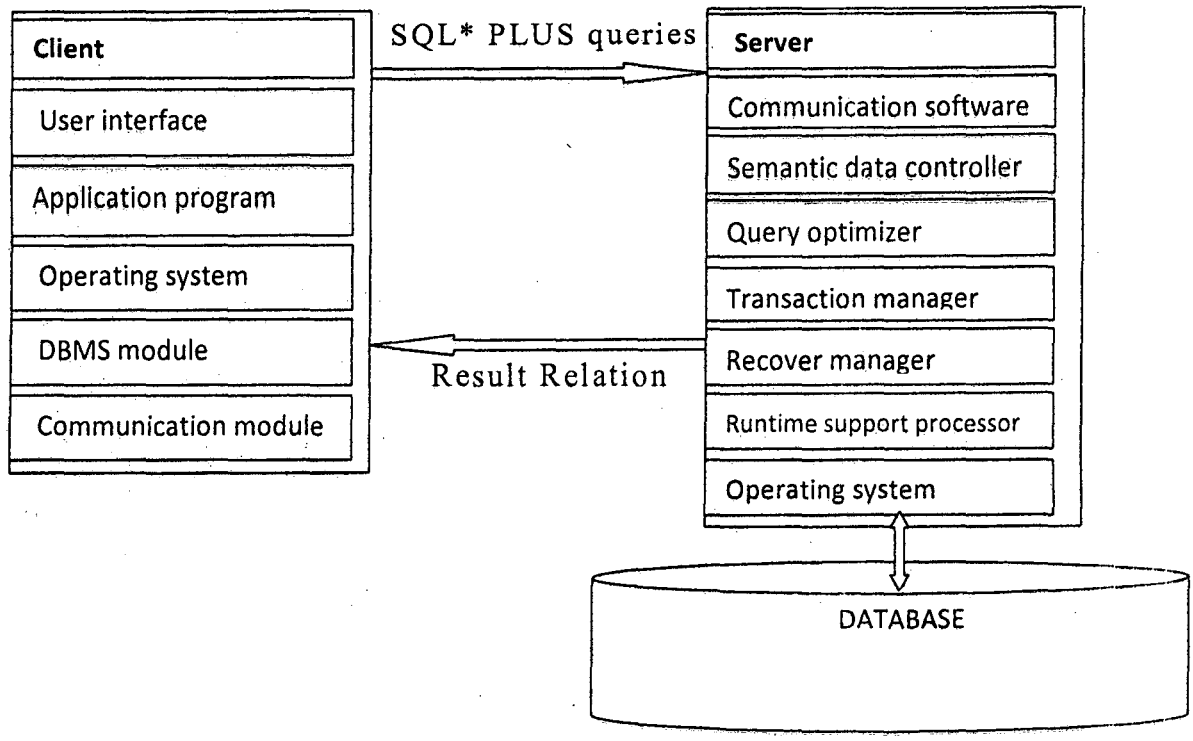


Fig1.2 client/server architecture [2]

1.11 Peer-to-peer distributed system

This architecture describes the data organizational view of system. On each peer (node) data which is physically distributed is probably different so to organize data on each node we need an internal schema definition called local conceptual schemas (LIS). The global conceptual schema (GCS) describes the logical structure of the data present at all the nodes. To handle the fragmented and replicated data in distributed database a distinct layer is present in this architecture, the local conceptual schemas (LCS). The global conceptual schema is the union of local conceptual schemas. The external schema (ESs) provides the user application and user access to the database.

Peer-to-peer distributed system architecture provides different level of transparency data independence, location and replication transparencies are provided by the LCS and GCS and mapping between them. GCS also provides network transparency.

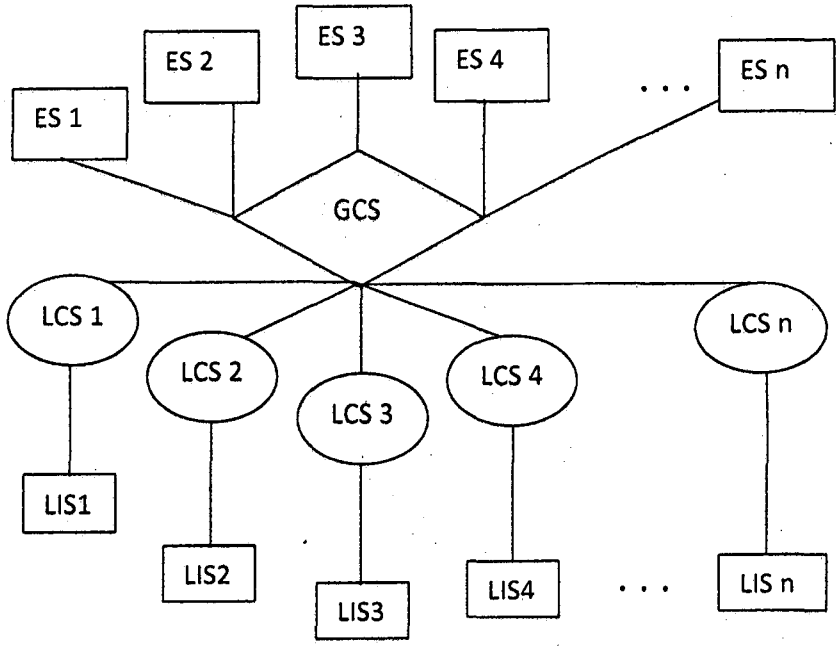


Fig1.3 peer-to-peer distributed system architecture [2]

1.12 Multi-database system architecture

Distributed database systems and multi-database systems both are different autonomous systems (i.e. they differ at autonomy level) and this difference reflects in their architecture. Basically the definition of global conceptual schema in both the databases is different. In former the global conceptual schema represents the conceptual view of entire database and is equal to the union of local databases while in the later global conceptual schema represents only the collection of some of the local databases that each local DBMS wants to share and is subset of the same union.

1.12.1 Multi-database system model using a global conceptual schema

In multi-database system the design the global conceptual schema includes the integration of either the local external schemas or the local conceptual schemas. An important difference between the design of the GCS in multi-databases and in DDBMSs is that in the former the association from is from local conceptual schemas to global schema and is a bottom-up design process while the association is in reverse direction in the later and is the top down design approach.

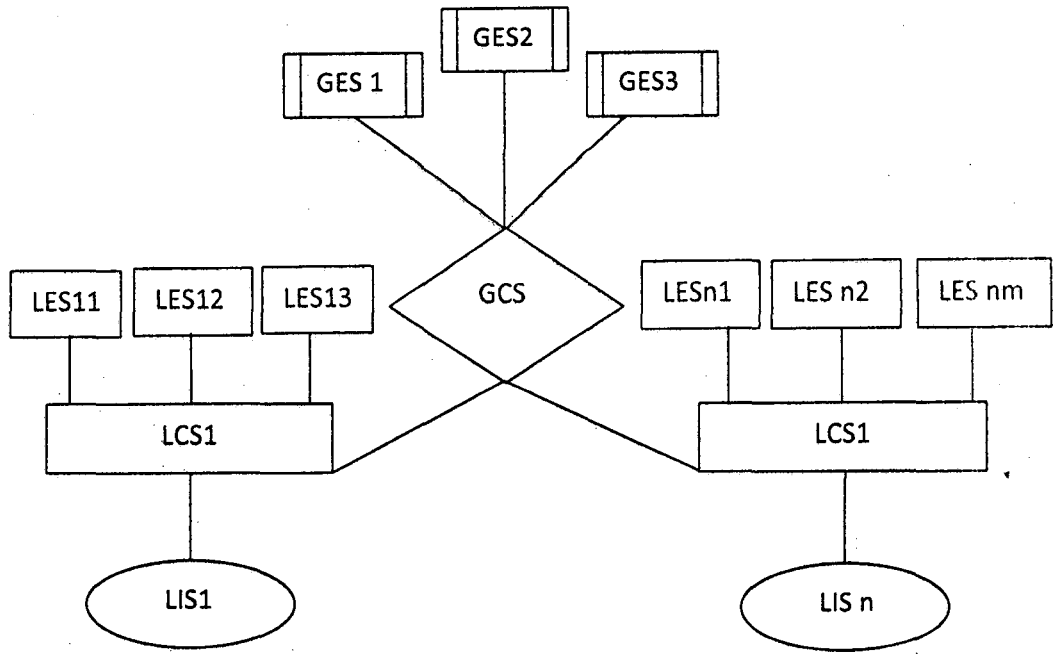


Fig1.4 multi-database system architecture using a global conceptual schema [2]

Once the global conceptual has been designed, views over the global schema may be defined for users who accesses global database.GCS and GES may be defined on different data models and languages. If the system is heterogeneous then two alternatives exist unilingual and multilingual.

In unilingual multi-DBMS on accessing a local database and the global database a user can use different data models and languages. In multilingual architecture the basic thing is that to permit each user to access the global database by means of an external schema, defined using the using the language of user's local DBM

1.12.2 Multi-database system model without using a global conceptual schema

This architecture identifies two layers structure, local system and multi-database layer. Local system layer consisting of multiple DBMSs which present to the multi-database layer the part of their local database they want to share with users of other databases. This shared data is presented either as the actual local conceptual schema

or as a local external schema definition, if heterogeneity exists each of these schemas, LCSi, may be use different data model.

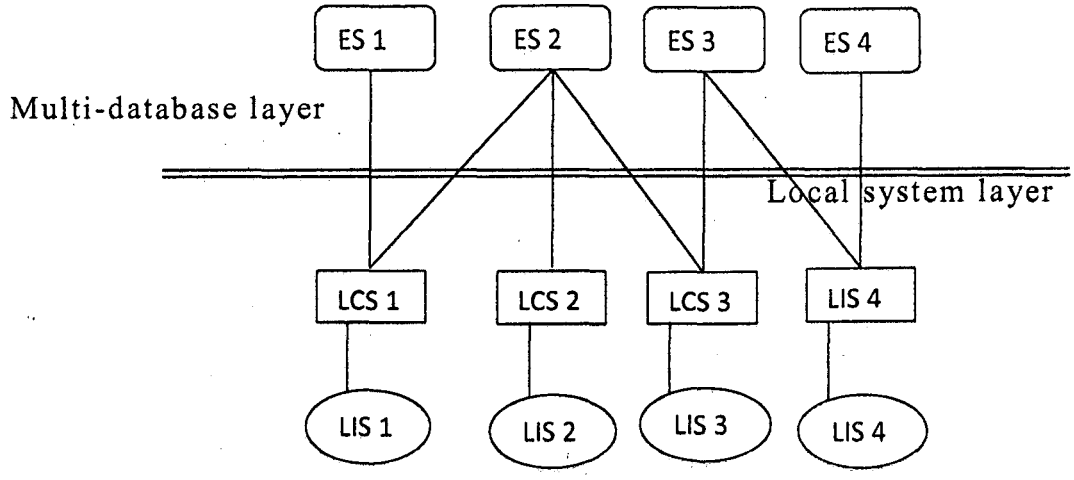


Fig1.5 multi-database architecture without a GCS [2]

External views are constructed over this layer where each view is defined on one LCS or multiple LCSs. Accessing multiple databases is the responsibility of the mapping between external schema and local conceptual schema.

