ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 10, October 2013

An Overview of Transparency in Homogeneous Distributed Database System

Nitesh Kumar, Saurabh Bilgaiyan , Santwana Sagnika

***Abstract*— a transparency is some aspect of the distributed database system that is hidden from the user (programmer, system developer, and application program). A transparency is provided by including some set of mechanisms in the distributed system at a layer interface where the transparency is required. A number of basic transparencies have been defined for a distributed system. This paper present an overview of different types of transparency such as (Access, Concurrency, Location, Replication, Failure, Migration, Performance, Scaling, Execution, Configuration, Network, and Naming), and three type of levels transparency which is (Fragmentation, Location, and Local mapping). It's important to realize that not all of these are appropriate for every system, or are available at the same level of interface. In this paper, if all transparency have an associated cost with others, matter for much research how the cost of implementing multiple transparencies interact and how to reduce operating system and communication stack. It considered that if using such type of approaches such as: Application Layer Framing and Integrated Layer Processing**

***Index Terms*— Database transparency, database replication, homogeneous database, distributed database system, distributed processing.**

I. INTRODUCTION

Database replication based on group communication systems has been proposed as an efficient and flexible for data replication. Replication is the key characteristic in improving the availability of data distributed systems. Replicated data is stored at multiple server sites so that it can be accessed by the user even when some of the copies are not available due to server/site failures [1]. A major restriction to using replication is that replicated copies must behave like a single copy. i.e., mutual consistency as well internal consistency must be preserved [2]. Copies of the data items can be stored at multiple sites. The potential of data replication for high data availability and improved read performance is crucial to RTDBS. Data replication introduces its own problems. Access to a data item is no longer control exclusively by a single sites; instead, the access control is distributed across the sites each storing the copy of the data item. It is necessary to ensure that mutual consistency of the replicated data is provided.

***Manuscript received Oct, 2013*.**

***Nitesh Kumar,** is currently pursuing M.Tech, degree program in School of Computer engineering in KIIT University, India,*

***Saurabh Bilgaiyan ,** is currently pursuing M.Tech, degree program in School of Computer engineering in KIIT University, India,*

***Santwana Sagnika ,** is currently pursuing M.Tech, degree program in School of Computer engineering in KIIT University, India,*

II. DISTRIBUTED DATABASE SYSTEM (DDBS)

Distributed DBMS can also be integrated as a multiple process, single data n/w called MPSD, to allow more than one computer to access a single database. Large corporations may require an enterprise database to support May users over multiple departments. This would require the implementation of a multiple process, multiple data scenario, or MPMD, in which many computers are linked to a fully distributed client/server DDBMS. The DDBMS offer more reliability by decreasing the risk of a single site failure. If one computer in the n/w fails, the workload is distributed to the rest of the computers. Furthermore, a DDBMS allows replication of data among multiple sites, data from the failed site may still be available at one sites. In a centralized database can be implemented as a single process, single data scenario or SPSD , in which one computer is linked to the host DBMS to retrieve data .A centralized DBMS different because a failed computer that houses the database will debilitate the entire system. A distributed database is a database in which storage devices are not all attached to a common processing unit such as the CPU [3], controlled by a distributed database management system(together sometimes called a distributed database system). It may be stored in multiple computers, located in the same physical location; or may be dispersed over an interconnected computers. Unlike parallel systems, in which the processors are tightly coupled and constitute a single database system, a distributed database system consists of loosely-coupled sites that share no physical components. System administrators can distribute collections of data (e.g. in a database) across multiple physical locations. A distributed database can reside on network servers on the Internet, on corporate intranets or extranets, or on other company networks. Because they store data across multiple computers, distributed databases can improve performance at end-user worksites by allowing transactions to be processed on many machines, instead of being limited to one [4].

Two processes ensure that the distributed databases remain up-to-date and current: replication and duplication.

1. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The

replication process can be complex and time-consuming depending on the size and number of the distributed databases. This process can also require a lot of time and computer resources.

2. Duplication, on the other hand, has less complexity. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, users may change only the master database. This ensures that local data will not be overwritten.

Both replication and duplication can keep the data current in all distributive locations [4]. Besides distributed database replication and fragmentation, there are many other distributed database design technologies. For example, local autonomy, synchronous and asynchronous distributed database technologies. These technologies' implementation can and does depend on the needs of the business and the sensitivity/confidentiality of the data stored in the database, and hence the price the business is willing to spend on ensuring data security, consistency and integrity. Distributed DBMS can choose to have multiple copies of relations at different sites or choose to have only one copy of a relation [5]. The benefits of data replication is increased reliability − if one site fails, then other sites can perform queries for the relation. The performance will increase, as transaction can perform queries from a local site and not worry about network problems. The problem with data replication is decreased performance when there are a large number of updates, as distributed DBMS have to ensure that each transaction is consistent with every replicated data. This adds additional communication costs to ensure that all copies of the data are updated at the same time. Inside the growing world of distributed databases, data replication acting an increasingly important role. Replication intends to increase data availability in the presence of site or communication failures and to decrease retrieval costs by local access if possible. The maintenance of replicated data is therefore closely related to inter site communication and replication management can have significant impact on the overall system performance. Replication management in distributed database systems concerns the decision when and where to allocate physical copies of logical data fragments (replica placement) and when and how to update them to maintain an acceptable degree of mutual consistency (replica control). The literature offers various algorithms for replica placement [6, 7, 8] as well as replica control [9, 10, 11].

A. *Advantages of Distributed database system (DDBS)*

- ✓ Increase reliability and availability.
- ✓ Reliable transactions - due to replication of the database
- ✓ Hardware, operating-system, network, fragmentation, DBMS, replication and location independence
- ✓ Distributed query processing can improve performance
- ✓ Distributed transaction management
- ✓ Single-site failure does not affect performance of system.

B. *Disadvantages of Distributed DBS*

- ✓ Complexity--DBAs may have to do extra work to ensure that the distributed nature of the system is transparent. Extra work must also be done to maintain multiple disparate systems, instead of one big one. Extra database design work must also be done to account for the disconnected nature of the database--for example, joins become prohibitively expensive when performed across multiple systems.
- ✓ Economics — increased complexity and a more extensive infrastructure means extra labour costs
- ✓ Security — remote database fragments must be secured, and they are not centralized so the remote sites must be secured as well. The infrastructure must also be secured (for example, by encrypting the network links between remote sites).

C. *Types of DDBMS*

1. Homogeneous: The same DBMS (e g., Oracle) is used at each node.
2. Heterogeneous: The different DBMS (e g., Oracle, DB2).

D. *Failures in Distributed DBS*

Several types of failures may occur in distributed database systems:

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:

a. Total Failure where all the sites of a distributed system fail.

b. Partial Failure where only some of the sites of a distributed system fail.

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.

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.

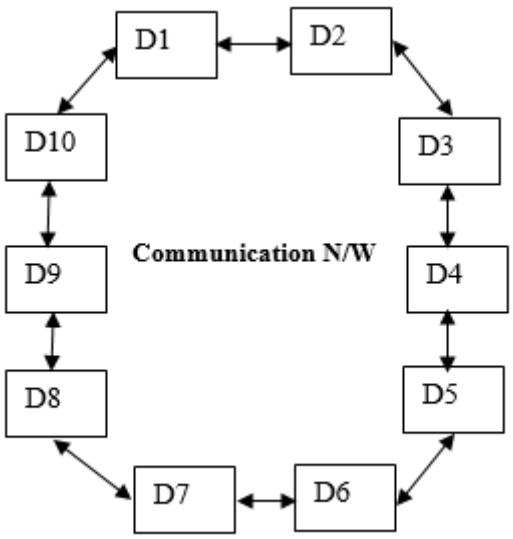

Figure 1. Distributed DBS architecture

III. FUNDAMENTALS OF DATABASE TRANSPARENCY

The distributed systems should be perceived as a single entity by the users or the application programmers rather than as a collection of autonomous systems, which are cooperating. The users should be unaware of where the services are located and also the transferring from a local machine to a remote one should also be transparent.

