A Survey on Distributed Deadlock and Distributed Algorithms to Detect and Resolve Deadlock

Ambika Gehlot
AITR
Indore, India
Ambikagehlot9@gmail.com

Akansha Jaiswal
AITR
Indore, India
akanshasmily@gmail.com

Vandana Kate
AITR
Indore, India
vandana.kate@gmail.com

***Abstract*- Operating system and distributed database system (DDS) are the most ordinary places where the chances for the occurrence of deadlock are very high. DDS is a database system which keeps the storage of multiple and logically interrelated databases on multiple computer systems connected over a large network in order to achieve optimal performance and resource sharing. In DDS a deadlock might encounter when a transaction request resource from other blocked transaction and enters into wait condition. It is more difficult to resolve deadlock in DDS as compare to resolution of deadlock in operating system because sites do not have exact knowledge of state of system. In this paper we are presenting a comparative study of various distributed algorithms such as B. M. Alom Algorithm and Edge-Chasing Algorithm for deadlock detection and resolution in DDS.**

***Keywords*- Distributed database system (DDS), Deadlock, Transaction, Local and Global deadlock cycle, Lock, Wait-For-Graph(WFG), Deadlock Detection and Resolution.**

I. INTRODUCTION

In distributed database system [1] data is stored on multiple sites and execution of transaction can initiate from any site. Transaction is sequence of read, write, lock and unlock operations which are executed as a single unit. In DDS transactions [2] are executed concurrently i.e. concurrent updates [3] are performed on data items. Deadlock in DDS results because of concurrency control mechanism which prevents multiple transactions from coming in one another's way while executing simultaneously. Deadlock leaves transactions in never ending waiting condition i.e. due to infinite waiting state transactions are killed automatically which decreases efficiency of DDS drastically so it is very important to prevent, detect and resolve deadlock as soon as possible.

A. Distributed Database System

In distributed database (DDB) [4] two or more computers are connected together by telecommunication and these computers store multiple and logically [5] interrelated databases within a network which contains enormous number of sites (computer) and each site has its own local database i.e. many databases are distributed over various locations and each site consists of many independent transactions. The software application which manages DDB is known as Distributed Database Management System (DDBMS) [6] and combination of DDB and DDBMS is known as Distributed Database System (DDS).

For example- There are various colleges of an university in several cities so it is obvious that databases used in these colleges is distributed over various locations logically. So data sharing is also achieved by DDS. DDS provides facility to process multiple transactions simultaneously on many machines.

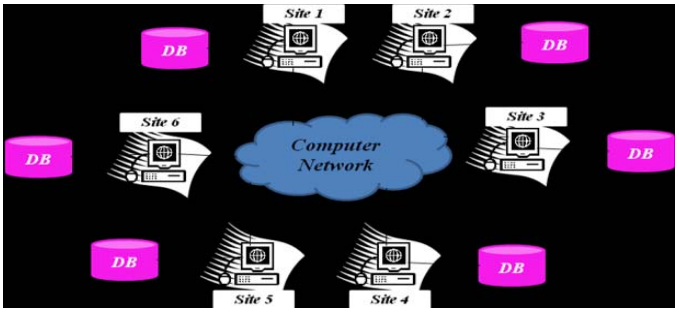

Fig. 1 Distributed Database System

B. Deadlock

Deadlock [7] is a condition which occurs in concurrent programming when a set of processes enter into infinite wait conditions. A process or transaction may exist in two states: running state or active state or blocked state. In active state all resources or data items which are required by a process have already been allotted to that process and that process is either executing or ready for execution. In blocked state a process has to wait for required resources because some other process is holding that resource. So in a system when a set of blocked processes end up their execution waiting for each other to free up the resources then the system is said to have gone into the state of Deadlock.

In Fig. 2 [8] process P_A is waiting for resource R_B which has already been allotted to process P_B. P_B in turn is requesting for resource R_A which is held by P_A so both processes will end up infinitely waiting for each other which results in Deadlock.

