International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 2, Issue 2, March – April 2013 ISSN 2278-6856

Deadlock Detection in Distributed Database

I.Priyadarshini[1], Prof. Dr. S. S. Sane[2], Rutuja Jadhav[3]

[1,3] Dept. of Computer Engg.
KKWIEER, Nasik, India
[2]Head, Dept. of Computer Engg.
KKWIEER, Nasik
Nasik, India

Abstract: ***In distributed database, data resides in various sites and many transactions can originate at any number of sites randomly. These transactions can execute concurrently. This concurrency leads to deadlock in which transactions may enter into an infinite waiting state so deadlock handling is an important criteria in distributed transaction processing. So, an efficient algorithm for detecting deadlock is to be devised. Various approaches are there for detecting deadlock in distributed database such as chandy & Mishra Algorithm [5], Sinha's Algorithm [8], Obermack's Algorithm [3].All of which have been implemented and tested in distributed database where data is distributed among various sites and data is not replicated. As data replication improves availability, it is necessary to implement and test the performance of deadlock detection algorithm in a replicated environment.***

Keywords: Distributed Database, Deadlock, Replication

1. INTRODUCTION

Deadlock handling is an important component of transaction management in a database system. In this paper an algorithm for detecting and resolving deadlock in distributed database is discussed which can improve in development of transaction Management. In distributed database with replication same data may reside in several locations. A transaction initiated at one site can request data for which it is not the owner. Deadlock occurs in database system that permits concurrent execution of transaction using locking protocol. Deadlock detection is very difficult in distributed database system because no controller has complete and current information about the system and data dependencies. This new algorithm is based on creating a Linear Transaction Structure (LTS), Distributed Transaction Structure (DTS) finding local and global cycle, deciding priority ID of the transaction and aborting the selected victim. It also ensures that it will not detect false deadlock.

In Section 2 a survey of existing algorithms is discussed and in section 3 a new technique is discussed and in section 4 architectural model is presented and in section 5 experimental setup and conclusions in section 6.

2. LITERATURE SURVEY

The taxonomy of databases and locking methods is presented below.

Taxonomy of database:

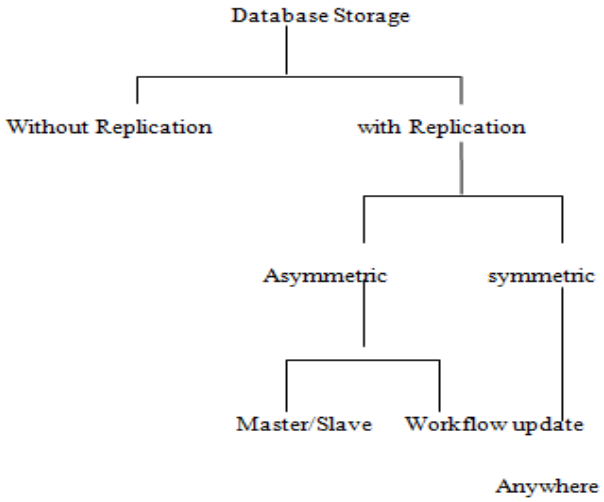

Figure 1. Classification of Data Storage

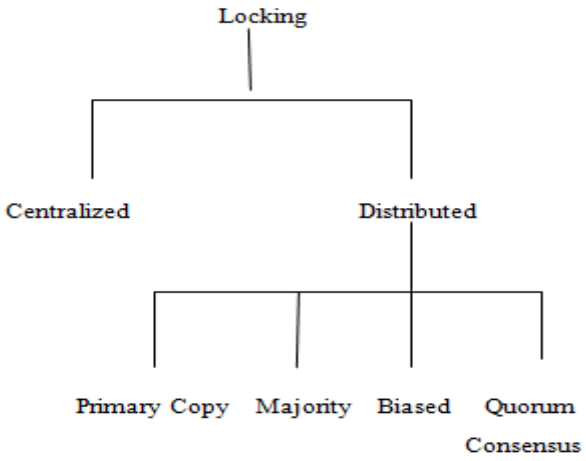

Figure 2. Classification of Locking Protocols

There are various algorithms existing for deadlock detection in distributed database. A survey of such algorithm is done in this section. The existing deadlock detection algorithm are divided into two category

1. Pass Information about transaction request to maintain a global wait for graph.
2. Simpler messages are sent among the transactions. No global wait for graph is explicitly constructed.

Ho's Algorithm [9]

In this Algorithm each site maintains a status table for all the processes that are initiated at the site. For each process, the tables keep track of the resources the process

has locked and the resources for which it is waiting for. Periodically a site is chosen as a controller and the following things happen.
Phase one:
(i)The Controller broadcast a message requesting all the sites to send their status table.
(ii) When all the sites have sent their status table it constructs a wait for graph.
(iii)If a cycle is detected then initiates the second phase else reports no deadlock and releases its control.
Phase Two:
(i) It is a verification phase. It broadcast a second message requesting everyone to send their status table.
(ii) When it receives the entire message then it constructs a wait for graph.
(iii) If there is a cycle it reports deadlock to a deadlock resolver. Else releases its control.
Advantage:
It is simple to implement.
Disadvantages:
It requires 4n messages to determine a deadlock in n site system.
It may detect false deadlock.

