An Overview of Replicated Data in DDBMS

Savyasachi, Amit Kumar Nirala

***Abstract*— A replication is some aspect of the distributed database system that is replicated from the user (programmer, system developer, and user or application program). A replication is provided by including some set of mechanisms in the distributed system at a layer interface where the replication is required. A number of basic replications have been defined for a distributed system. It is important to realize that not all of these are appropriate for every system, or are available at the same level of interface. In fact, all replicated have an associated cost, and it is extremely important for the distributed system implement or to be aware of this. It describes engineering solutions to archiving these replications, and attempts to outline the cost of the solutions. It is a matter for much research how the costs of implementing multiple duplications interact. This is part of current research in how to reduce operating system and communications stack overheads through such approaches as Application Layer Framing and Integrated Layer Processing.**

***Index Terms*— Database system, Distributed database system, Replicated data, Homogeneous data.**

I. INTRODUCTION

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 required 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 . 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 of 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.

Manuscript received oct, 2014.

***Savyasachi**, HOD CSE,AKU/ KKCEM/ KKCEM NALANDA BIHARSHARIF, INDIA, 9431266566*

***Amit kumar nirala**, CSE, AKU/ KKCEM/ KKCEM, BIHARSHARIF, INDIA.*

II. REPLICATED DATA IN DDBMS

Replication is the process of creation and maintenance of duplicate versions of database objects in a distributed data base system . 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.

A. Advantages to data replication

a. *Reliability:* If one of the sites containing the relation (or database) fails, a copy can always be found at another site without network traffic delays. Also, available copies can all be updated as soon as transactions occur, and unavailable nodes will be updated once they return to service.

b. *Fast response:* Each site that has a full copy can process queries locally, so queries can be processed rapidly.

c. *Possible avoidance of complicated distributed transaction integrity routines:* Replicated databases are usually refreshed at scheduled intervals, so most forms of replication are used when some relaxing of synchronization across database copies is acceptable.

d. *Node decoupling:* Each transaction may proceed without coordination across the network. Thus, if nodes are down, busy, or disconnected (e.g., in the case of mobile personal computers), a transaction is handled when the user desires. In the place of real-time synchronization of updates, a behind-the-scenes process coordinates all data copies.

e. *Reduced network traffic at prime time:* Often updating data happens during prime business hours, when network traffic is highest and the demands for rapid response greatest. Replication, with delayed updating of copies of data, moves network traffic for sending updates to other nodes to non-prime-time hours.

B. Disadvantages to data replication

a. *Storage requirements:* Each site that has a full copy must have the same storage capacity that would be required if the data were stored centrally. Each copy requires storage space (the cost for which is constantly decreasing), and processing time is required to update each copy on each node.

b. *Complexity related to maintaining database integrity:* Unless very costly mechanisms for maintaining identical copies of the database in real-time are used, it is essential ensure that potential discrepancies between the copies do not lead to business problems caused by inconsistent data. This requires potentially complex coordination requirements at the application level. This may cause undesirable coupling between the database and applications.

c. *Complexity and cost of updating:* Whenever a relation is updated, it must (eventually) be updated at each site that holds a copy. Synchronizing updating in near real- time can require careful coordination, as will be clear later under the topic of commit protocol.

C. Types of Replication

The replication tools may be selected based on type of replication it supports. The capabilities and performance characteristics varies from one type of replication to another. A replication strategy may be selected based on two basic characteristics: *Where* and *When*. When the data is updated at one site, the updates have to be propagated to the respective replicas. When the updates can be propagated can be achieved by Synchronous (eager) and Asynchronous (lazy) methods and where the updates can take place can be achieved by update everywhere and primary copy (master-slave) methods. *Synchronous replication* (Master-Slave replication) works on the principle of Two-Phase commit protocol. In a two-phase commit protocol, when an update to the master database is requested, the master system connects to all other systems (slave databases), locks those databases at the record level and then updates them simultaneously. If one of the slaves is not available, the data may not be updated. The consistency of data is preserved; however it requires availability of all sites at the time of propagation of updates. There exists two variations of *Asynchronous replication* (Store and Forward replication) i.e. Periodic and a periodic. In Periodic replication, the updates to data items are done at specific intervals and in a periodic replication the updates are propagated only when necessary (usually based on firing of event in a trigger). The time at which the copies are inconsistent is an adjustable parameter which is application dependent In Update anywhere method, the update propagation can be initiated by any of the sites. All sites are allowed to update the copy of the datum whereas in a Primary Copy method there is only one copy (primary copy or master) which can be updated and all other (secondary or slave) copies are updated reflecting the changes to the master. Various forms of replication strategies are as follows:

a. Snapshot Replication: In snapshot replication, a snapshot or copy of data is taken from one server and moved to another server or to another database on the same server. After the initial synchronization, snapshot replication can refresh data in published tables periodically. Though snapshot replication is easiest form of replication, it requires copying all data items each time a table is refreshed.