1.13 Design of distributed database system

An important motivation in DDBMS design is to achieve maximum locality of databases and applications [10], sites are communicating with each other so allocating databases as close as possible to the applications which uses these databases, this reduce the communication cost. A well designed distributed database indicates that 90% of the databases should be access locally while only 10% databases should be access on remote site [10].

There are two different ways to design a distributed database viz. top-down and bottom-up approach the former is described as a distributed database developed from beginning while the later is typical of the development of a multi-database as the aggregation of existing database [2, 10].

1.13.1 The interrelated issues of distributed database design

- How a global relation should be fragmented
- How many fragmented copies are replicated
- How fragmented copies should be allocated to the sites of the computer network
- What are the necessary information for fragmentation and allocation [11]

The top-down design approach considers that the designer understands the requirements of a database application from the user, and incorporates them into formal specifications. During this process, the designer performs conceptual, logical, and physical design phases, which progressively refine high-level, system-independent specifications of the database into low-level, system-dependent specifications. During conceptual design, the designer is expected to ignore any detail concerning the physical implementation (in particular, data distribution). The result is a global database schema which incorporates, at an abstract level, all the data elements of the database and the patterns of their use. A design phase specific to distributed databases, called distribution design, maps the global schema to several, possibly overlapping subschema, each one representing the subset of information which is associated with one site. Then the design of each individual database is completed [2, 9, 11, 12].

The bottom-up approach assumes, instead, that a specification of the databases at each site exists already, either because there are existing databases that have to be interconnected to form a multi-database (or federated)system or because the conceptual specification of the databases has been done for each site independently. In either case, the site specifications have to be integrated in order to generate a global specification [2, 9, 11, 12].

Design of DDBMS includes an additional phase called distribution design other than four phases (requirement analysis, conceptual design, logical design, and physical design) described as in centralize database.

Distribution design, which assumes as input a global, site-independent schema and produces as a result the sub-schemas for each site of the distributed database. In principle, distribution design can be applied to any of the global conceptual, logical, or physical schemas. This choice is subject to the following tradeoff:

- Details about implementation should be decided only when the database distribution is given, to allow concentrating on the physical design of each local database independently.
  Note: Independent physical design is mandatory if the site DBMSs are heterogeneous.
- To estimating the performance of various distributions a precise description of operations and data is necessary.

Sub-schemas are also known as fragments of databases which are distributed over multiple sites. To determine the fragmentation and allocation distribution of databases is required.

1.13.2 Fragmentation

Fragmentation is a process to subdivide a global schema (relational table, entity) into several sub schema, these sub schemas are called fragments.

Fragment of data is the most appropriate unit of allocation, thus a good design should provide a way by which all the instances of fragments are uniformly accessed by means of transaction [2, 46].

There are two types of fragmentation viz. horizontal and vertical.

1.13.2.1Horizontal fragmentation

This partitions a global schema along its tupple.

CREATE TABLE STUDENT (S_NAME VARCHAR2 (24), S_ID NUMBER (6), NOT NULL, S_PHONE NUMBER, S_CITY CHAR (10), S_STATE VARCHAR2 (20));

Table1.1: Student

S_name	S_enroll	S_phone	S_city	S_state	S_id
Suresh	234123	9953333997	Kakri	UP	a132
Ajay	4333421	9953333531	Varanasi	UP	f365
Vikram	543425	9910345234	Bhopal	MP	g145
Vinay	456372	9993123456	Kuru	Haryana	n324
Renu	475389	9899567423	New Delhi	Delhi	z111

SELECT FROM STUDENT WHERE S_ID='a132' AND S_ID='Z111';

Table 1.2: Horizontally fragmented

S_name	S_enroll	S_phone	S_city	S_state	S_id
Suresh	234123	9953333997	Kakri	UP	a132
Renu	475389	9899567423	New Delhi	Delhi	z111

1.13.2.2 Vertical fragmentation

This partitions a global schema along its attributes or column.

SELECT S_NAME, S_ID FROM STUDENT;

Table1.2: Vertically fragmented

S_name	S_id
Suresh	a132
Ajay	f365
Vikram	g145
Vinay	n324
Surrender	v390
Renu	z111

1.13.2.3 Mixed fragmentation

The sub-schema which contains attributes and tupple both is known as mixed fragmentation.

SELECT S_NAME, S_ENROLL, S_PHONE, S_ID FROM STUDENT WHERE S_ENROLL='234123'AND'475389';

Table1.3: Mixed fragmented

S_name	S_enroll	S_phone	S_id
Suresh	234123	9953333997	a132
Renu	475389	9899567423	z111

1.13.3 Allocation

Allocation is the process to associating each fragment to one or more sites.

Fig 2.6 depicts that two additional phases are required in DATAID-D design methodology viz. analysis of distribution requirements and distribution design other than four design phases (requirement analysis, conceptual, logical, physical design) which is described as in DATAID-1 design architecture for centralize database.

1.13.4 DATAID-D methodology for distributed database design

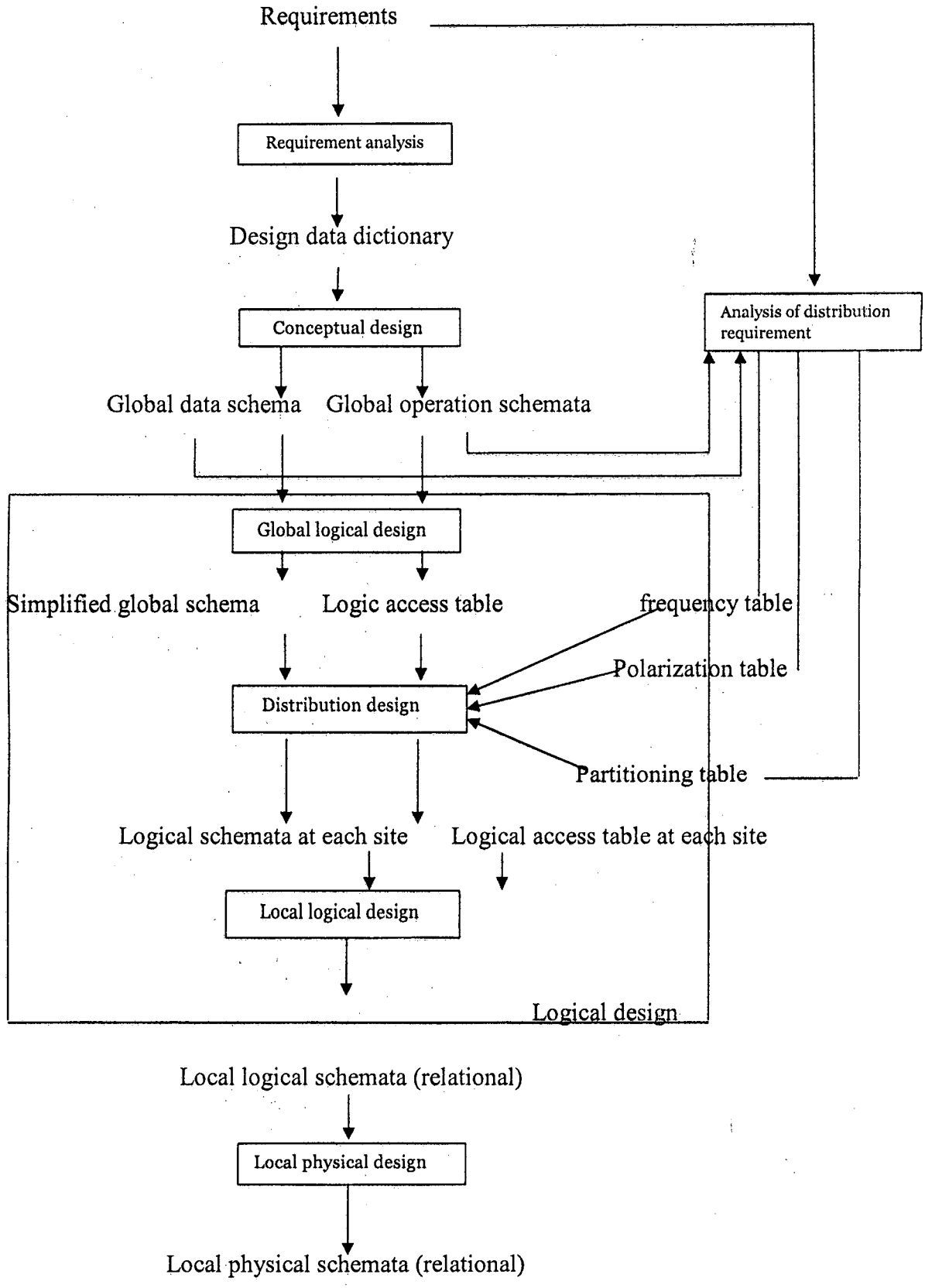


Fig 1.6 DDBMS design architecture DATAID-D [2, 9, 12]

1.13.5 Analysis of distribution requirements

The requirement of this phase is to gather information about distribution, such as partitioning predicates for horizontal fragmentations and the frequency of activation of each application from each site. Since the data structure and applications must be known in order to gather information about their distribution, requirements of distribution are gathered starting from few of the results of the conceptual design phase.

1.13.6 Distribution design

Distribution design phase starts from the specification of the global database schema and from the gathered distribution requirements, and produces many database schemata, one for each site of the distributed database, each one explaining the portion of data that will be allocated to that site.

1.14 Transactions in distributed database system

In DDBS transactions can be considered as a metrics for integrity, consistency and reliability. An important criteria or issue in transaction management is that if a database was in a consistent state earlier in time to the initiation of a transaction, then the database should return to a consistent state after the completion of transaction, irrespective of the fact that transactions were successfully executed simultaneously or they experience any kind of failure during the execution of operations (read/write) [2]

Definition

A transaction is consists of several read/write operations which starts with begin transaction, executes read/write operation and terminate with end transaction statements. Every transaction must be terminate either by executing successfully which we call commit or due to failure which we call abort transaction. DDBS has to follow the ACID property of transaction to ensure the consistency and reliability.

1.15 Properties of Transaction in Distributed Database System

1.15.1 Atomicity

If the transaction interrupted due to any type of failures, it requires atomicity. It refers that either execution of all the transactions operation (read/write) are completed or none of them.

1.15.2 Consistency

It refers to the correctness property of the transaction. Transaction maps one consistent database state to another.

1.15.3 Isolation

This property of transaction requires that each transaction assumes a consistent database all the time.

1.15.4 Durability

The changes made to the databases by transaction after completing successfully are permanent. Thus DDBS ensures the result of the transaction will survive on system failures.

1.16 Problem Definition

A reliable distributed database system must provide its user with correct and consistent data whenever and wherever they need them. Reliability is an important component of any computing system either centralize database or distributed database system. The reliability of a distributed database management system is the probability that an operation which executed on multiple computing devices (geographically dispersed) and needs to be communicated with each other for remote database will be executed successfully in a given time slot.

If the computer fails in centralize database system entire database is unavailable to its user until the computer is repaired while in distributed database system when a node fails only a portion of the database is made unavailable to its user.

