Deadlock Managing Process in P2P System

Akshaya A.Bhosale
Department of Information Technology
Gharda Institute Of Technology,Lavel, Chiplun,Maharashtra, India

Ashwini B.Shinde
Department of Information Technology
Gharda Institute Of Technology,Lavel, Chiplun,Maharashtra, India

Abstract- Here P2P database system is used for sharing data between autonomous and heterogeneous decentralized system on web. Deadlock managing process is used for distributed and replicated database in P2P based on quorum system. It introduces a distributed serialization graph based approach to concurrent control and recovery in P2P environment. In order to solve the problem of concurrent updates and node failure, architecture which is based on quorum is used which helps in assigning unique timestamp to each distributed transaction. This process is fully decentralized because each client decides if it has the priority to write access.

Keywords – Partial replication, Replicated database, P2P, Deadlock, Quarum systems.

I. INTRODUCTION

In recent years P2P[2] systems have gained tremendous popularity. Support of a transaction processing facility in P2P systems would provide databases at a low cost. Extending distributed database algorithms such as 2PC and ROWA to P2P environments might not provide the best performance because the P2P systems [2] are characterized by high site failure rates and an unpredictable network topology. The choice of algorithms in building P2PDB is difficult because of the lack of information about the performance of database algorithms in P2P environments. This thesis analyzes the performance of one such algorithm, the epidemic algorithm against the performance of traditional database algorithms in simulated P2P environments.

P2P system and quorum system are used for sharing data between autonomous and heterogeneous decentralized system.P2P system is nothing but, (1) scalability in terms of the node number and the resource number; (2) node autonomy; (3) dynamicity, (4) resource heterogeneity, (5) decentralized control and (6) self configuration. In this system each node can act as: (1) a server when it offers its resources to another nodes, (2) client when it uses resources of other nodes, (3) router when it propagates coming queries and message to other nodes and (4) data source when it share its own data with system node. In P2P system peers are work to achieve specific needs. The nodes can be classified into several areas as: file sharing (music), distributed computing, distributed storage, communication and other areas.

Quorum system is a collection of subset of server replicas, every pair of which intersects. Quorum system allows solving the problem of consistency. Each subset or quorum act on behalf of the whole replicas group, which reduces server replicas load and decrease the number of message needed. Also overall availability is enhanced, since a single quorum is sufficient for the server to operate.

The solution for query and transactions processing depending on the replication framework. For managing deadlock algorithm is used, which order conflicting transaction per data item, then to break cycles between transactions. It achieves lower latency and message cost, or does not unnecessarily abort concurrent conflicting transaction. A fully decentralized approach to transaction management in P2P environments [4] is suitable for long running transactional processes. Each transaction owns a local serialization graph that reflects the conflicts it is involved in.

II. RELATED WORK

A. Distributed Database Simulation

This section gives details about high-level design and implementation of distributed database simulation. A site of a distributed database system consists of a Transaction Manager and a Local Database which contains a Scheduler.