Obermack's Algorithm [3]

In this approach an external node T_{ext} is added to a local wait for graph to indicate the agent at remote site.
(i) When a transaction T1 at site s1, creates an agent at site s2 then an edge is added to the local WFG from T1 to T_{ext} node at site1.
(ii) Similarly at site S2 an edge is added to the local wait for graph from the T_{ext} node to the agent of T1.
(iii) If a local WFG contains a cycle that does not include T_{ext} then the site is in deadlock and the deadlock can be broken at local site.
(iv)A global deadlock is detected if any local WFG contains a cycle including T_{ext} node. Then to determine the deadlock graphs has to be merged.
(v) If site s1 has a deadlock, its local WFG is of the form $T_{ext} \rightarrow Ti \rightarrow Tj \rightarrow \ldots Tk \rightarrow T_{ext}$.
(vi) A time stamp is allocated to each transaction and imposes a rule that site s1 will send WFG to the site Tk is waiting for, say Sk if and only if ts (Ti) < ts(Tk).
(vii)Site Sk will include it in its WFG and check for cycle not involving T_{ext}
(viii). If there is no cycle, the process continues until either a cycle appear or entire global WFG is constructed and no cycle has been detected.
Performance Analysis:
(i) It requires n (n-1) messages to be transmitted for n sites.
Advantages:
The number of messages to be transmitted is less when compared to HO'S Algorithm.
Disadvantages:
It may detect false deadlock because wait for graph constructed do not represent a snap shot of global TWFG at any instant.

Chandy & Mishra Algorithm [5]

Chandy-Misra-Haas's distributed deadlock detection algorithm for AND model is based on edge-chasing.
(i) The algorithm uses a special message called probe, which is a triplet (i, j, k), denoting that it belongs to a deadlock detection initiated for process Pi and it is being sent by the home site of process Pj to the home site of process Pk
(ii) A probe message travels along the edges of the global WFG graph, and a deadlock is detected when a probe message returns to the process that initiated it.
Performance Analysis:
Every single deadlock detection computation involves at most e probes, where e is the number of communicating pairs of controllers in the network. Hence in the worst case e = N (N - 1) and N is the number of nodes. It may detect false deadlock.

3. MICHAEL'S ALGORITHM [1]

The algorithm proposed by Michael [1] for deadlock detection is presented below.

The new technique uses the following:

1. Linear Transaction Structure (LTS) for each local site.
2. Distributed Transaction Structure (DTS) for global resource transaction communication.

In this technique, a Linear Transaction Structure (LTS) is maintained at each site.

i. LTS Creation:

If any transaction Tp requests a data item that is held by another transaction Tq of same site then this technique stores the values of p and q to the linear transaction structure (LTS), where p and q represents their corresponding transaction number.

ii. DTS Creation:

Distributed Transaction Structure (DTS) stores all the transactions which are interconnected (requests for data item from other sites) from one site to another site. DTS also records the transaction's (i.e. for transactions connected to other site) intra requests DTS is managed by Data Manager (DM).

To detect local deadlock LTS of the site is checked. If there is cycles then the priority (which is assigned by local transaction manager at the time of initiation using timestamp) of the transaction involved in the cycle are entered into a queue Q maintained by that transaction manager of that site. Based on the priority, a victim is chosen.

To detect a global deadlock GTM records priority transaction id in TQ for those transactions which form cycles in DTS. The priority id which is least has lowest priority and it is the youngest transaction. Less priority id for the transaction's data request from one site to another

site, is given in DTS, global deadlock cycles become free from deadlock after aborting the transaction's data request from one site to another site.

Performance Analysis:

It detects local deadlock as well as global deadlock and it resolve deadlock by selecting a victim transaction and it does not detect false cycle. When any transaction has to wait its status is entered into the table.