In this dissertation we use the replication technique to achieve the reliability. Replication is the process of copying the data from one or more sites and to distribute them at several sites. Distribution and replication of the resources (e.g. data, objects, processes) are assumed to achieve the reliability in DDBS.

Distribution isolates the failures in the system while replication provides alternate resources available. Replication is one of the keys to reliability, performance of DDBS by ensuring system availability.

1.17 Motivation

Actually the motivation to build a distributed database system is to increase the reliability of the system by distributing the database on different sites to make system reliable from failures.

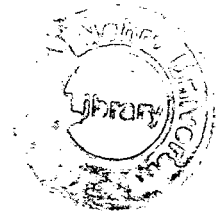
1.18 The rest of the dissertation as follows

Chapter 2 is organized according to the problem definition of this dissertation and solution approach, reliability, replication and replication protocols are discussed to maintain the consistency and integrity between databases.

Chapter 3 illustrates the related work.

Chapter 4 is concerned about the group structure system model in two dimensional matrix form, databases are replicated row wise which is defined in Horizontal Replication Technique (HRT).

Chapter 5 is the last chapter of this dissertation, in this chapter I have concluded my work regarding to reliability in distributed database management system by using replication technique to ensure the reliability of the system, and future work is discussed.



Chapter 2

Reliability in Distributed Database Management System

This is my dissertation problem in distributed database management system and I will use the replication of databases to achieve reliability.

2.1 Why reliability?

The performance of distributed database system depends on the reliability of the system; a system greatly decreases the response time, update, throughput of the transaction due to some kind of failures (transaction, media, communication link, etc.). Replication of databases increases the availability, hence reliability, if system is unreliable it must not be available for the database users. Correctness of the data is the most important criteria of the database reliability, by replicating databases we are increasing redundancy and redundancy leads to the inconsistency between databases so on accessing, data must not be correct. Reliability of DDBS deals with various types of complexity like fault tolerance, high and continuous availability, integrity, security, privacy, precise specification and implementation, time lines (system should be available in a given time period), fail-stop failures, network partitioning and communication failures etc.

Before discussing the reliability problem in DDBMS we will discuss few basic concept which affects reliability of the system.

2.2 Some Basic Concepts

2.2.1 System

System refers to a mechanism that consists of a collection of components and interacts with its environment by responding to stimuli from the environment with a recognizable pattern of behavior. [2].

2.2.2 State of the system

Two states are considered viz. external and internal state.

2.2.2.1 External state

This state is according to the response that a system gives to an external stimulus. It changes according to repeated stimuli from the environment.

2.2.2.2 Internal state

This state is the combination of the external states of the components that make up the system. It also changes according to stimuli from environment.

2.2.3 Specification of the system

It refers to the valid behavior of the system state, and is necessary for a successful system design and reliability of the system.

2.3 Failure

If system experiences any deviation from the system behavior described in the specification, is considered as a failure

2.3.1 Transaction failures

When a transaction fails, it aborts. Thereby, the database must be restored to the state it was in before the transaction started. Transactions may fail for several reasons. Some failures may be due to deadlock situations or concurrency control algorithms.

Site failures: Site failures are usually due to software or hardware failures. These failures result in the loss of the main memory contents. In distributed database, site failures are of two types:

- Total Failure where all the sites of a distributed system fail,
- Partial Failures where only some of the sites of a distributed system fail.

2.3.2 Media failures

Such failures refer to the failure of secondary storage devices. The failure itself may be due to head crashes, or controller failure. In these cases, the media failures result in the inaccessibility of part or the entire database stored on such secondary storage.

2.3.3 Communication failures

Communication failures, as the name implies, are failures in the communication system between two or more sites.

This will lead to network partitioning where each site, or several sites grouped together, operates independently. As such, messages from one site won't reach the other sites and will therefore be lost. The reliability protocols then utilize a timeout mechanism in order to detect undelivered messages. A message is undelivered if the sender doesn't receive an acknowledgment. The failure of a communication network to deliver messages is known as performance failure

2.3.4 Byzantine failure

The failure occurs in the system due to flood, earthquake, fire are considered as byzantine failures. In most of cases system can't be recovered.

2.4 Reliability

Reliability refers to the probability that the distributed database system does not experience any failure in a given time interval. [2]. Reliability of the system varies according to topology of the DDBS, communication link reliability etc.

Formally reliability is the capacity of a system that how it can tolerate and recover from failure, if R (t) denotes the reliability then is defined as the conditional probability

R (t) = pr {0 failures in time [0, t]: no failures occurs as the t=0}

2.5 Availability

It indicates the probability that the distributed database system is also operational according to the specification at a given point in time t. Steady state availability A of the system where failures follow a Poisson distribution with a failure rate β and that repair time is exponential with a mean repair time of $1/\alpha$ can be defined as [2]

$$A = \alpha / (\alpha+\beta)$$

2.6 Replication

Replication of databases at different sites in distributed database is an approach to achieve the reliability in distributed database management system.

Definition

Replication is a process of copying data (global relational table, global entity, etc.) or even fragments of data from one or more sites and to distribute them on several sites. The original data copy resides at all the site is termed as physical copy while the exact copy of the original data is known as replica.

2.7 Types of Replication

Replication is of two types viz.

2.7.1 Partial replication

Two aspects of this replication are considered, some databases are replicated at all the sites and some databases are replicated at some sites.

2.7.2 Full replication

All the replicas of databases are replicated at all the sites.

2.8 Categorization of Replication Protocols

For reliability and performance reasons, the system implements various replication protocols either synchronously or asynchronously.

2.8.1 Synchronous system

Synchronous system updates all the replicas before the transaction commits. Updates to all replicas are treated in the same way as any other data item. It produces globally serializable schedules. Synchronous strategy is also known as eager replication. [2, 3, 14]

2.8.2 Asynchronous systems

Asynchronous system updates only subset of the replicas. Other replicas are brought up-to-date lazily after the transaction commits. This operation can be triggered by the commit operation of the executing transaction or another

periodically executing transaction. Asynchronous strategy is known as lazy replication.

The replication strategies can be classified on the concept of primary copy. [13] are as follows.

2.8.3 Group

Any site having a replica of the data item can update it. This is also referred as update anywhere.

2.8.4 Master

This approach delegates a primary copy of the replica. All other replicas are used for read-only queries. If any transaction wants to update a data item, it must do so in the master or primary copy.

2.9 Quorum Based replication protocol

An interesting proposal to update only a subset of replicas and still not compromise with correctness and consistency is based on quorums.[4] Every copy of the replica is assigned a non-negative vote (quorum). Read and write threshold are defined for each data item. The sum of read and write threshold as well as twice of write threshold must be greater than the total vote assigned to the data. These two conditions ensure that there is always a non-null intersection between any two quorum sets. The non-null set between read quorum and write quorum guarantees to have at least one latest copy of the data item in any set of sites. This avoids the read/write and write/write conflict. All transactions must collect a read/write quorum to read/write any data item. A read/write quorum of a data is any set of copies of the data with a weight of at least read/write threshold.

Quorum-based protocols maintain the consistency of data in spite of operating only on a subset of the replicated database.

Let q be total number of votes (maximum quorum) = number of sites in the replicated system (assuming each site has equal weight), qr and qw be the read and write quorum respectively

To read or write a data item, a transaction has to collect a quorum of at least qr votes and a quorum of at least qw votes. The overlapping between read and write quorum makes sure that a reading transaction will at least get one up-to-date copy of the replica. The quorums must satisfy following two threshold constraints.

(i) $q_r + q_w > q$ and
(ii) $q_w + q_w > q$

Quorum-based replicated system may continue to operate even in the case of failures (sites or communication) if it is successful in obtaining the quorum for the data item.

2.10 ROWA replication protocol

The system knows which data items have replicas and where are they located. Read One Write All (ROWA) is one of the simple replica control protocol. In this protocol, if a transaction requests to read data items then system access the data (or replica) from the local sites. If a write operation is requested, the system must update all the replicas. Data accessed from local sites so that read operation is beneficial in replication, as it can find a replica near the site of request. But, write operations may adversely affect the performance of the system [3].

2.11 ROWA-Available replication protocol

ROWA-Available is an alternative of ROWA protocol for replica control. ROWA-A is more flexible than ROWA algorithm in presence of failures. The read request of transaction of ROWA-A are performed in similar fashion to ROWA, i.e. on any replica.

But write operations are performed only on the available copies and it ignores any failed replicas.

ROWA-A solves the availability problem, but the correctness of the data may have been compromised. After the failed site has recovered, it stores the stale value of the data. Any transaction reading that replica, reads an out-of-date copy of the replica and thus the resulting execution is not 1SR [3].

Chapter 3

Related Work

3.1 Literature Survey

This section discusses about the related work of different distributed database models according to the two aspects

1. Which aspects of the system are/aren't described in the model like uniform or non-uniform data access, replication, communication etc.
2. How these aspects are modeled?

3.2 Database site models

There are some models which uses queuing system. Some of the queuing models of distributed databases are described in [20], [21] and [22].

These model a fully replicated database of n local sites by an N/N/n/FCFS queuing system. Transactions which arrive according to a Poisson process are served on a first-come-first-serve basis by n servers and require an exponentially distributed service time. All the n servers are involved in processing the read transactions in parallel, and all the n servers are busy to process write transactions during their service time. These models uses shared read and exclusive write operations [20].

Writes have preemptive priority over read operations. This is modeled as an N/N/n system with preemptive service interruptions, where the interruptions correspond to the service periods of an N/N/n system which represents the arrival and service of updates [21].

The model described in [22] assumes non-preemptive processing of write operations and compare parallel updating with sequential updating of replicas.

The advantage of this site models is the read/write throughput of transaction is increased; data are accessed locally at reduced communication cost.

The drawbacks are inter-site communication is neglected, all sites share a single queue of incoming transactions, full replication is assumed which is considered an extreme case.

3.3 Replication models

Many distributed databases models simply assume no replication, i.e. each logical data item is represented by exactly one physical copy [24, 25]. Models which assume partial replication, either consider the fraction of replicated data (how many data are replicated?) or the degree of replication (to how many sites are data replicated?), but not both. This distinction leads to the following classification.

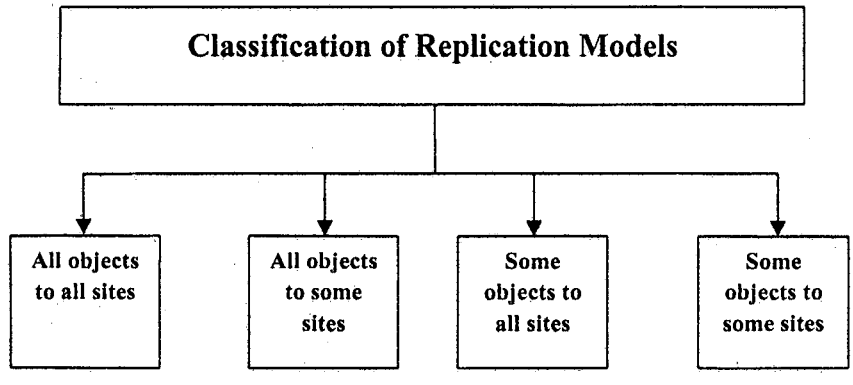


Fig3.1 Classification of Replication Models

3.3.1 All objects to all sites

i.e. the replicas of all data are stored at all sites therefore each site holds a complete copy of the distributed database [20].