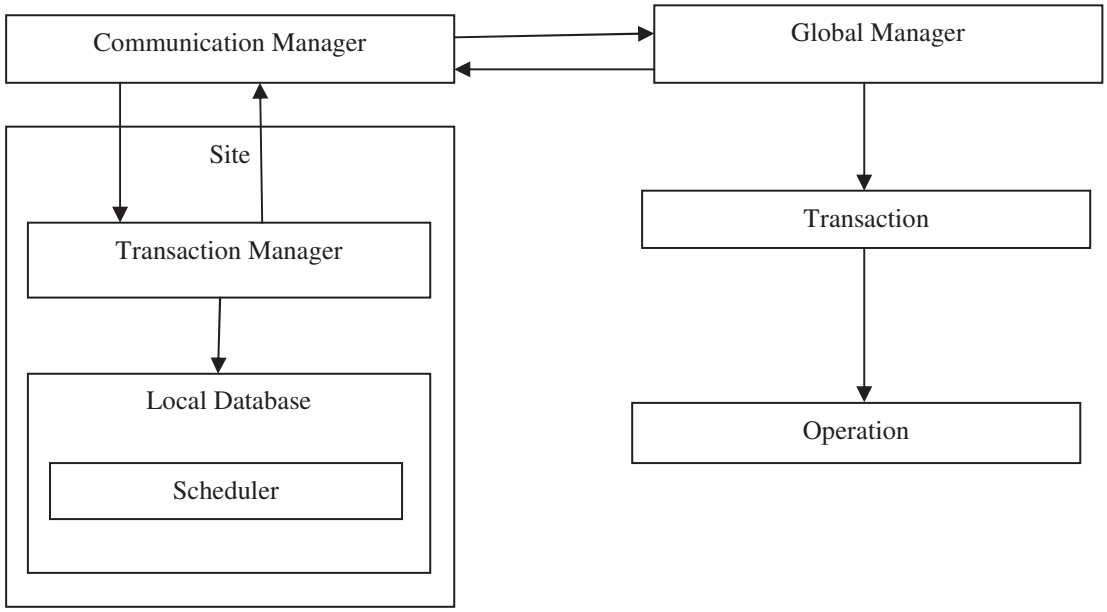

Figure 1: System Model for the Distributed Database Simulation

B. Global Manager

The Global Manager interacts with the transactions in the system, receives their operations and ensures that they take place at the right sites by sending them to the appropriate Transaction Manager via the Communication Manager. It is also equipped with a distributed deadlock detection mechanism, which is called at fixed intervals. If a deadlock is detected then the youngest deadlocked transaction is aborted. At every clock tick, the Global Manager has a method called *Process*, which is executed by the event clock. Figure 4.3 gives a description of the *Process* method.
The Global Manager has a receive Operation method which is called by the Communication Manager whenever it wants to send an operation to the Global Manger for reporting the results of an operation.

For all transactions that are not blocked

1. Get an operation.
2. If the operation is read, commit or abort the GlobalManager
    a) Sets the communication time.
    b) Sets the TransactionManager ID to the TransactionManagerID of the Coordinator site of the transaction.
    c) Enqueues the operation in the CommunicationManager's queue
    d) Wait for this operation to complete or abort before obtaining another operation from this transaction
3. If the operation is a write
    a) Submit an operation to each of the sites containing a copy of the data item in the manner described in step 2
    b) Wait for all the operations at all the sites to be completed before obtaining another operation from this transaction

Figure 2: The algorithm for process method of global manager

At a present deadlock detection time interval, construct a wait-for-graph and abort the youngest transaction by sending an abort operation to all the sites where the transaction took place.

C. Transaction Manager

The Transaction Manager is responsible for the execution of the operations at the site, ensuring atomicity of the transactions by implementing the 2PC protocol. It also maintains a TTF (Time to Failure) and a TTR (Time to Recover) for the particular site assigned during its instantiation. The event clock decrements the failure time counter

of each of the Transaction Managers for each of its ticks and as soon as the counter hits the TTF the Transaction Manager makes the site unavailable for a period of time equal to the TTR thus simulating site failure. After TTR has passed it executes the recoverFromFailure method and then resets its TTF to the present value and the cycle continues. The TransactionManager has a processOperationOrMessage method which is called by the CommunicationManager every time an operation has to be sent to this transaction manager.

1. If the site is not available due to failure, ignore the operation
2. If the site is available
   a) If the parameter is a read or write operation
      i. Submits the operation to the scheduler
      ii. Process the results from the scheduler
      iii. If the operation is not blocked simulate its execution by decrementing it for the duration equal to the duration of the operation
      iv. If simulated execution of the operation is completed send via the CommunicationManager
      v. If the operation did not obtain a lock, it sets the operation and thereby the transaction to the blocked status
      vi. If it is a commit operation. TransactionManager becomes the coordinator of the transactions and stats the 2PC Protocol
   b) If the parameter is abort operation
      i. Submit it to scheduler and processes all the newly unlock operations of other transactions as a result of the abort
      ii. Send the abort operation in a completed status to the GlobalManager via CommunicationManager
   c) If the parameter is message of Two Phase Commit protocol then the TransactionManager respond

Figure 3: processOperationorMessage method of transaction manager

D. Communication Manager

The Communication Manager mimics the network existing between the sites of the distributed database. When the CommunicationManager receives an operation/message from either the GlobalManager or the TransactionManger, it puts them in a queue. The event clock calls a method of the CommunicationManager called handleOperations which goes through the operations in the queue and decrements their communication counter. If the communication counter becomes zero it delivers the operation to the respective TransactionManager or the GlobalManager by calling their processOperationOrMessage function or the receiveOperation function respectively. This simulates the delivery of the operation or message through the network.

E. Operations

The Operations class used here is identical to the one used in the Centralized Database. It also serves as a message in the 2PC and has a communication time parameter called commTime which is set before the operation is placed in the CommunicationManager's queue for the purpose of communication.

There is a flag called *is Message* which when set indicates that this is a message of the 2PC. If it is a message, then the text of the message can be obtained from the parameter *message*.

F. Global Architecture

Global architecture is consisting of five layers: Timestamp Manager, Transaction Manager, Coordinator Manager, DBMS, and Database.