A. Types of Transparencies

The implementation of the distributed system is very complex, as a number of issues have to be considered to achieve its final objective. The complexities should not worry the user of the distributed system from using it i.e., the complexities should be hidden from the user who uses the distributed system. This property of the distributed system is called its transparency. There are different kinds of transparencies that the distributed system has to incorporate. The following are the different transparencies encountered in the distributed systems [12, 13].

1. Access Transparency: There should be no apparent difference between local and remote access methods. In other words, explicit communication may be hidden. For instance, from a user's point of view, access to a remote service such as a printer should be identical with access to a local printer. From a programmer's point of view, the access method to a remote object may be identical to access a local object of the same class. This transparency has two parts:
   a. Keeping a syntactical or mechanical consistency between distributed and Non-distributed access,
   b. Keeping the same semantics. Because the semantics of remote access are more complex, particularly failure modes, this means the local access should be a subset. Remote access will not always look like local access in that certain facilities may not be reasonable to support (for example, global exhaustive searching of a distributed system for a single object may be unreasonable in terms of network traffic).

2. Location Transparency: Clients should see a uniform file name space. Files or groups of files may be relocated without changing their pathnames. A location transparent name contains no information about the named object's physical location. This property is important to support the movement of the resources and the availability of services. The location and access transparencies together are sometimes referred as Network transparency. The examples are File system in NFS and the pages of the web.

3. Concurrency Transparency: Users and Applications should be able to access shared data or objects without interference between each other. This requires very complex mechanisms in a distributed system, since there exists true concurrency rather than the simulated concurrency of a central system. The shared objects are accessed simultaneously. The concurrency control and its implementation is a hard task. The examples are NFS, Automatic Teller machine (ATM) network.

4. Replication Transparency: This kind of transparency should be mainly incorporated for the distributed file systems, which replicate the data at two or more sites for more reliability. The client generally should not be aware that a replicated copy of the data exists. The clients should also expect operations to return only one set of values. The examples are Distributed DBMS and Mirroring of Web pages.

5. Failure Transparency: Enables the concealment of faults, allowing user and application programs to complete their tasks despite the failure of hardware or software components. Fault tolerance is provided by the mechanisms that relate to access transparency. The distributed systems are more prone to failures as any of the components may fail which may lead to degraded service or the total absence of that service. As the intricacies are hidden the distinction between a failed and a slow running process is difficult. Examples are Database Management Systems [14].

6. Migration Transparency: This transparency allows the user to be unaware of the movement of information or processes within a system without affecting the operations of the users and the applications that are running. This mechanism allows for the load balancing of any particular client, which might be overloaded. The systems that implement this transparency are NFS and Web pages.

7. Performance Transparency: Allows the system to be reconfigured to improve the performance as the load varies.

8. Scaling Transparency: A system should be able to grow without affecting application algorithms. Graceful growth and evolution is an important requirement for most enterprises. A system should also be capable of scaling down to small environments where required, and be space and/or time efficient as required. The best-distributed system example implementing this transparency is the World Wide Web.

9. Execution Transparency: Migration of processes will not be visible to the user.

10. Configuration Transparency: The configuration does not affect the programming or use of the system.

11. Network Transparency: The programmer/user does not have to know about the network. The system looks like a stand-alone computer.

12. Name Transparency: Names will not change when the system is reconfigured.

B. Levels of transparency

The three levels of transparency to hide certain complexities from the user, effectively managing the database as if it were centralized.