Fig. 2 Deadlock

978-1-5090-0669-4/16/$31.00 © 2016 IEEE

C. Locking

Locking [6] is a part of concurrency control and maintains integrity of data while permitting multiple users to access data simultaneously. There are two rules for performing operations (read or write) in DDS:

1) If different transactions are performing read [9] operation (performed by SELECT command of SQL) on a same data item then any number of transactions can read that data item concurrently at any time.
2) Write [9] operation performed by transactions work in different manner as compare to read operation performed by transactions. If any transaction is performing the write operation (performed by INSERT, UPDATE, DELETE command of SQL) on a same data item then the value of that data item goes into inconsistent state and that value remains inconsistent until the time at which write operation gets over.

On the basis of above two rules regarding read or write operation two types of locks are there:

1) Shared Lock

To read any data item, a transaction holds shared lock [10] on that data item and that same data item can also be read (shared locked) by other transactions. Thus this type of lock is known as shared lock.

2) Exclusive Lock

To read/write the any data item, a transaction holds exclusive lock [10] on that data item and that data item and that same data item cannot be read/write (either shared or exclusive locked) by any other transaction. Thus this type of transaction is known as exclusive transaction. Exclusive lock provides us the assurance that we can't make multiple updates on the same data item simultaneously.

However, deadlock in DDS will occur only because of exclusive lock. Shared lock can never cause deadlock.

D. Transactions in deadlock condition in DDS:

When there is a set of transactions [6] in which each transaction waits in a circular way to release lock on data item held by other transaction then only deadlock occurs in DDS. When two transactions T1 and T2 [11] exist in following fashion then deadlock occurs in DDS.

T1: Transaction T1 is running at site 1. It first accesses (INSERT, UPDATE or DELETE) data item A i.e. applies lock on data item A at site1 and then tries to access data item B i.e. tries to apply lock on data item B at site2.

T2: Transaction T2 is running at site 2. It first access(INSERT, UPDATE or DELETE) data item B i.e. applies lock on data item B at site2 and then tries to access data item A i.e. tries to apply lock on data item A at site1.

Now if T1 doesn't unlock A then T2 won't be able to apply lock on A and if T2 doesn't unlock B then T1 won't be able to apply lock on A as a result of which both of the transactions will have to wait infinitely in order to apply lock on required resources and will be killed automatically. This is the simplest form of Deadlock comprising of only two transactions. Deadlock may be much more complex comprising of more than two transactions.

TABLE I DEADLOCK CONDITION IN DISTRIBUTED ENVIRONMENT

Time	Transaction	Response	Data Item "A"	Data Item "B"
0				
1	T1: Lock-X(A)	Lock Granted!!	Unlocked	Unlocked
2	T2: Lock-X(B)	Lock Granted!!	Locked	Locked
3	T1: Lock-X(B)	T1 Waits for T2 to Release Lock on B at Site 2	Locked	Locked
4	T2: Lock-X(A)	T2 Waits for T1 to Release Lock on A at Site 1	Locked	Locked
5	T1: Lock-X(B)	T1 Waits for T2 to Release Lock on B at Site 2	Locked	Locked
6	T2: Lock-X(A)	T2 Waits for T1 to Release Lock on A at Site 1	Locked	Locked

II. RELATED WORK

Here we shall discuss few algorithms for deadlock detection and resolution which have been proposed earlier.

Chandy et. al [28] have proposed an algorithm which uses transaction wait-for-graphs (TWFG). This technique makes the use of probe computation for deadlock detection. Probe computation is a method by which transaction Ti determines if it is deadlocked or not. It checks the status of transactions at the local sites and uses probes to detect global deadlocks. A probe is issued whenever a transaction begins to wait for another transaction and is directed from one site to another based on the status of the transaction that received the probe. The probes are meant only for deadlock detection but they do not have any relation with requests and replies. A transaction sends at most one probe in any probe computation. If the initiator of the probe computation gets back the probe, then it is involved in a deadlock. This scheme does not suffer from false deadlock detection even if the transactions do not obey the two-phase locking protocol.