The drawback, this is considered as an extreme case of replication and many applications uses neither full nor no replication [26]. Full replication can be considered as worst case [27].

3.3.2 All objects to some sites

All objects to some sites i.e. replicas of all data are stored at some sites [28]. Let r be the degree of replication, r ∈ {1, 2, ... , n}, implies that each logical data item is represented by r physical copies, where n is the number of sites. r = 1, indicates no replication whereas r = n implies full replication, and r > 1, means every data item is replicated. Consequently, either no or all data items are replicated.

Drawbacks are it is assumed that the replicas are distributed on sites, but which copies are placed on which sites is still undefined, such that different degrees of quality of a replication schema can be modeled. Data which is updated frequently should not be replicated to avoid update propagation overhead. However, data which is updated rarely but read frequently should be replicated to increase local availability and avoid communication delays. To select appropriate data items for replication, this can't be modeled with all data to some sites scheme.

3.3.3 Some objects to all sites

Some objects to all sites i.e. replica of some data are stored at all the sites, if r be the degree of replication, it can be defined as $r \in [0;1]$ implies that the fraction of logical data items that are fully replicated to all sites. r = 0 indicates no replication, r = 1 means full replication [29]. It is used for performance evaluation of relaxed coherency in partially replicated databases [30] and it can be modeled as a client server information system and assumed that 0% to 20% of the server data is cached at each workstation. This is comparable to the some data to all sites replication scheme with $r \in [0; 0.2]$ [30].

Some objects to all sites model is used to examine the correctness of a replication protocol based on group communication [31].

The some objects to all sites scheme is orthogonal to the all objects to some sites approach in the sense that the degree of replication is defined along the fraction of replicated data items as opposed to the number of copies.

Drawbacks are for r < 1, it is undefined which data item is selected for replication, and this undefined replica selection can be used to model the quality of replication. Full replication of some data items and no replication of others is a choice between two extremes and entails considerable update propagation overhead for the former and a severely reduced availability of the latter group of items. Since this situation is not typical in real-world applications, the some data to all sites scheme is again a questionable modeling approach.

3.3.4 Some objects to some sites

Some objects to some sites i.e. two-dimensional replication model. This model combines the two orthogonal some objects to all sites scheme and all objects to some sites. In this two-dimensional replication model, replication is modeled by a pair (r1, r2) ∈ [0;1] ×{2,...,n} such that r1∈ [0;1] describes the fraction of logical data items which are represented by r2 physical copies each, i.e. they are replicated to r2 of the n sites. A share of 1 - r1 logical data items is not replicated. r1 = 0 implies no replication while (r1=1, r2=n) models a full replication. For d logical data items, a replication schema (r1, r2) increases the number of physical copies from d (no replication) to (r1 × d × r2) + (d × (1 - r1)). Viewing the number of copies of replicated objects ($r1 \cdot d \times r2$) as the actual extent of replication, we express it independently from d and normalized to the interval [0; 1] as an overall level of replication. This is achieved through dividing by d × n, yielding (r1 × r2)/n. [32]

Due to high update propagation overhead it is hardly affordable to replicate some data items to all sites in large wide area distributed databases and reducing their availability. Thus, the some objects to all sites scheme is not realistic. Furthermore, in many applications there is update-intensive data which should be replicated to very few sites while read intensive data should be replicated to many sites. This cannot be modeled with the all objects to some sites scheme. The two dimensional approach can capture such scenarios and models realistic replication [32].

Drawbacks this model does not define the selection of data items to replicate nor their placement. This property of undefined replica selection and placement can be exploited to model the quality of replication. This scheme integrates the two

orthogonal some objects to all sites scheme is orthogonal to the all objects to some sites approach and has not yet been used in existing performance evaluations of replicated databases [32].

3.4 Communication Models

A lot of distributed database models consider that the transmission capacity of communication network is unlimited and transmission time is constant [33].

According to queuing theory network is implicitly modeled as an N/D/∞ system which is an infinite server that introduces a constant delay for each message, regardless of message size or network load [23]. Infinite means unlimited transmission capacity, no queuing of messages, and the network is never considered to be the bottleneck.

For simplicity, these details are usually omitted:

Some models relax the restriction of constant transmission delay but still presume unlimited network capacity and consider exponentially distributed communication delay by modeling the network as an N/N/∞ server [34, 28]. [36] Uses an N/G/∞ system to model arbitrarily distributed network delay.

Drawbacks these models only considers the response time as a performance metric. And predict that replication always deteriorates throughput but never increases it due to the infinite service capacity, situations in which the network starts getting congested cannot be captured.

However, many large wide area applications and wireless and mobile information systems suffer from low bandwidth. There, the communication links may indeed become a bottleneck, especially when a large amount of replicas is to be maintained. Unfortunately, very few attempts have been made to combine a detailed analytical database model with an analytical model of limited network capacity. Unfortunately, the database part of the model in [36] contains simplifying assumptions like full replication, uniform data access, and a workload of 100% updates (i.e. no read-only transactions). The capabilities of simulations to evaluate more complex system models have rarely been exploited to capture interdatabase communication details.

3.5 Data Access Models

A database model can either consider uniform data access or define a model of non-uniform data access. For mathematical observation, most model consider uniformly distributed data access, i.e. each data item is accessed with equal probability [33], Non-uniform data access is more realistic but used in very few models of distributed database systems [29]. These models of non-uniform data access are usually adopted from evaluations of centralized databases. They can be classified to be either hot-spot models or locality models.

In hot-spot models certain data groups (hot spots) are more likely to be accessed than others. In locality models local data is more likely to be accessed than remote data.

3.5.1 Hot-spot models

The classical hot-spot model of non-uniform data access for centralized database systems is b-c access. The model of b-c access describes that b % of the data requests are made to c % of the data items. A fraction c of the data items in the database are called regular granules and a lock request is with probability b for a regular granule. Among regular granules, each granule is accessed with equal probability, and the same is assumed for non-regular granules [37]. For many applications 0.99-0.01 access is a realistic model [38].

A generalization of the b-c access pattern to allow for more arbitrary distribution of non-uniform data access. They assume that the database D is divided into k classes of granules, D1, ... ,Dk, with [39]

$$D = \bigcup_{j=1}^{k} D_j$$

The probability of a request being made to a particular granule in class Dj is pj. The probability that a request is made to any granule in Dj is | Dj|.pj such that [39]

$$\sum_{j=1}^{k} | D_j | \cdot p_j = 1$$

For $k = 2$ this generalized model corresponds to the b-c access model in [37]. Such hot-spot models capture the skewness of the data access pattern which in turn has significant impact on the evaluation of lock conflicts.

3.5.2 Locality models

A data access model called b-l access describes that b % of the data request can be satisfied locally. This is different from b-c access model. If a transaction requests a local data, each local data is accessed with equal probability and the same holds for remote data respectively [40]. A value of b = 0.8 used in [40], while b = 0.5 used in [41] to express that 50% of the primary copies are accessed locally.

3.6 Transaction Processing Models

Replica control protocols assumed in performance evaluations of distributed databases include ROWA, primary copy with synchronous and asynchronous update propagation, as well as optimistic and quorum based algorithms [45]. Concurrency control protocols (distributed two-phase locking, optimistic methods, etc.) to capture lock conflicts and blocking of transactions are usually only modeled to compare concurrency control algorithms [26, 33]. Such models are of considerable complexity. They typically use simulations and simplified modeling assumptions concerning replication and communication.

3.6.1 Transaction Models

Since data accessing via transactions (queries) are easier to process for a database system than updates [33], a distinction in performance models is recommended unless a worst-case analysis is intended. Many models still consider updates only [40]. Several studies model two transaction types. This is sufficient for general

performance considerations while the evaluation of real-world applications desires the ability to model more detailed workload patterns [29].

For simplicity, most models do not consider distributed transaction processing. Some models assume a fully replicated database and that transactions can always be processed locally [33]. Others assume that due to skillful data fragmentation and allocation transactions can always be executed at a single site which is either the local or a remote site [30]. Distributed transaction processing is addressed in [42]. Under varying assumptions regarding the data access pattern, the number of data objects, and the number of database sites, these studies calculates the average number of data objects referenced per transaction and the average number of remote sites accessed per transaction as performance measures. Drawbacks are, these models of distributed transactions are not used to compute response times or transaction throughput. Performance studies that consider distributed transaction processing in the response time and throughput analysis are typically simulation studies. Two classes of transactions say class 1 transactions submitted at site k only access data locally available at site k. Class 2 transactions access local as well as well remote data items using distributed two-phase locking with primary copy or a distributed optimistic protocol [26].

3.6.2 Lock Conflict Models

Classical locking protocols (i.e. two-phase locking with blocking or abort-and-restart upon lock conflict) are the most familiar concurrency control methods for database systems. Lock conflicts and the resulting effects on transaction performance have been investigated extensively for centralized databases. Many results can be extended to distributed databases [24, 26].While some models assume dynamic locking (i.e. locks are not acquired before they are needed) [26], the performance of static locking in distributed database systems where all locks are obtained at the beginning of the transaction has been analyzed in [34]. Dynamic locking is more realistic, because a priori identification of all required locks is only possible at a very coarse granularity of locking or in special applications. The modeling concepts for concurrency control is given in [43] provide an introduction and various examples for performance modeling of distributed concurrency control.

Basically, the probability of lock conflicts depends proportionally on the average transaction arrival rate, the transaction size (i.e. number data objects accessed per transaction) and the lock holding time, and is inversely proportional to the total number of data items in the distributed database. The lock holding time depends on the delay of blocked transactions which in turn depends on the lock conflict probability. Hence, an iterative calculation is used in [26, 33] and the lock conflict probability also depends on the data access pattern.

The performance of locking under b-c access can be proved under three approximations (A1), (A2), (A3) in a database of size D is the same as that for uniform access in a database of size.

The three approximations are

(A1) The number of locks held by a single transaction is negligible compared to the total number of locks held.

(A2) The rate of aborted and restarted transactions due to deadlocks is negligible compared to the throughput.

(A3) The number of lock conflicts among three or more transactions is negligible compared to the number of lock conflicts that involve only two transactions.

These assumptions are generally accepted and also used for other purposes in analytical performance evaluations of databases. They can be justified by probabilistic considerations [37]

$$\frac{D}{1+(b-c)^2/c\cdot(1-c)}$$

This finding has been named the **database reduction approach** in [39] or **the effective database size paradigm (EDSP)** in [42].

Most models of distributed databases that capture lock conflicts consider updates and exclusive locks only [26, 33, 40]. With another type of EDSP it can be shown that the performance of a database of size D with shared and exclusive lock requests is equivalent to that of a database of size D/(1 – s2) with exclusive locks only, where s

denotes the fraction of lock requests that are in shared mode [37, 42]. Some models consider shared and exclusive locks and assume that lock conflicts are negligible and investigates only the overhead requesting and releasing locks [44]. [26] Distinguish between weak and strong locks, weak locks are requested during the execution of a transaction but can be preempted by strong locks in which case the preempted transaction is aborted. At the beginning of the two-phase commit of a transaction, weak locks are upgraded to strong locks. If any strong lock request is rejected (due to a conflict with another strong lock) the transaction is aborted.