Fragmentation transparency**:** the highest level of transparency divides the original database into fragments and disperses them throughout the DDBMS. Therefore, the user does not need to specify fragment names or locations to gain access.

Location transparency: it only requires the user to know the names of the fragments.

Local mapping transparency: the lowest level of transparency requires the user to know the name and location of a fragments.

IV. REPLICATED DATA IN THE DDBMS

Replication is the process of creation and maintenance of duplicate versions of database objects in a distributed database system [15]. Replication improves the performance and increases the availability of applications by providing alternate data access options. For example, users can access a local database rather than a remote server to minimize network traffic and provide location transparency. Furthermore, the application can continue to function if parts of the distributed database are down as replicas of the data might still be accessible. Database replication is needed in the case of a system failure where in if a primary copy of the database is failed the secondary copy will be still there to retain the data. A replication service is required to maintain data consistency across these diverse environments. Distribution reduces the network costs for query access, and it improves application availability and consistency.

V. HOMOGENEOUS DISTRIBUTED DATABASE MANAGEMENT SYSTEM

In homogeneous distributed database system, the sites involved in distributed DBMS use the same DBMS software at every site but the sites in heterogeneous system can use different DBMS software at every site. While it might be easier to implement homogeneous systems, heterogeneous systems are preferable because organizations may have different DBMS installed at different sites and may want to access them transparently. [16] Distributed DBMS can choose to have multiple copies of relations at different sites or choose to have only one copy of a relation. The benefits of data replication is increased reliability – if one site fails, then other sites can perform queries for the relation. The performance will increase, as transaction can perform queries from a local site and not worry about network problems. The problem with data replication is decreased performance when there are a large number of updates, as distributed DBMS have to ensure that each transaction is consistent with every replicated data. This adds additional communication costs to ensure that all copies of the data are updated at the same time.

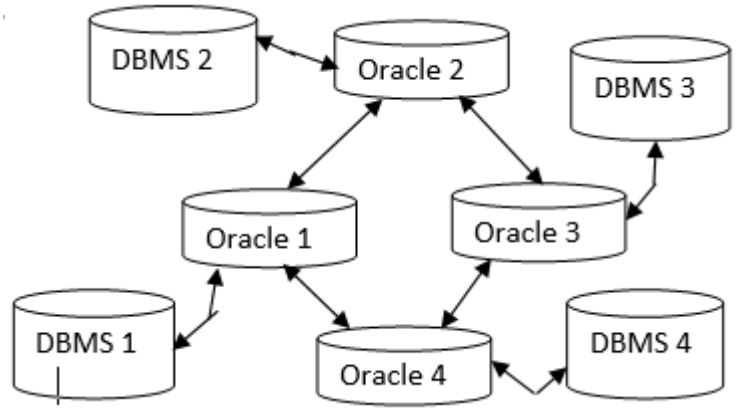

Figure 2. Homogeneous distributed database system

A homogeneous distributed database system is depicted in Figure 2. This environment is typically defined by the following characteristics (related to the non-autonomous category described previously):

- Data are distributed across all the nodes.
- The same DBMS is used at each location.
- All data are managed by the distributed DBMS (so there are no exclusively Local data).

VI. CONCLUSIONS

Homogeneous distributed database system based on transparency, the users should be unaware of where the services are located and also the transferring from a local machine to a remote one should also be transparent, such as communication may be hidden, physical location where is located, failure of hardware or software components, load

balancing of any particular client, Allows the system to be reconfigured to improve the performance as the load varies, Graceful growth and evolution is an important requirement for most enterprises etc. In the all situation we find the transparency in homogeneous distributed database system hidden all information and unaware of the users. We are using different types of transparency and levels of transparency but we find that we could not reduce operating system and communication stack through such types of transparency.