Obermack et al. [27] proposed an algorithm which at each site constructs and analyzes directed TWFG (transaction wait-for-graph) and uses a distinguished node at each site. This node is called "external" and is used to represent the portion of TWFG that is external (unknown) to the site. This algorithm does not work in a correct manner; rather it detects false deadlocks because the wait-for

graphs constructed to represent a snap-shot of the global TWFG at any instant.

Ho et al. [29] proposed an approach, in which the transaction table at each site maintains information regarding resources held and waited on by local transactions. The resources table at each of the sites maintains information regarding the transactions holding and waiting for local resources. Periodically, a site is chosen as a central controller responsible for performing deadlock detection. The drawback of this scheme is that it requires 4n messages, where n is the number of sites in the system.

III. REPRESENTATION AND DETECTION OF DEADLOCK IN DISTRIBUTED SYSTEM

Deadlock in DDS can be represented or detected by a graph called wait-for-graph (WFG) [12] which is a directed graph. WFG consists of set of vertices and set of edges. Transactions are represented by set of vertices and waiting states of transactions are represented by set of edges. Suppose there are two transactions T_A and T_B. A directed edge [13] from T_A to T_B { $T_A \rightarrow T_B$ } will represent that T_A is blocked and waiting for T_B to release the data item on which T_B has acquired exclusive lock. The necessary condition for deadlock to occur is cycle in WFG. The deadlock cycle in WFG may be local or global [14].

When the multiple transactions of a single site are part of deadlock cycle in WFG then it is said to local deadlock cycle and when multiple transactions of multiple sites are part of WFG is said to be global deadlock cycle.

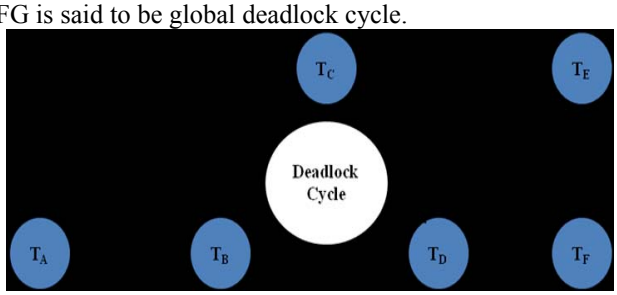

Fig. 3 WFG with cycle as well as with free transactions

Here transactions T_B, T_C and T_D are part of cycle therefore T_B, T_C and T_D have gone into the state of deadlock while T_A, T_E and T_F are not the part of cycle therefore T_A, T_E and T_F are deadlock free states.

Whenever a deadlock cycle is detected in WFG it must be resolved by aborting [18] one of the transactions present in a cycle.

A. Deadlock handling strategies in DDS

1) Deadlock Avoidance

It deals with deadlock [15] before the occurrence of deadlock by performing a check which represents that resource will be granted to process only when the resultant global state will be secured. A global state will be safe when all processes/transactions included in distributed system will complete their execution. But because of some reasons it is not workable [16] for distributed system.

2) Deadlock Prevention

It assures that at least one of the conditions responsible for occurrence of deadlock [15] must never hold. It is achieved when all resources are granted to process concurrently before process starts its execution [16] or if any active process requests for the resource which has been held by blocked process then the blocked process frees up that resource. But the disadvantage of deadlock prevention is that it is not workable and also provides inefficiency to the distributed system.

3) Deadlock Detection and Recovery

It permits deadlock [15] to occur in system and then discover deadlock and then finally bring deadlock to end by depending on WFG. First of all it is analyzed that as a result of interactions of transactions and data items whether the cycle [11] exists in WFG or not. If the cycle exists then it will remain in system till the time until it is broken. Once the cycle gets broken then deadlock gets recovered from the system. Deadlock detection and recovery proceed simultaneously with the regular activities of the system so the overall performance of system does not get affected.