3.7 System Model

A distributed system under fixed peer-to-peer network environment consists of a set of distinct sites that communicate with each other, and share both data and resources over a communication network. A site may become inaccessible due to site or partitioning failure. No assumptions are made regarding the speed or reliability of the network.

A distributed database system in peer-to-peer environment consists of a set of data objects stored at different sites in a computer network. Users interact with the database by means of transactions, which are partially ordered sequences of atomic read and write operations. The execution of a transaction must appear atomic: a transaction either commits or aborts [4].

In a replicated database, exact copies (replica) of a data object may be stored at several sites in the system. Multiple copies of a data object must appear as a single logical data object to the transactions. This is termed as one copy equivalence and is enforced by the replica control technique. The correctness criterion for replicated database is one-copy serializability [4], which ensures both one-copy equivalence and the serializable execution of transactions. In order to ensure one-copy serializability, a replicated data object may be read by reading a quorum of copies, and it may be written by writing a quorum of copies. The selection of a quorum is restricted by the quorum intersection property to ensure one-copy equivalence: For any two operations o[a] and o'[a] on a data object a, where at least one of them is a write, the quorum must have a non-empty intersection. The quorum for an operation is defined as a set of copies whose number is sufficient to execute that operation [5].

Briefly, a site i initiates a DRG transaction to update its data object. For all accessible data objects, a DRG transaction attempts to access a DRG quorum. If a DRG transaction gets a DRG write quorum without non-empty intersection, it is accepted for execution and completion, otherwise it is rejected. We assume for the read quorum, if two transactions attempt to read a common data object, read operations do not change the values of the data object. Since read and write quorums must intersect and any two DRG quorums must also intersect, then all transaction executions are one-copy serializable.

3.8 DRG Techniques

This technique is used on the fixed peer-to-peer network environment network. For the fixed network, all sites are logically organized in the form of two-dimensional grid structure.

For example, if a DRG consists of nine sites, it will logically organized in the form of 3 x 3 grid as shown in Fig 3.2. Every site containing a master data file. We consider that replicas are data files. A site is either available or failed and the state (available or failed) of each site is independent to the others. When a site is available, the copy at the site is available; otherwise it is unavailable. In the fixed network, the data file will replicate to diagonal sites. The logical structure for fixed network is shown as in Fig 3.2. The square in the grid represent the sites under the fixed network environment and a, b... and i represent the master data files located at site 1,2,...,and 9 respectively [5].

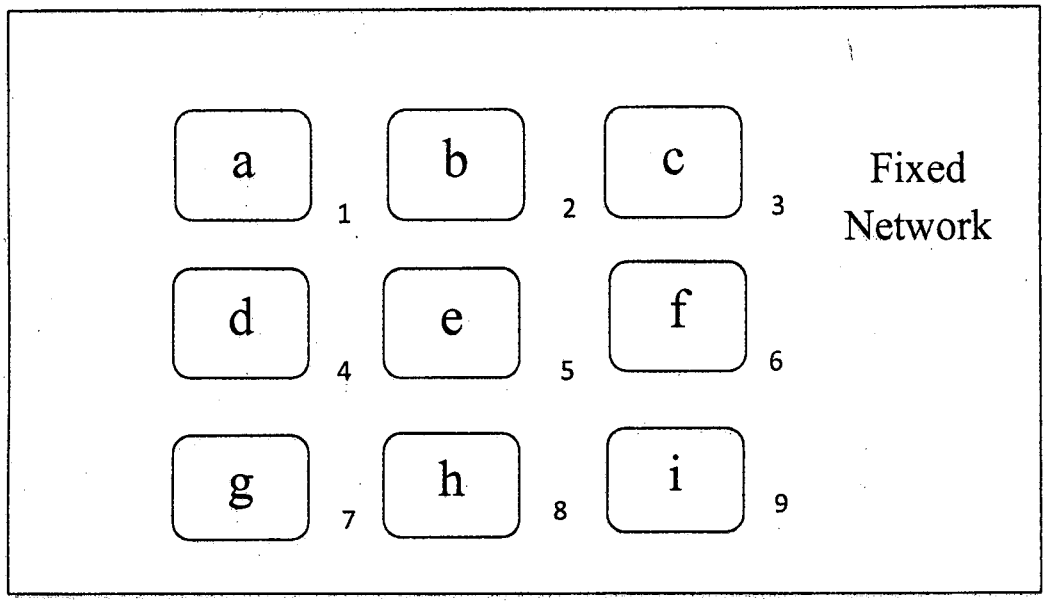


Fig 3.2 A DRG organization with fixed network [5]

3.9 Definition of Diagonally Replication

We consider that the fixed network environment consists of n x n sites that are logically organized in the form of two dimensional grid structure. All sites are labeled s (i, j), 1 ≤ i≤ n, 1 ≤ j ≤n. The diagonal site to s(i, j) is {s(k, l) k=i+1, l = j+1, and k ,l≤n, if i=n, initialized i = 0, if j=n, initialized j = 0}[5].

As an example, Assume that n =3, then the diagonal site to s(1,1) is s(2,2), the diagonal site to s(2,2) is s(3,3), the diagonal site to s(2,1) is s(3,2), and s(3, 2) is s(1, 3) etc.

The commonly visited site is defined as the most frequent site that requests the same data at the fixed network (the commonly visited sites can be given either by a user or selected automatically from a log file/ database at each center). This site will replicate the data asynchronously, therefore it will not be considered for the read and write quorums. Since the data file is replicate to only the diagonal sites, then it minimizes the number of database update operations, misrouted and dropped out calls. Also, sites are autonomous for processing different query or update operation, which consequently reduces the query response time.

3.10 Diagonal Set

A diagonal set, D(s), is a set of diagonal sites and the number of diagonal set equals to n, and the mth diagonal set is denoted by D^m (s), for m=1,2,...n.

For example, from Fig 3.2, n = 3, then the diagonal sets are;

$D^1(s) = \{s(1,1), s(2,2), s(3,3)\}$,

$D^2(s) = \{s(2,1), s(3,2), s(1,3)\}$,

$D^3(s) = \{s(3,1), s(1,2), s(2,3)\}$,

3.11 Vote Assignment Function

The primary site of any data file and for simplicity, its diagonal sites are assigned with vote one and vote zero otherwise, which is analogous to binary vote assignment [15]. A vote assignment V on grid, is a function such that $V(s(i,j)) \in \{0,1\}$, $1 \leq i \leq n, 1 \leq j \leq n$ where V(s(i,j)) is the vote assigned to site s(i,j). This assignment is treated as an allocation of replicated copies and a vote assigned to the site results in a copy allocated at the diagonal site. That is, 1 vote ≡ 1 copy.

$$\text{Let } L_V = \sum_{\substack{s(i,j) \in R(s) \\ 1 \leq i,j \leq n}} V(s(i,j)) \qquad \text{---------- (3.1)}$$

Where, L_V is the total number of votes assigned to the primary site and its diagonal sites. Thus, L_V= dr (dr represents the number of data replication from each site).

Let the read quorum and write quorum sets are denoted by qr and qw respectively. This protocol ensures that read operations always accesses currently updated copies, for this the following condition should hold.

$q_r + q_w$ must be greater than total number of votes assigned to replicated copies to all sites. The following condition ensures consistency among replicated copies.

$$1 \leq q_r \leq L_V,\ 1 \leq q_w \leq L_V$$

$$1.\ q_r + q_w = L_V + 1 \Rightarrow q_w = L_V - q_r + 1$$

These two conditions ensure that there is a non-empty intersection between read and write quorum. Hence read operation can access the current updated copy of replicated data

Let S(V) denote the set of sites at which replicated copies are stored corresponding to the assignment V, then

$$S(V) = \{s(i,j):V(s(i,j)) = 1,\ 1 \leq i,j \leq n\}$$

3.12 Definition (quorum group)

Let q be a quorum, any subset of S(V) whose size is greater than or equal to q is a quorum group. Quorum set is the collection of quorum group. Let Q(V, q) denote the quorum set with respect to the assignment V and quorum q, then

$$Q(V,q) = \{q_g : q_g \subseteq S(V) \ \&\ |q_g| \geq q\}$$

For example, from Fig 3.2, let site s(1,1) be the primary site of the master data file a. Its diagonal sites are s(2,2),s(3,3). Consider an assignment V for the data file a, such that

$$V_a(s(1,1)) = V_a(s(2,2)) = V_a(s(3,3)) = 1$$

$$\text{and } L_{Va} = 3.$$

$$\text{Therefore, } S(V_a) = \{ s(1,1), s(2,2), s(3,3)\}.$$

If a read quorum for data file *a*, r =2 and a write quorum $w = L_{Va} - r + 1 = 2$, then the quorum sets for read and write operations are $Q(V_a,2)$ and $Q(V_a,4)$, respectively.

3.13 Communication Costs and Availability

The quorum required to execute the operation, the communication cost of read/write operation is directly proportional to the size of quorum. To calculate the availability, all copies are assumed to have the same availability p. $C_{X, Y}$ denotes the communication cost with X technique for Y operation, which is R (read) or W (write).

Let p_i denote the availability of site i. Read/write operations on the replicated data are executed by acquiring a read/write quorum respectively. For simplicity, we choose the read quorum equals to the write quorum.

Thus, the communication cost for read and write operations equal to $\left\lceil \frac{L_{Va}}{2} \right\rceil$ that is, $C_{DRG,R} = C_{DRG,W} = \left\lceil \frac{L_{Va}}{2} \right\rceil$. For example, if the primary site has four neighbors, each of which has vote one, then CDRG,R = CDRG,W = $\left\lceil \frac{3}{2} \right\rceil = 2$. (3.2)

For any assignment V and quorum q for the data file a, define availability $\phi(V_a, q)$ to be the probability that at least q sites in $S(V_a)$ are available, then

$\phi(V_a\ q)$ = Pr{at least q sites in $S(V_a)$ are available}

Read availability for DRG

$$\phi(V_a, r) = \sum_{i=k}^{n} \left(\frac{n!}{i!(n-i)!} \right) (p^i)\ (1-p)^{n-i} \qquad \text{---------- (3.3)}$$

Writes availability for DRG

$$\phi(V_a\ w) = \sum_{i=n+1-k}^{n} \left(\frac{n!}{i!(n-i)!} \right) (p^i)\ (1-p)^{n-i} \qquad \text{---------- (3.4)}$$

Chapter 4

Proposed Work: Horizontal Replication Technique (HRT)

The work represented in this chapter is our proposed work which describes to calculate the system read/write availability, read/ write communication cost via quorum based replication protocol by replicating data horizontally.

4.1 Model Assumption

Let us consider a distributed database system which is consists of N independent, autonomous systems and are distributed in a peer-to-peer environment over a computer network. Sites are communicating with each other by passing messages. Sites are independent means that if a site fails, it can't affect the overall execution of the system and autonomous refers that they can execute their own data. Each site is assumed to have homogeneous database management system.