b. Transactional Replication: In transactional replication, the replication agent monitors the server for changes to the database and transmits those changes to the other backup servers . This transmission can take place immediately or on periodic basis. Transactional Replication is used for server-server scenarios.

c. Merge Replication: Merge replication allows the replicas to work independently . Both entities can work offline. When they are connected, the merge replication agent checks for changes on both sets of data and modifies each database accordingly. If transaction conflict occurs, it uses a predefined conflict resolution algorithm to achieve consistency. Merge replication is used mostly in wireless environments.

d. Statement based replication: The statement based replication intercepts every SQL query and sends it to different replicas . Each replica (server) operates independently. To resolve conflicts, Read-Write queries are sent to all servers whereas read only queries can be sent to only one server. This enables the read workload to be distributed. Statement based replication is applicable for optimistic approaches where each cache maintains the same replica.

e. Near-real-time replication: For near-real-time requirements, store and forward messages for each completed transaction can be broadcast across the network informing all nodes to update data as soon as possible, without forcing a confirmation to the originating node (as is the case with a coordinated commit protocol)before the database at the originating node is updated. One way to generate such messages is by using triggers. A trigger can be stored at each local database so that when a piece of replicated data is updated, the trigger executes corresponding update commands against remote database replicas. With the use of triggers, each database update event can be handled individually and transparently to programs and users. If network connections to a node are down or the node is busy, these messages informing the node to update its database are held in a queue to be processed when possible.

f. Pull replication: The schemes just presented for synchronizing replicas are examples of push strategies. Pull strategies also exist. In a pull strategy, the target, not the source node, controls when a local database is updated. With pull strategies, the local database determines when it needs to be refreshed and requests a snapshot or the emptying of an update message queue. Pull strategies have the advantage that the local site controls when it needs and can handle updates. Thus, synchronization is less disruptive and occurs only when needed by each site, not when a central master site thinks it is best to update.

g. Database integrity with replication: For both periodic and near-real-time replication, consistency across the distributed, replicated database is compromised. Whether delayed or near-real-time, the DBMS managing replicated databases still must ensure the integrity of the databases. Decision support applications permit synchronization on a table-by-table basis, whereas near-real-time applications require transaction-by-transaction synchronization. But in both cases, the DBMS must ensure that copies are synchronized per application requirements. The difficulty of handling updates with a replicated database also depends on the number of nodes at which updates may occur. In a single-updater environment, updates will usually be handled by periodically sending read-only database snapshots of updated database segments to the no updater nodes. In this case, the effects of multiple updates are effectively batched for the read-only sites. This would be the situation for product catalogs, price lists, and other reference data for a mobile sales force. In a multiple-updater environment, the most obvious issue is data collisions. Data collisions arise when the independently operating updating nodes are each attempting to update the same data at the same time. In this case, the DBMS must include mechanisms to detect and handle data collisions. For example, the DBMS must decide if processing at nodes in conflict should be suspended until the collision is resolved.

D. When to Use Replication

Whether replication is a viable alternative design for a distributed database depends on several factors:

a. ***Data timeliness*** Applications that can tolerate out-of-date data (whether this is for a few seconds or a few hours) are better candidates for replication.

b. ***DBMS capabilities*** an important DBMS capability is whether it will support a query that references data from more than one node. If not, the replication is a better candidate than the partitioning schemes, which are discussed in the following sections.

c. ***Performance implications*** Replication means that each node is periodically refreshed. When this refreshing occurs, the distributed node may be very busy handling a large volume of updates. If the refreshing occurs by event triggers (e.g., when a certain volume of changes accumulate), refreshing could occur at a time when the remote node is busy doing local work.

d. ***Heterogeneity in the network*** Replication can be complicated if different nodes use different operating systems and DBMSs, or, more commonly, use different database designs. Mapping changes from one site to *n* other sites could mean *n* different routines to translate the changes from the originating node into the scheme for processing at the other nodes.

e. ***Communications network capabilities*** Transmission speeds and capacity in a data communications network may prohibit frequent, complete refreshing of very large tables. Replication does not require a dedicated communications connection, however, so less expensive, shared networks could be used for database snapshot transmissions.

E. Fault tolerance in replicated data

Fault tolerance describes a procedure/technique that keeps the system in a consistent state in event of a failure by using redundancy. If a site fails the data it contains may become unavailable. By keeping several copies of the data at different sites, single site failure ensures the overall availability of the system.

a. Failover

Failover is the process of moving database, metadata and user connections from a failed or shutdown primary site to a secondary site so that users can still access the data thus preserving the availability. Failover may be an implicit or an explicit process. Full functionality of alternate site or secondary site is to be ensured in case if disaster happens at the primary site.

b. Failback

When the primary site resumes operations, the user performs a failback to restore the primary site in consistent state. The restoration process is usually initiated when the load on the secondary server is less. Otherwise it degrades the performance of the system.

c. Active-Active

In Active-Active mode, two servers are configured with independent workloads . They run on primary and secondary nodes respectively until one fails. When failover occurs, the secondary node takes over the primary and resumes activities until primary server fails back.

d. Active-Passive

In Active-Passive mode, the services are provided either by the primary node or by the secondary node. Initially the services are provided by a primary node before a fail over and by secondary node after fail over.