"Deadlock Detection and Recovery" found to be most commonly used [16] and best approach to deal with deadlock because it is very difficult to implement "Deadlock Avoidance" and "Deadlock Prevention".

IV. DEADLOCK RESOLUTION USING DISTRIBUTED ALGORITHMS

Distributed algorithms [17] make sites to be cooperative equally in detecting deadlock and are able to detect short deadlock cycle in efficient manner. However deadlock resolution is difficult task for distributed algorithms because many sites involved in detection may detect same deadlock due to unawareness of presence of each other.

A. B. M. Alom Algorithm

B. M. Alom in 2009 [19] has proposed an algorithm which uses concept of LTS (Linear Transaction Structure), DTS (Distributed Transaction Structure) and priorities to detect and resolve deadlock. This algorithm detect local as well as global deadlock. This algorithm comprises of two tables, first table consist of LTS or DTS which maintains the list of transactions requesting data items to other transactions while the second table consist of list of transaction ID and corresponding priority of that transaction. Whenever deadlock cycle exists in WFG, priorities of the transactions which are part of deadlock cycle are examined. Transaction having the lowest priority is aborted so that the data items held by aborted transaction becomes available for transactions in waiting state in order to resolve

deadlock. But disadvantage of this algorithm is that if order of priorities [20] gets changed then this algorithm fails to detect deadlock.

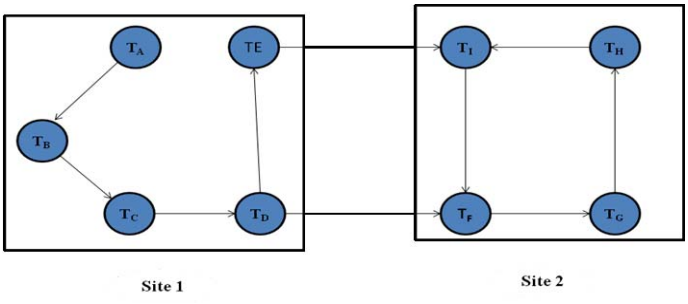

Fig. 4 WFG with cycle as well as with deadlock free transactions

Consider Fig. 4 which consists of two sites. Transactions T_A, T_B, T_C, T_D, T_E are running on site 1 while Transactions T_F, T_G, T_H, T_I are running on site 2.

1) Detection and Resolution of Local Deadlock

LTS [21][22] is used to detect deadlock at each site locally [23]. If there is an edge from T_X to T_Y { $T_X \rightarrow T_Y$ } such that T_X is requesting data item held by T_Y then corresponding entry of T_X and T_Y is done in their LTS as follow by considering figure 4:

TABLE II

LTS 1 FOR SITE 1

T_X	T_Y
T_A	T_B
T_B	T_C
T_C	T_D
T_D	T_E

TABLE III

PRIORITIES OF TRANSACTIONS AT SITE 1

Transaction ID	Priority
A	1
E	2
B	3
D	4
C	5

TABLE IV

LTS 2 FOR SITE 2

T_X	T_Y
T_I	T_F
T_F	T_G
T_G	T_H
T_H	T_I

TABLE V

PRIORITIES OF TRANSACTIONS AT SITE 2

Transaction ID	Priority
G	1
F	2
H	3
I	4

In Table III and V, lower number is representing lower priority i.e. number 1 represents lowest priority. In LTS 1 (Table II) there is no deadlock cycle so all transactions at site 1 are deadlock free but in LTS 2 (Table IV) there is deadlock cycle as { $T_I \rightarrow T_F$, $T_F \rightarrow T_G$, $T_G \rightarrow T_H$, $T_H \rightarrow T_I$} so in order to make site 2 as deadlock free site, transaction T_G is aborted as it is having the lowest priority i.e. transaction pair { $T_G \rightarrow T_H$} is aborted.