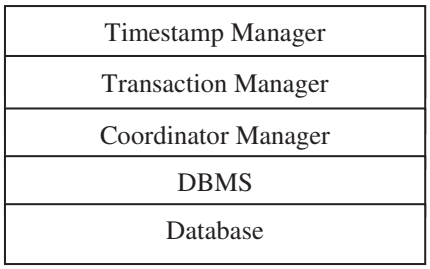

Figure 4: Global Architecture

I. Timestamp Manager [6]: This service assigns a unique timestamp to each transaction.
II. Transaction Manager [7]: The function of this service is to analyze a transaction from a client node, to divide it into sub transactions and queries and send them to the concerned servers nodes. It also has to balance the load between server's nodes.
III. Coordinator Manage[8]r: This service allows it's node to acts as coordinator in order to monitor the success of the transaction that has been divided into sub transactions and queries and sent to several different servers' nodes.
IV. DBMS: Each node has its own Data Base Manager System.
V. Database: This architecture is based on the database partially replicated, i.e. each node has got a fragment of the database.

G. Replication and Transaction Model

A single database is considered which is consisting of R1, R2… Rn relations that is partially replicated over a set S of m nodes {N1, N2…Nm}. Each relation Ri is duplicated in a replica subset Si (Si c s). The sunset replicas S1, S2…Sn consist of partition of S(i.e. $S1 \cap S2 \cap \ldots Sn = \phi$ and $S1^{U}S2^{U}\ldots Sn$).

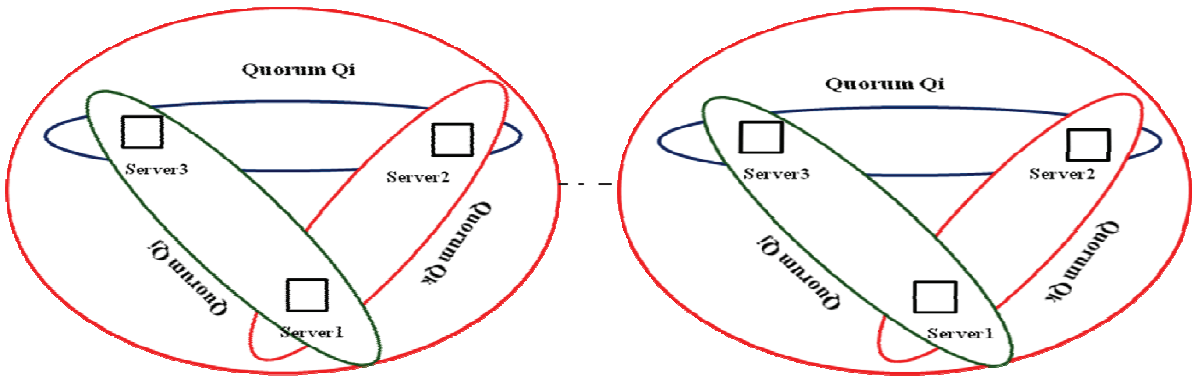

Figure5: Database partially replicated based on quorum system

Given a set of sites M, a set system universe Q={Q1, Q2,…Qn} is a collection of subset Qi $\subseteq$ N over N. A quorum system defined over N is a set system Q that has the following intersection property: $\forall i, j \in \{1\ldots n\}$, $Qi \cap Qj \neq \phi$

Here lazy multi-master replication scheme is used. Each node can be updated by any incoming transaction and is called the initial node of the transaction. Other nodes are later refreshed by propagating the updates through refresh transaction. Three transactions are involved in this scheme,

I. Update transactions are composed of one or several SQL statements which update the database.

II. Refresh transaction are used to propagate update transactions to the other nodes for refreshment. They can be seen as "replying" an update transaction on another node than the initial one. Refresh transactions are distinguished from update transaction by memorizing in the shared directory, for each data node, the transaction already routed to that node.

III. Queries are read-only transaction. Thus, they do not need to be refreshed.

II. Proposed Algorithms

A. *Deadlock Managing Process*

The proposed solution is fully decentralized in P2P environment [6] based on quorum system; because the client decides to write access if it collects the majority of the server accept. Thus, each node handles a clients queue which contains client with an associated number of the servers accept.