4.2 Group Structure

We consider a group structure of N sites in a two dimensional matrix form. If there are n2 sites in system then we can structure it in a two dimensional matrix form as $N=n\times n$, the connection between each site is per-to-peer.

4.3 Horizontal Replication Technique (HRT) in Group Structure

Let us consider that the peer-to-peer network environment is consists of $n\times n$ sites (globally dispersed), and are logically organized in group structure. The sites in this

structure are designated as s(i,j), $1 \leq i, j \leq n$. the horizontal site to s(i,j) is s(i,k) where k=j+1, $k \leq n$. if j = n then initialize j = 0.

For example if a system consists of N = 81, then it can be structured as N=n×n as N=9×9. The horizontally replicated sites to s(3,5) is s(3,6), to s(1,9) is s(1,1), and to s(9,9) is s(9,1) etc.

4.3.1 Group Structure for N=81

We organize a group structure for N=81 as N=9×9, in which squares represents the sites which are connected by peer-to-peer network (both way arrow represents the peer to peer communication between sites) and $x_1, x_2, - - - , x_{81}$ inside the boxes are the master data files. The one way arrow represents that further site contains the horizontally replicated copies of previous site.

s(i,k) ← replica of — s(i,j) where $k = j+1, 1 \leq i, j, k \leq 9$, if j = 9 put j = 0.

According to the above definition horizontally replicated sites to s(1,1) is s(1,2), to s(5,5) is s(5,6) etc. and the horizontal site to s(9,6) is s(9,7), to s(1,9) is s(1,1).

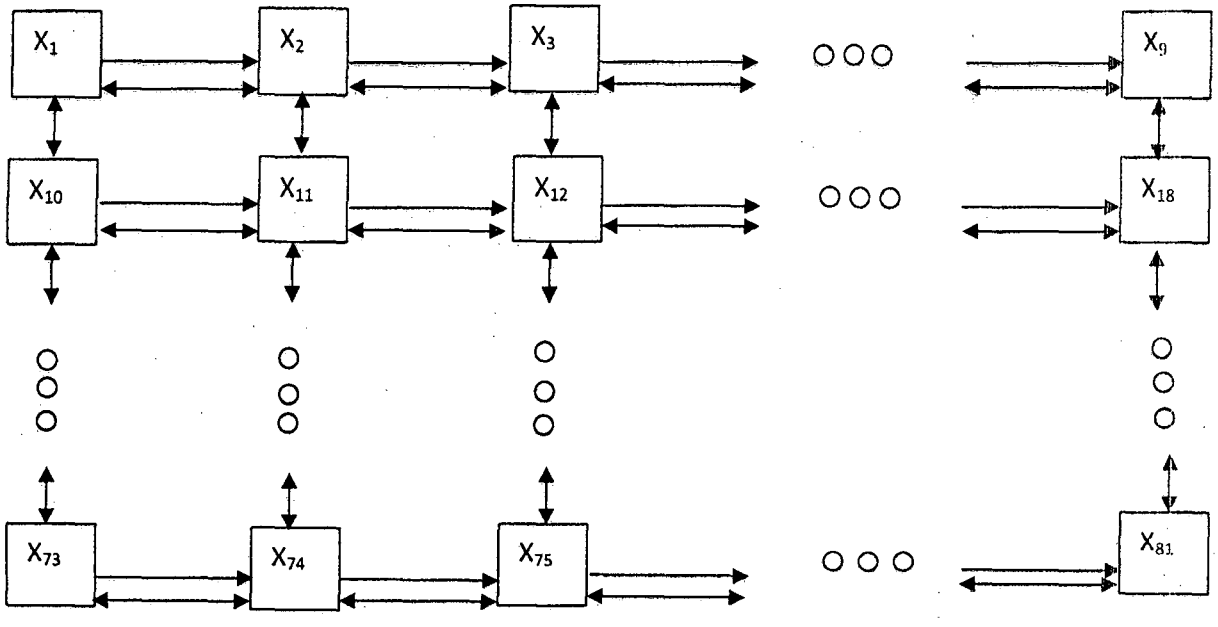


Fig4.1 Example of group structure with N=9×9 sites

4.4 Row Set

A set which is consists of horizontal sites is a row set and is denoted by R(s). Sites in the row set having common replica copies.

Sites are processing different read or write request by transaction resulting the reduced query response time, databases are replicated row wise then it reduces the write operation.

Let dr represents the number of data replication from each site.

The number of row sets is equal to n, and the m^{th} row set is denoted by $R^m(s)$ for m=1, 2, 3,..., n [5].

The different row sets are

$R^1(s) = \{s(1,1),s(1,2),s(1,3), \ldots , s(1,9)\}$

$R^2(s) = \{s(2,1),s(2,2),s(2,3), \ldots , s(2,9)\}$

$R^3(s) = \{s(3,1),s(3,2),s(3,3), \ldots , s(3,9)\}$

.....

.....

$R^9(s) = \{s(9,1),s(9,2),s(9,3), \ldots , s(9,9)\}$

$R^4(s)$ is one of the Row set and each site in this set will have the same replicas, $\{x_{28}, x_{29}, x_{30}, \ldots , x_{36}\}$.

We can assigns vote on primary site of any data file and its horizontal replicated sites with vote one or zero otherwise [8].

Let V be a voting assignment function on group structure and is defined as

$V(s(i,j)) \in \{0,1\}; 1 \leq i,j \leq n,$

Where V(s(i,j)) is the vote assigned to s(i,j) [5, 8].

We calculate total number of votes assigned to the primary sites and its horizontal replicated sites to use the quorum based replication protocol [8].

4.5 Communication Cost in HRT

On replicated data read and write operation of the transaction performed on read and write quorum respectively. The communication cost of read or write operation is depend on the size of the quorum.

Let $C_{HRT,R}$ and $C_{HRT,W}$ be the read and write communication cost in horizontal replication technique respectively. To calculate read/write communication cost DRG assumes equal read and write quorum sets [5], but in general it not happens. In HRT we consider read quorum 1/3 and write quorum 2/3 and take floor value. We compare read/write cost of HRT and DRG, in HRT the read cost is reduced and write cost increases according to the number of sites. L_{VX1} is the total number of votes assigned to master data file x_1. Since cost directly depends on the size of the quorum therefore read/write cost is given by equation (4.2) and (4.3) respectively.

$$C_{HRT,R} = \lfloor L_{VX1} / 3 \rfloor \quad \text{----------(4.1)}$$

$$C_{HRT,W} = \lfloor 2(L_{VX1}) / 3 \rfloor \quad \text{---------- (4.2)}$$

4.6 Availability in HRT

Let A(VX1,q) denote the availability, Where q is the quorum and V is a vote assignment function on data item x1 and defined as

A(VX1, q) = probability{at most q sites are available in $S(V_{X1})$}, where $S(V_{X1})$ is the collection of replicas of master file x1.

Availability in DRG is defined as the probability that at least q sites in $S(V_{X1})$ are available, we have changed this probability from at least to at most in HRT.

Let RA_{HRT} represents the read availability in HRT, p is the probability; the value of quorum q represents at most site availability. From the above definition the read availability will be

$$RA_{HRT} = \sum_{j=1}^{q} \left(\frac{q!}{j!(q-j)!} \right) (p)^j \ (1-p)^{q-j} \ , \ q \leq n \qquad \text{---------- (4.3)}$$

$q \leq n$ since we are considering that at most q sites are available.

Let WA_{HRT} represents the write availability in HRT, p is the probability; the value of quorum q represents at most site availability. From the above definition the write availability will be.

$$WA_{HRT} = \sum_{j=q-1}^{q} \left(\frac{q!}{j!(q-j)!} \right) p^j \ (1-p)^{q-j} \quad q \leq n \qquad \text{---------- (4.4)}$$

$q \leq n$ since we are considering that at most q sites are available.

4.7 Simulation Results

4.7.1 Results of Read Availability

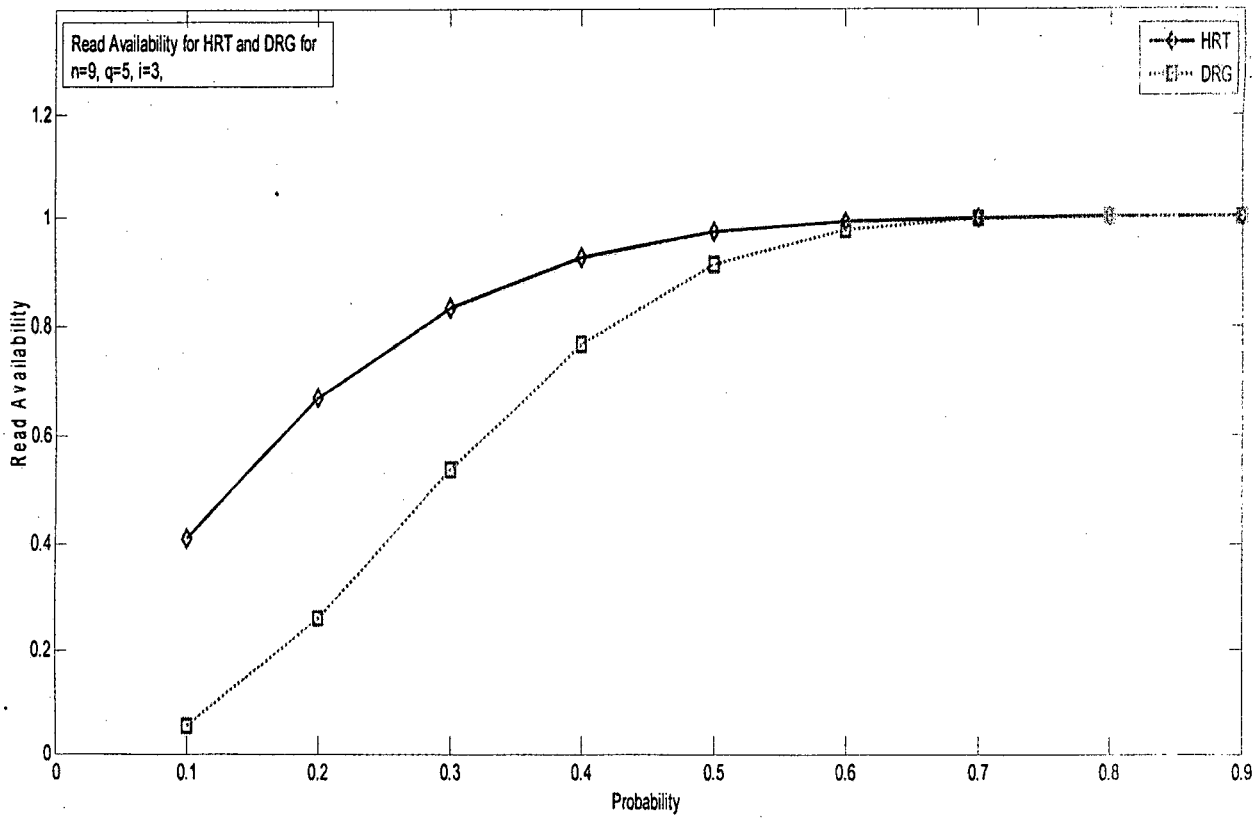


Fig4.2 Read availability for n=9, j=3(initiates), q=5

The read availability of HRT and DRG is calculated by equations (4.3) and (3.3) and result is depicted in fig4.2. This graph shows the read availability of HRT and DRG on group structure for different probabilities p. The read availability for probabilities 0.1, 0.2, 0.3, 0.4, is increased significantly, for probabilities 0.5, 0.6, 0.7 read availability in HRT also increases than DRG while constant at 0.8 and 0.9 for both the techniques, from table 4.1 we can observe that how availability of sites varies for different probability in HRT and DRG.