2) Detection and Resolution of Global Deadlock

DTS [21][22] is used to detect global deadlock [23] comprising of transactions from several sites. If there is an edge from T_X to T_Y { $T_X \rightarrow T_Y$ } such that T_X is requesting data item held by T_Y then corresponding entry of T_X and T_Y is done in their DTS as follow by considering figure 4:

TABLE VI

DTS FOR SITE 1 AND SITE 2

T_X	T_Y
T_D	T_E
T_E	T_I
T_I	T_F
T_F	T_D

TABLE VII

PRIORITIES OF TRANSACTIONS WHICH ARE PART OF DEADLOCK

Transaction ID	Priority
F	1
E	2
I	3
D	4

In Table VII, lower number is representing lower priority i.e. number 1 represents lowest priority. In DTS for Site1 and Site2 (Table VI), there is global deadlock cycle as { $T_D \rightarrow T_E$, $T_E \rightarrow T_I$, $T_I \rightarrow T_F$, $T_F \rightarrow T_D$ } so in order to make a deadlock free distributed system, T_F is aborted as it has the lowest priority i.e. transaction pair {$T_F \rightarrow T_D$} is aborted.

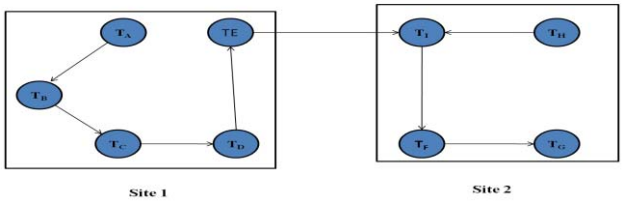

Fig. 5 Deadlock free Distributed System

B. *Edge-Chasing Algorithm*

Algorithms of this class use special messages named probes [24][25] to detect cycle in WFG. Whenever a transaction is in blocked state i.e. waits for another transaction to release a lock on data item a probe is issued. Probe message gets propagated only for blocked transactions along edges of graph and is sent to all transactions on which blocked transaction depends. Whenever active transaction receive any probe then it may discard probe. When blocked transaction receives probe then path information in probe gets updated and that updated probe again starts propagating. Finally if probe arrives at transaction which owns the lock then no deadlock exists and if probe comes back to the transaction that initiated probe communication then deadlock cycle exists.

For example: If transaction T_A is blocked [26] then probe will be issued and sent to all transactions on which T_A depends. If probe comes back to T_A then deadlock exists and then detected deadlock cycle is resolved by aborting one of transactions in deadlock cycle based on priorities or time stamping of transactions. The main drawback of this algorithm is that it fails to detect deadlock when the initial transaction is not the part of deadlock cycle.

V. CONCLUDING REMARKS

Deadlock is one of the most common problems in DDS and as deadlock adversely affects overall performance of DDS so it is essential to detect and resolve deadlock so that DDS could continue to work efficiently. In this paper we have analyzed B. M. Alom algorithm and edge chasing algorithm which are distributed algorithms. B. M. Alom Algorithm detects and resolves deadlock correctly if we don't change order of priority i.e. for detecting and resolving deadlock problem same as Fig. 4 we have to take same order of priority as presented in this paper. It detects local as well as global deadlock by constructing LTS and DTS respectively.

While Edge-Chasing Algorithm Detects and resolves deadlock by using updated WFG and sending it to neighbor transaction along the edges of graph. Probes are messages of fixed length. It doesn't detect false deadlock. The main drawback of this algorithm is that it fails to detect deadlock when the initial transaction which initiated probe propagation is not the part of deadlock cycle.