III. HOMOGENEOUS DISTRIBUTED DBMS

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 . 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.

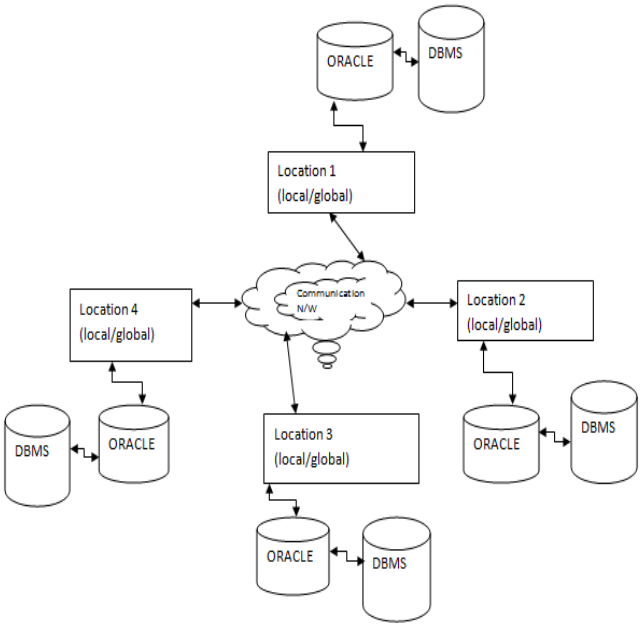

Fig I: Homogeneous distributed database environment.

A homogeneous distributed database environment is depicted in Figure II. 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).

IV. CONCLUSION

Homogeneous distributed database system based on replication, is provided by including some set of mechanism in distributed system at a layer interface, where the replication is required. its realize that not all of these are appropriate for every system, or are available at the same level of interface. all replication have an associated cost, matter for much research how the cost of implementing multiple replications interact and how to reduce operating system and communication stack overheads through such approaches as Application Layer Framing and Integrated Layer Processing.

REFERENCES

[1]. Sang Syuk Son, "Replicated Data Management in Distributed Database System", ACM Simgod,, Vol.17, Issue 4, Dec 1998, Newyork, USA, pp-62-69.

[2]. W. Cellary, E. Gelenbe, and T. Morzy. *Concurrency Control in Distributed Database Systems*. North-Holland, 1988.

[3]. R. Abbott and H. Garcia-Molin a, "Scheduling 'Real-Time Transactions: a Performance Evahration", F&c. of 14th VLDB Conj., August 1988.

[4]. R. Ramakrishnan. *Database Management Systems*. McGraw-Hill Book Company, 1998.

[5]. 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.

[6]. 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.

[7]. S. Acharya and S.B. Zdonik, An Efficient Scheme for Dynamic Data Replication, Technical Report CS-93-43, Brown Univ., Sept. 1993.

[8]. 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.

[9]. 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.

[10]. A.A. Helal, A.A. Heddaya, and B.B. Bhargava,Replication Techniques in Distributed Systems.Kluwer Academic, 1996.

[11]. This article incorporates public domain material from the General Services Administration document "Federal Standard 1037C".

[12]. O'Brien, J. & Marakas, G.M.(2008) Management Information Systems (pp. 185-189). New York, NY: McGraw-Hill Irwin

[13]. George Coulouris, Jean Dollimore, Tim Kindberg, "*Distributed Systems Concepts and Design"* 3rd edition, Addison-Wesley.

[14].http://www.cs.ucl.ac.uk/staff/W.Emmerich/lectures/DS97-98/dsee3.pdf.

[15]. http://www.cs.ucl.ac.uk/staff/J.Crowcroft/ods/node18.html - SECTION00530000000000 0000000.

[16]. 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.

[17]. Sujoy P.Paul, "Pro SQL Server 2008 Replication"

[18]. "PostgreSQL 8.4 Server Administration", Volume II- The PostgreSQL Global Development Group, 1996.

[19]. Will Schmied, Orin Thomas, "Implementing & Managing Exchange Server 2003", pp.219, 2004.

Savyasachi received his B.E degree in the year 2011 from RGPV Bhopal, and M.Tech from SRIT Jabalpur RGPV University, Bhopal. Currently Working as an H.O.D CSE, in KKCEM, Nalanda. His research area includes Data Mining, Operating System, Computer Network, Network Security.

Amit Kumar Nirala received his B.E. degree in the year 2010 from SIT, Tumkur and M.Tech in Computer Science from EWIT, Bangalore. Currently working as Assistant Professor in KKCEM, Nalanda. His research area includes Database systems, Computer Network, Operating System and Data Mining.

\