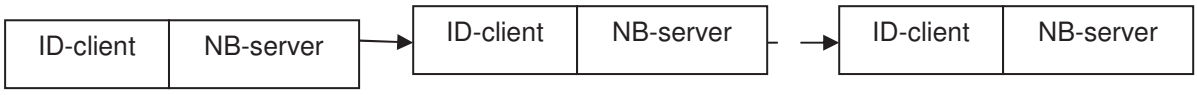

Figure 6: Clients queue

First, the client selects a quorum, and then it sends a WRITE message to each server replica of a selected quorum. The Write message contains the request to write or update value in the partial database. When server's replicas receive the WRITE message, they process to send back one of the two kinds of message. When a server replica receives more than one WRITE message, it send back an ACCEPT message to the first client and the REFUSE messages to the others clients. The REFUSE message contains the ID of the client which have receives the ACCEPT message from this server replica. When the client received the ACCEPT messages from servers replicas of the quorum selected, it increments it's NBR-Of-ACCEPT value. And, when it receives the REFUSE message from a server replica, it inserts the ID of the client (having received an ACCEPT message from this server replica) in the clients queue if is not in and initialized the NB-server to one (01), otherwise, it increments the NB-server value of the find ID-client. When the client receives all messages (ACCEPT and refuse) from servers replicas, it compares its NB-server value with the others of the clients which are saved in its clients queue. We can distinguish two cases.

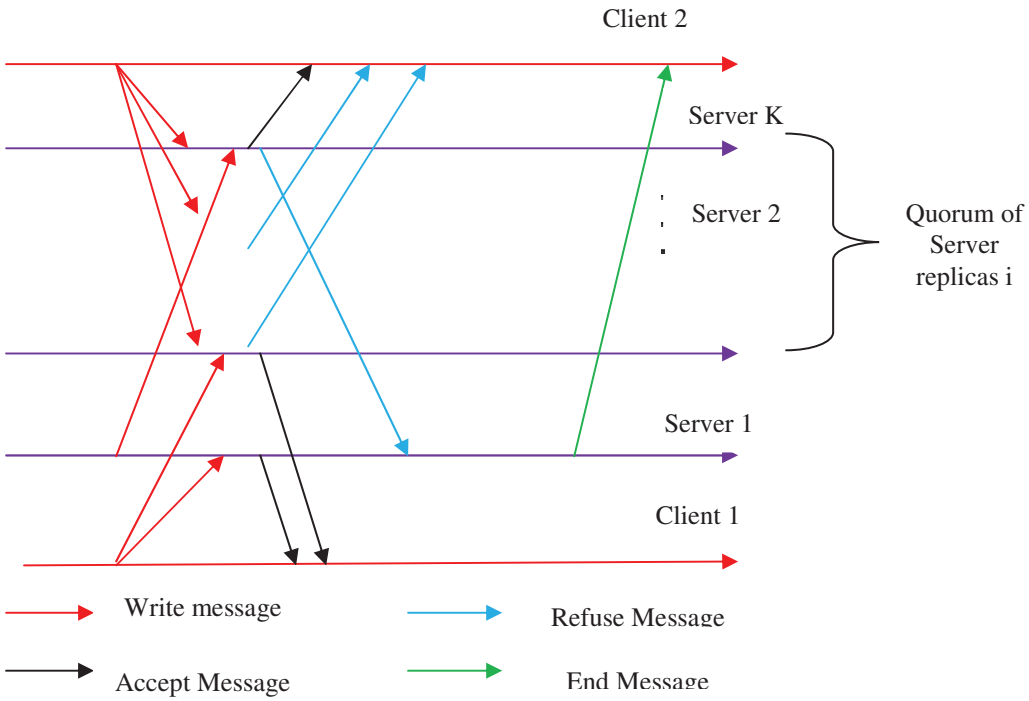

Figure 7: Clients interacting with servers replicas of a quorum to prevent deadlock

B. Algorithm runs on server and client node

This algorithm receives message from client nodes or from server's replica nodes. When the node is server replica, it can receive two kind of message: READ message and WRITE message, and when node is a client, it can receive the STATE message.

```
00  S_Q= {S_1,S_2,…,S_n}; //set of server replica in each quorum selected//
01  S_R= nil; //List of clients that have sent a READ message to this client//
02  Novel-stamp: integer;
03  begin
04   while true do
05   wait for next message from m;
06    switch message type of m do
07    case READ
08     if stamp is lock then
09     insert Ci in the end of the S_R list; //client C_i has sending READ message//
10     else
11      lock(stamp);
12      send-message STATE(locked, value of its stamp);
13      end-if;
14     end-case;
15     case STATE
16       stamp:=Max(stamp,STATE.stamp);
17          S_Q:=S_Q-{S_i};   //S_i: is a server replica has sending STATE message//
18       if S_Q=ϕ then
19       Novel-stamp: =stamp+1;
20       send-message WRITE(Novel-stamp,unlock);
21       end-if;
22       end-case;
23       Case WRITE
24          stamp: =novel-stamp;
25          unlock(stamp);
26          if S_R not empty then
27          Extract the first client message from the list S_R
28          goto 08;
29          end-if;
30        end-case;
32  end.
```