In this paper we investigate that if all transparency have an associated cost with others, matter for much research how the cost of implementing multiple transparencies interact and how to reduce operating system and communication stack, and

We also investigated and considered that if using such type of approaches such as: Application Layer Framing and Integrated Layer Processing, then it will reduce operating system and communication stack. In the future we will find that how to reduce the operating system and communication stack, when it was transparent an associated cost with others. And implement multiple transparencies interact how to.

REFERENCES

[1] R. Abbott and H. Garcia-Molin a, "Scheduling 'Real-Time Transactions: a Performance Evahration", F&c. of 14th VLDB Conj., August 1988.
[2] Sang Syuk Son, "Replicated Data Management in Distributed Database System", ACM Simgod, Vol.17, Issue 4, Dec 1998, Newyork, USA, pp-62-69.
[3] This article incorporates public domain material from the General Services Administration document "Federal Standard 1037C".
[4] O'Brien, J. & Marakas, G.M. (2008) Management Information Systems (pp. 185-189). New York, NY: McGraw-Hill Irwin
[5] R. Ramakrishnan. *Database Management Systems*. McGraw-Hill Book Company, 1998.
[6] O. Wolfson, S. Jajodia, and Y. Huang, an Adaptive Data Replication Algorithm, ACM Trans. Database Systems, vol. 22,no. 2, pp. 255-314, June 1997
[7] M.C. Little and D.L. McCue, the Replica Management System: A Scheme for Flexible and Dynamic Replication, Proc. Second Workshop Configurable Distributed Systems, Mar. 1994.
[8] S. Acharya and S.B. Zdonik, an Efficient Scheme for Dynamic Data Replication, Technical Report CS-93-43, Brown Univ., Sept. 1993.
[9] S. Ceri, M.A.H. Houtsma, A.M. Keller, and P. Samarati, a Classification of Update Methods for Replicated Databases, Technical Report STAN-CS-91-1392, Stanford Univ., Oct. 1991.
[10] T. Beuter and P. Dadam, Principles of Replication Control in Distributed Database Systems, Informatik Forschung und Technik, vol. 11, no. 4, pp. 203-212,1996, in German.
[11] A.A. Helal, A.A. Heddaya, and B.B. Bhargava, Replication Techniques in Distributed Systems. Kluwer Academic, 1996.
[12] George Coulouris, Jean Dollimore, Tim Kindberg, "Distributed Systems Concepts and Design" 3rd edition, Addison-Wesley
[13] http://www.cs.ucl.ac.uk/staff/W.Emmerich/lectures/DS97-98/dsee3.pdf.
[14] http://www.cs.ucl.ac.uk/staff/J.Crowcroft/ods/node18.html-SECTION0053000000000 0000000.
[15] May Mar Oo, "Fault Tolerance by Replication of Distributed Database in P2P System using Agent Approach", International Journal of Computers, Issue 1. vol.4, 2010.
[16] W. Cellary, E. Gelenbe, and T. Morzy. Concurrency Control in Distributed Database Systems. North- Holland, 1988.

Nitesh Kumar received his B.Tech degree (CS & E) in the year 2012 from AMIETE, New Delhi, and currently pursuing M.Tech in Computer Science Engineering with a Specialization in Database Engineering from KIIT University,Orissa, Bhubaneswar. His research area includes Distributed Database systems, Fuzzy Object Databases, Fuzzy Object-Oriented Database Modeling.

Saurabh Bilgaiyan received his B.E degree with Information Technology in the year 2012 from BIRT, Bhopal, and currently pursuing M.Tech. in Computer Science & Engineering from KIIT University,Orissa, Bhubaneswar. His research area includes Distributed Database systems, Fuzzy Object Oriented Database Modeling, Cloud Computing, and Soft Computing.

Santwana Sagnika received her B.Tech degree with computer science & engineering in the year 2011 from KIIT University, Orissa, Bhubaneswar, and currently pursuing M.Tech. in Computer Science & Engineering from KIIT University,Orissa, Bhubaneswar. Her research area includes Distributed Database systems, Image Processing, and Soft Computing.