Table4.1: Comparison between DRG and HRT read availability

Probability	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
DRG	0.053	0.2618	0.5372	0.7682	0.9102	0.975	0.9957	0.9997	1
HRT	0.4095	0.6725	0.8319	0.9222	0.9688	0.9898	0.9976	0.9997	1
Comparison	Increases	Increases	Increases	Increases	Increases	Increases	Increases	Constant	Constant

4.7.2 Results of Read Communication Cost

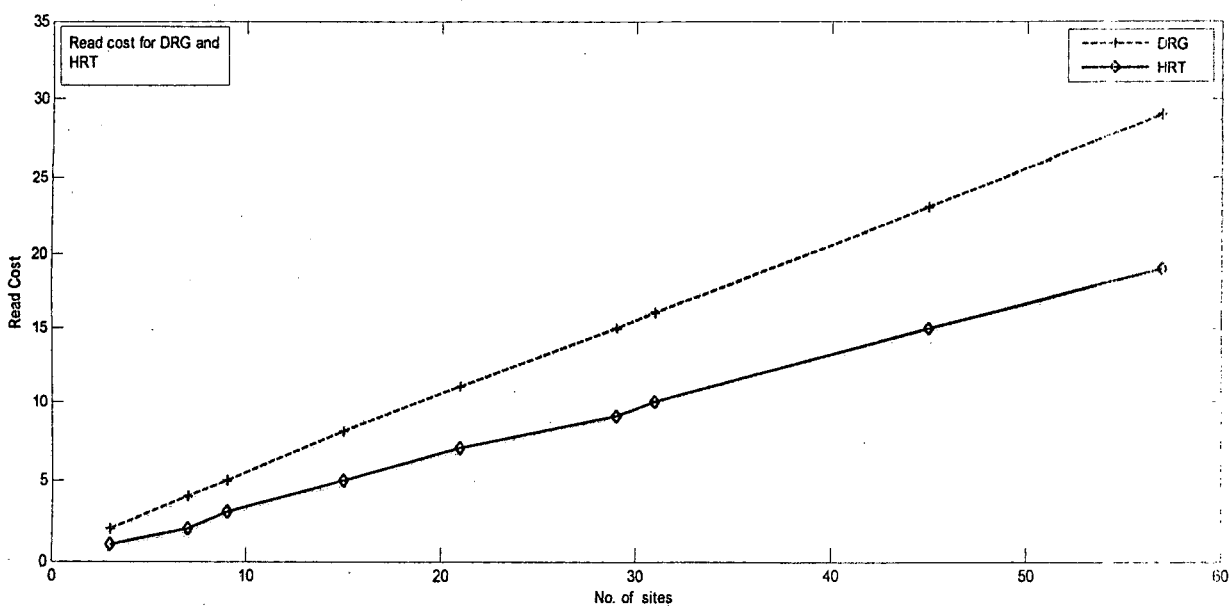


Fig4.3 Read communication cost

The read communication cost of HRT and DRG is calculated from equations (4.1) and (3.2), for different values of n say 3, 7, 9, 31 etc. and different number of sites say 9, 49, 81, 961 etc. and results are depicted in fig4.3. This graph is plotted between numbers of sites and read cost. The lower line shows read cost in HRT and is reduces in comparison with DRG (whose read cost is represented by upper line), from table4.2 we easily observe that for any number of sites the read cost of sites in HRT is reduced than the DRG in group structure. E.g. if group structure consists of N=441 sites then n equals 21, the read cost in DRG is 11 while in HRT is 7.

Table4.2: Comparison between DRG and HRT read cost

n\no. of sites	3\9	7\49	9\81	15\225	21\441	29\841	31\961	45\2025	57\3259
DRGread	2	4	5	8	11	15	16	23	16
HRTread	1	2	3	5	7	9	10	15	19
Comparison	Decreases	Decreases	Decreases	Decreases	Decreases	Decreases	Decreases	Decreases	Decreases

4.7.3 Results of Write Availability

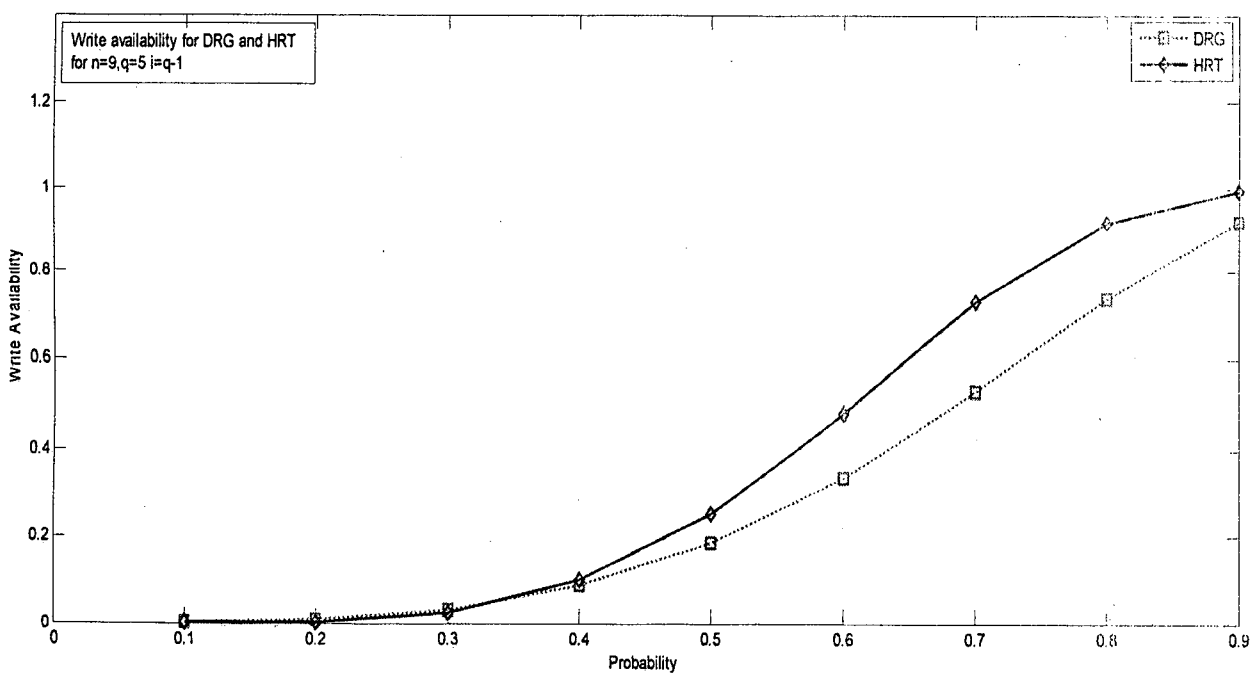


Fig4.4 Write availability for n=9, i,j=3and q=5

The write availability of HRT and DRG is calculated by equations (4.4) and (3.4) for n=9, i, j = 3 and q = 5 and result is depicted in fig4.4. This graph describes that write availability of sites in DRG and HRT on group structure. Write availability decreases for probabilities 0.1, 0.2, 0.3 in HRT than DRG and then increases from 0.3 to 0.9

significantly. How write availability varies according to different probabilities we easily observe from table 4.3.

Table4.3 Comparison between DRG and HRT write availability

Probability	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
DRG	0.0005	0.0067	0.0308	0.087	0.1875	0.337	0.5282	0.7373	0.9185
HRT	0.0001	0.0031	0.0253	0.0994	0.2539	0.4826	0.7297	0.9144	0.9917
Comparision	Decreases	Decreases	Decreases	Increases	Increases	Increases	Increases	Increases	Increases

4.7.4 Results of Write Communication Cost

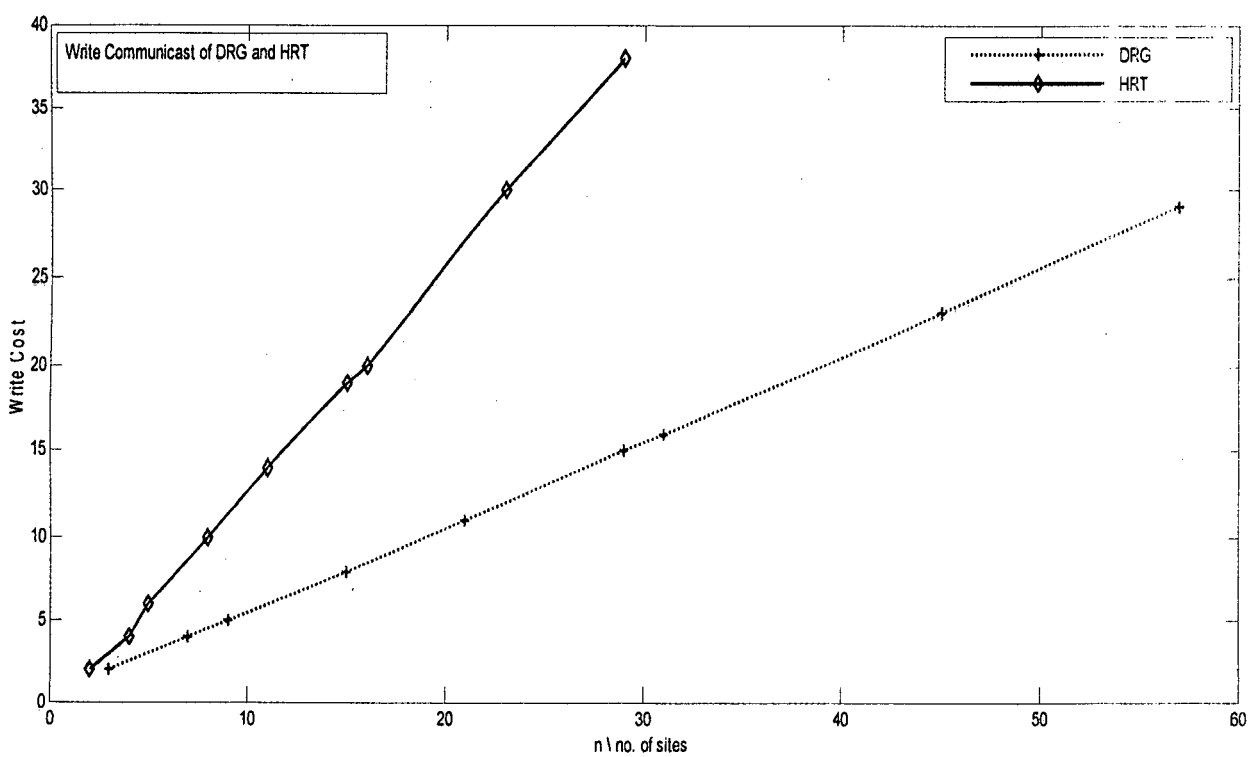


Fig4.5. Write Communication Cost

The write communication cost of HRT and DRG is calculated from equations (4.2) and (3.2), for different values of n say 3, 7, 9, 31 etc. and different number of sites say 9, 49,

81, 961 etc. and results are depicted in fig4.4. This graph is plotted between numbers of sites and write-communication cost. The upper line shows the write cost in HRT which increases than DRG (whose write cost is given by the below line). so write cost in HRT is increases in comparison with DRG. Table4.4 describes write cost of sites in DRG and HRT according number of sites.