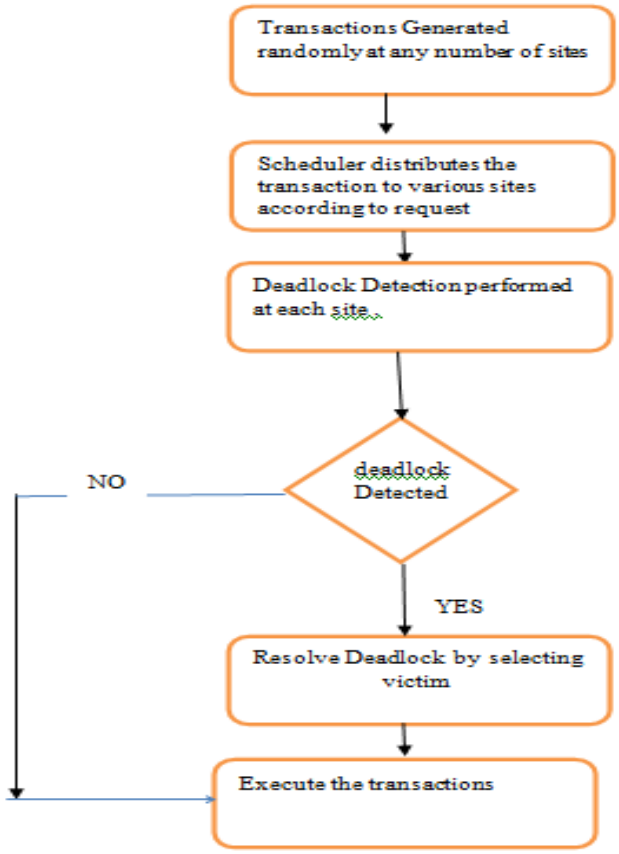

Figure 3: System Architecture diagram

4. EXPERIMENTAL SETUP

A. Software used
   a. Java Development Kit 1.6 or later
B. Hardware specification
   a. RAM 1GB
   b. Pentium 4 or later
C. Platform
   a. Windows
D. Tools
   a. Net beans 6.8 IDE

A. Sample Test Cases:

Distributed Database:

The data is randomly distributed across required number of sites.

Input:

The input to the above system is given in two manners.

1. User can define the transaction in files and give it as input to the system. Transactions are set of read and write statements. The system is tested with various scenarios of deadlock and without deadlock.

Sample Scenario 1:

Site 1	Site 2	Site 3
Transaction 1	Transaction 3	Transaction 5
Read 1	Write 3	Read 5
Write 2	Write 4	write 6
Write 3	Write 5	Write 1
Transaction 2	Transaction 4	Transaction 6
Read 10	write 0	read 11
Read 11	read 10	read 10
Read 12		read 12

DTS (for deadlock detection)

Requestor (trans.id)	Holder (trans.id)
1	3(for resource 3)
3	5(for resource 5)
5	1(for resource 1).

Circular wait condition

Deadlock Detected between transactions 1, 3 and 5.
Victim selected as T5 based on Timestamp of transaction T5:aborted.all other executed successfully.

2. Alternatively, The transactions are not user defined in files rather they are generated randomly by simulation methods in which 80% transaction are read only as in real case and the system is tested.

6. Conclusions

This Paper makes a survey of various algorithms of detecting deadlock in distributed database and also shows the implementation details of one such algorithm. Now, the same algorithm has to be tested on the database of replicated environment and its correctness has to be verified. In that case the locking mechanism will change, any of replicated concurrency control protocol such as primary copy has to be incorporated and further testing has to be done in order to analyze the correctness.

REFERENCES

[1]B. M. M. Alom, F. Henskens, and M. Hannaford, "Deadlock Detection Views of Distributed Database," in International conference on Information Technology & New Generation (ITNG-2009) Las Vegas, USA: IEEE Computer Society, 2009.

[2] A. K. Elmagarmid, "A Survey Of Distributed Deadlock Detection Algorithms," SIGMODRECORD, vol. 15: 3, pp. 37-45, 1986.

[3] R. Obermarck, "Distributed Deadlock Detection Algorithm," ACM Transaction on Database Systems, vol. 7:2, pp. 187-208, 1982.
[4] J. N. Gray, "A discussion on distributed systems," IBM Research Division, 1979.
[5] K. M. Chandy, J. Misra, and L. M. Hass,"Distributed Deadlock Detection," ACM Transaction on Computer Systems, vol. 1:2, pp.144-56, 1983.
[6] X. M. Chandy and J. Misra, "A Distributed Algorithm for Detecting Resource Deadlocks in Distributed Systems" in ACM, 1982.
[7] D. A. Menasce and R. R. Muntz, "Locking and Deadlock Detection in Distributed Data Bases "IEEE Transaction on Software Engineering, vol.5:3, pp. 195-202, 1979.
[8] M. K. Sinha and N. Natarjan, "A Priority Based Distributed Deadlock Detection Algorithm "IEEE Transaction on Software Engineering, vol. 11:1, pp. 67-80, 1985.
[9] G. S. Ho and C. V. Ramamoorthy, "Protocols for Deadlock Detection in Distributed Database Systems " IEEE Transaction on Software Engineering, vol. 8:6, pp. 554-557, 1982.
[10] S. Kawazu, S. Minami, K. Itoh, and K. Teranaka,"Two-Phase Deadlock Detection Algorithm in Distributed Databases" in IEEE, 1979.

AUTHORS

Prof. I.Priyadarshini, BE Computer engr, K.K.Wagh Insitute of engg.education anresearch.Nasik.

Dr. S.S.Sane, PHD, HOD K.K.Wagh Insitute of engg.education and research. Nasik.

Prof.Rutuja Jadhav, BE Computer engr, K.K.Wagh Insitute of engg.education and research.Nasik.