Figure 8: Algorithm runs on server and client node

C. Algorithm runs on client node

This algorithm treats only the case where the client sends a READ message to the entire server replica, and then it waits for the STATE message from this server replica according to above algorithm.

```
00  S_Q= {S_1,S_2,…,S_n}; //set of server replica in each quorum selected//
01  S: server replica;
02  begin
03  lock(stamp);
04  foreach S ϵ S_Q do
05  send-message READ(read stamp value, lock);
06  end-foreach;
07  wait for replay from all server replica in S_Q;
08  end;
```

Figure 9: Algorithm runs on client node

D. Algorithm to avoid deadlock in quorum system

In the first case, when the NB-server value of this client is the highest, this client decides to write access if it is the unique client having the highest NB-server value, else the client having the lower ID-client decides to write access. And in case when the NB-server value of this client is not the highest, this client must waiting the END message from the client (having the highest NB-server value), to remove it from its clients queue and repeat the comparison until it became the selected client to Write access.

Ss={s1,s2,……,sn};//set of server replica in each quorum selected//

```
1.  NB-server : Number of server having send an accept message initialized to 0;
2.  L-C:list clients having received an accept message from server replicas;
3.  begin
4.  while true do
5.     sends WRITE message to all server replicas of the selected quorum
6.  Repeat
7.       wait for receiving message from server replica;
8.       switch message type do
9.          case ACCEPT message
10.           NB-server:=NB-server+1
11.         end-case
12.         case REFUSE message
13.           ID-client:=extract ID client from REFUSE message
14.          if ID-client not exist in L-C then
15.             insert L-C.ID-client;
16.             L-C.NB-server:=1;
17.          else
18.             L-C.NB-server:=L-C.NB-server+1
19.          end if;
20.       end-case;
21.       sends END message when finishing write access
22. end-while;
23. end.
```

Figure 10:-Algorithm to avoid deadlock in quorum system

IV.CONCLUSION

It provide architecture for fully decentralized approach to a distributed transaction management in P2P environment is based on timestamping, managing transactions and construction of local precedence order graph. The algorithm in the timestamp manager attaches unique timestamp to each transaction in P2P environment. The algorithm in transaction manager transactions are divided into sub transactions and queries and it also selects server replica from a quorum system for each sub queries. It also uses coordinator manager layer, which coordinate with others coordinators in order to execute the distributed transaction correctly. To avoid deadlock algorithm is used, which gives fully decentralized solution because each client decides itself if it had the priority to write access or not.

V.FUTURE WORK

This research integrates new functionality in this architecture, like how the transaction manager selects server's replicas in order to balance the load between these servers.

REFERENCES

[1] A .Sousa, F. Pedone, R. Oliveira, and F. Moura, "Partial replication inthe database state machine". IEEE International Symposium on Network Computing and Applications (NCA'01), 2001 pp.0298.

[2] DD. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B.Richard, S. Rollins and Z. Xu..: "Peer-to-Peer Computing". Tech.Report: HPL-2002-57, available on line at:http://www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf

[3] S. A. Theotokis and D. Spinellis. : "A survey of peer-to-peer content distribution technologies". ACM Computing Surveys, 36(4), 2004, pp335-371.

[4] K. Haller, H. Schuldt and C. Turker, "A Fully decentralized Approach to coordinating transactional processes in Peer-to-Peer environments".Technical Report, , Institute of Information Systems, Swiss FederalInstitute of Technology (ETH), Zürich, Switzerland, n°463, 2004.

[5] N. Schiper, R. Schmidt, and , F. Pedone, "Optimistic algorithms for partial database replication". Proc. of the 10th International Conferenceon Principles of Distributed Systems (OPODIS'2006), December 2006,pp. 81–93.

[6] S-M. Hemam and K-W. Hidouci.: "A Fully Decentralized Algorithm to Timestamping Transactions in Peer-To-Peer Environments". IEEEInternational Conference on Machine and Web Intelligence ICMWI'10.October 2010, pp 164-168.

[7] S-M. Hemam and K-W. Hidouci.: "A Fully Decentralized Algorithm To Processing Transactions in a Peer-To-Peer Environments". International Conference on Information Technology and e-Services ICITeS'11. April 2011, PP 264-269.

[8] S-M. Hemam and K-W. Hidouci.: "Replicated Database Transactions Processing in Peer-To-Peer Environment". Journal of Networking Technology Volume 2 Number 2 June 2011, pp 63-72.