Table4.4 Comparison between DRG and HRT write cost

n\no. of sites	3\9	7\49	9\81	15\225	21\441	29\841	31\961	45\2025	57\3249
DRG	2	4	5	8	11	15	16	23	29
HRT	2	4	6	10	14	19	20	30	38
Comparision	Constant	Constant	Increases	Increases	Increases	Increases	Increases	Increases	Increases

Chapter 5

Conclusion

Reliability of the DDBS doesn't come for free due to extra overheads like storing replica at several sites, replica synchronization, securing data at several sites, communication cost between replicas etc. Replication is an emerging and next generation technique in distributed database system, by replicating databases at several sites we provide alternate resources, so user can access or modify databases locally at reduced communication cost even in presence of failures.

Correctness and availability are the two aspects for DDBS reliability we consider only availability of the alternate resources which increases the reliability of the system.

In this dissertation, we proposed a replication technique, HRT (Horizontal Replication Technique) in group structure. In which we organized the sites in a two dimensional square matrix form (called group structure) and replicate data horizontally. We derive formulas to calculate the read/write availability and read/write communication cost by using quorum based replication. On comparing the results of HRT with results of DRG by the means of graph, and observe that the read availability increases for different probabilities p = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 and 0.7 while constant at p=0.8, 0.9 in HRT.

The write availability increases for different probabilities p = 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9 while decreases at p=0.1, 0.2, and 0.3.

Thus the overall read/write availability of sites increases by using HRT on group structure. According to number of sites the read communication cost of sites by using HRT is reduced on comparison with DRG while write cost increases.

We will use the horizontal replication technique on different DDBS models like heterogeneous, middleware, and try to minimize the write communication cost of transactions.

References

1. Bell, Grimson, "Distributed Database Environment" (2009 Pearson education Inc. Publication prentice hall chapter 14), (1982).

2. M. Tamer Ozsu, Patrick Valduriez, S. Sridhar, "Principles of Distributed Database Systems", second impression, (2007).

3. Sushant Goel, Raj Kumar Buyya, "Data Replication Strategy In Wide Area Distributed Systems".

4. Bernstein P. A. Hadzilacos V. and Goodman N, "Concurrency Control And Recovery In Database Systems", Massachusetts, Addision-Wesley Publishers (1987).

5. Mat deris, M, nathrah, B., suzuri, M.H., Abu Osman, M.T., "Improving Data Availability Using Hybrid Replication Technique In Peer-To-Peer Environments", AINA 1, 593–598 (2004).

6. Hector Garcia-Molina, Robert k. Abbot, "Reliable Distributed Database Management" proceedings of the IEEE, vol. 75, no. 5, may (1987).

7. Rohaya Latip, Hamidah Ibrahim, Mohamed Othman, Md Nasir Sulaiman, and Azizol AbdullahO. Gervasi and M. Gavrilova (Eds.), "Diagonal Data Replication In Grid Environment", ICCSA 2007, LNCS 4707, Part III, pp. 763–773, (2007).

8. M.Mat Deris, D.J. Evans, M.Y. Saman, A.Noraziah,"Binary Vote Assignment On A Grid For Efficient Access Of Replicated Data", International Journal of Computer Mathematics, vol.80, 2003(in press).

9. S .Ceri Ed., "Methodology and Tools for Database Design", Amsterdam, the Netherlands North Holland (1983).

10. J.B. Rothnic and N. Goodman, "A Survey Of Research And Development In A Distributed Database Management System", in proc. 3rd conference on very large scale database (1977).

11. Yin-Fu Huang and Jyh Her Chen, "Fragment Allocation In Distributed Database Design" (2001).

12. Stefan. Ceri, Barbara Pernici, And Gi Wiederhold, "Distributed Database Design Methodologies", proceedings of the IEEE, vol. 75, no. 5, may (1987).

13. Jim Gray, Pat Helland, Patrick O'Neil, Dennis Shasha ,"The Dangers Of Replication And A Solution", SIGMOD '96 6/96 Montreal, CanadaQ ACM (1996)

14. Yuri Breitbart, Raghavan Komondoor, Rajeev Rastogi, S. Seshadri, Avi Silberschatz, "Update Propagation Protocols For Replicated Databases", Bell Labs.

15. Ghazi Alkhatib, Ronny S. Labban, "Transaction Management in Distributed Database Systems: The Case of Oracle's Two-Phase Commit", Journal of Information Systems Education, Vol. 13(2)

16. Khalil N., Eid D., Khair M., "Availability And Reliability Issues In Distribute Databases Using Optimal Horizontal Fragmentation", DEXA'99, LNCS 1677, pp. 771-780, 1999. Springer-Verlag Berlin Heidelberg (1999)

17. F. Korth, Henry, Abraham. Silberschatz, S... Sudarshan "Database System Concepts" fourth Edition McGraw-Hill ISBN 0-07-295886-3 (2002)

18. Kenneth P. Birman, "Reliable Distributed Systems Technologies, Web Services" and Applications ISBN-13 978-0-387-21509-9 Springer New York, Heidelberg, Berlin_c 2005 Springer Science+Business Media, Inc.

19. Ghazi Alkhatib, Ronny S. Labban, "Transaction Management in Distributed Database Systems: the Case of Oracle's Two-Phase Commit" , Journal of Information Systems Education, Vol. 13(2)

20. E.G. Coffmann, Erol Gelenbe, Brigitte Plateau, "Optimization of the number of copies in a distributed system", IEEE Transactions on Software Engineering, Vol. 7, pp. 78-84, January 1981.

21. F. Bacelli, E.G. Coffmann, "A database replication analysis using an M/M/m queue with service interruptions", Performance Evaluation Review, Vol. 11, No. 4, pp. 102-107, 1983.

22. Randolph D. Nelson, Balakrishna R. Iyer, "Analysis of a Replicated Database", Performance Evaluation, Vol. 5, pp. 133-148, 1985

23. R. Jain, "The Art of Computer Systems Performance Analysis - Techniques for Experiment Design, Measurement, Simulation and Modeling", John Wiley & Sons, 1991.

24. D.M. Dias, P.S. Yu, B.T. Bennett, "On centralized versus geographically distributed database systems", 7th International Conference on Distributed Computing Systems, pp. 64-71, 1987.

25. S.H. Son, N. Haghighi, "Performance Evaluation of Multiversion Database Systems", Proceedings of the 6th International Conference on Data Engineering, pp. 129-136, February 1990.

26. B. Ciciani, D.M. Dias, P.S. Yu, "Analysis of Replication in Distributed Database Systems", IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 2, pp. 247-261, June 1990.

27. T. Anderson, Y. Breitbart, H. Korth, A. Wool, "Replication, Consistency, and Practicality: Are These Mutually Exclusive?" Proceedings of the ACM SIGMOD International Conference on Management of Data, June 1998.

28. Ravi Mukkamala, "Design of partially replicated distributed database systems", Technical Report TR-87-04, University of Iowa, Department of Computer Science, 1987.

29. R. Alonso, D. Barbará, H. Garcia-Molina, "Data Caching Issues in an Information Retrieval System", ACM Transactions on Database Systems, Vol. 15, No. 3, pp. 359-384, 1990.

30. Rainer Gallersdörfer, Matthias Nicola, "Improving Performance in Replicated Databasesthrough Relaxed Coherency", 21th Conference on Very Large Databases, pp. 445-456, September1995.

31. G. Alonso, "Partial Database Replication and Group Communication Primitives", Proceedings of the 2nd European Research Seminar on Advances in Distributed Systems (ERSADS'97), March 1997.

32. Matthias Nicola, Matthias Jarke, "Performance Modeling of Distributed and Replicated Databases" p n. 12

33. Hector Garcia-Molina, Gio Wiederhold, "Read-Only Transactions in a Distributed Database", ACM Transactions on Database Systems, Vol. 7, No. 2, June 1982, pp 209-234.

34. A.C. Shyu, V.O.K. Li, "Performance Analysis of Static Locking in Distributed Database Systems", IEEE Transactions on Computers, Vol. 39, No. 6, pp. 741-751, June 1990.

35. J.F. Ren, Y. Takahashi, T. Hasegawa, "Analysis of impact of network delay on multiversion timestamp algorithms in DDBS", Performance Evaluation, pp. 21-50, July 1996.

36. A.P. Sheth, A. Singhal, M.T. Liu, "An Analysis of the Effect of Network Parameters on the Performance of Distributed Database Systems", IEEE Transactions on Software Engineering, Vol.11, No. 10, pp. 1174-1184, October 1985.

37. Y.C. Tay, N. Goodman, R. Suri, "Locking Performance in Centralized Databases", ACM Transactions on Database Systems, Vol. 10, No. 4, pp. 415-462, December 1985.

38. Jim Gray, Andreas Reuter, "Transaction Processing: Concepts and Techniques", Morgan Kaufmann, 1993.

39. B. Zhang, M. Hsu, "Modeling Performance Impact of Hot Spots", In Vijay Kumar (Ed.), Performance of Concurrency Control Mechanisms in Centralized Database Systems, Prentice Hall, pp. 148-165, 1996.

40. A. Raghuram, T.W. Morgan, B. Rajaraman, Y. Ronen, "Approximation for the mean value performance of locking algorithms for distributed database systems", Annals of Operations Research, Vol. 36, No. 1-4, pp. 299-346, May 1992.

41. S.Y. Hwang, K.S. Lee, Y.H. Chin, "Data Replication in a Distributed System: A Performance Study", 7th International Conference on Database and Expert Systems Applications, pp. 708-717, 1996.

42. Alexander Thomasian, "Determining the Number of Remote Sites Accessed in Distributed Transaction Processing", IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 1, pp. 99-103, January 1993.

43. Wojciech Cellary, Erol Gelenbe, Tadeusz Morzy, "Concurrency Control in Distributed Database Systems", Elsevier Science Publishers, Holland 1988.

44. Eike Born, "Analytical performance modeling of lock management in distributed systems", Distributed Systems Engineering, Vol. 3, No. 1, pp. 68-76, March 1996.

45. A.A. Helal, A.A. Heddaya, B.B. Bhargava, "Replication Techniques in Distributed Systems", Kluwer Academic Publishers, 1996.

46. Benjamin w., Yao-nan Lien, "Design of Distributed Databases on Local Computer Systems with a Multiaccess Network", IEEE transactions on software engineering, vol. se-il, no. 7 pp 606, july 1985