The proposed algorithms we discussed in our paper have some drawbacks like B.M. algorithm has priority problem while Edge-Chasing algorithm fails to detect deadlock if initial transaction or process which initiated probe propagation is not part of deadlock cycle. These drawbacks of proposed algorithms prove to be reasonable for the necessity of designing new algorithms for removing drawbacks of proposed algorithms.

Apart from aborting transactions based on priorities we can also use the time stamping notion for aborting transaction in the years to come. In time stamping the transaction which is having greater timestamp is considered as younger transaction and that younger transaction is aborted to make the system deadlock free.

V1 REFERERENCES

[1] Priyadarshini, Prof. Dr. S. S. Sane, Rutuja Jadhav, "Deadlock Detection in Distributed Database," International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 2, Issue 2, ISSN 2278-6856, March – April 2013.

[2] http://www.scribd.com/doc/18110136/Distributed-Database-Management-Notes-2#scribd

[3] T. Anthony Marsland and Sreekaanth S. Isloor, "Detection of Deadlocks in Distributed Database Systems*!," Technical Papers, INFOR vol. 18, no. 1, February 1980.

[4] Johann Gamper,"Distributed Databases Chapter:1 Introduction," DDB 2008/09.

[5] https://en.wikipedia.org/wiki/Distributed_database

[6] M. Tamer Özsu, Patrick Valduriez "Principles of Distributed Database Systems," London Springer, Third Edition: 2011.

[7] https://en.wikipedia.org/wiki/Deadlock

[8] R.C. Holt, "Some deadlock properties of computer systems," ACM Computing Surveys, pp. 179-195 Sept. 1972.

[9] http://www.eazynotes.com/pages/database-management-system/concurrency-control.html

[10] https://technet.microsoft.com/en-us/library/ms175519(v=sql.105).aspx

[11] Ashutosh Kumar Dubey, "Database Management Concepts," First Edition: September 2010.

[12] http://exploredatabase.blogspot.in/2014/04/deadlock-detection-technique-in-database.html

[13] http://www.cs.uni.edu/~fienup/cs146s01/in-class-overheads-and-activitie/lecture9.lwp/odyframe.htm

[14] Edgar Knapp, "Deadlock Detection in Distributed Databases," ACM Computing Surveys, Vol. 19, No. 4, December 1987.

[15] Mukesh Singhal,"Deadlock Detection in Distributed Systems," Ohio State University, October 1989.

[16] Ajay Kshemkalyani and Mukesh Singhal, "Deadlock Detection in distributed Sysrems," Distributed Computing: Principles, Algorithms, and Systems Chapter 10.

[17] PEI-YU LI, "The formal description of resource deadlock in distributed systems," A dissertation presented to the faculty of the graduate School of the university of MISSOURI-ROLLA, 1962.

[18] Distributed deadlocks Assignment T3. Time and Global state TDT4190 - Distributed systems, spring 2014.

[19] Abdullah Mohammed Rashid, Nor ashikin Ali, "Deadlock Detection and Resolution in Distributed Database Environment," International Journal of Scientific and Research Publications, Volume 5, Issue 9, September 2015.

[20] Sailen Dutta Kalita, Minakshi Kalita, Sonia Sarmah, "A Survey on Distributed Deadlock Detection Algorithm and its performance Evolution," IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 4, April 2015.

[21] Swati Gupta , Meenu Vijarania, "Analysis for Deadlock Detection and Resolution Techniques in Distributed Database," International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 7, July 2013.

[22] Himanshi Grover, Suresh Kumar, "Analysis Of Deadlock Detection and ResolutionTechniques in Distributed Database Environment," International Journal of Computer Engineering & Science, Sept 2012.

[23] Swati Gupta, "Deadlock Detection Techniques in Distributed Database System," International Journal of Computer Applications (0975 – 8887) Volume 74– No. 21, July 2013.

[24] Gupta Dhiraj, Gupta V.K., "Approaches for Deadlock Detection and Deadlock Prevention for Distributed systems," Research Journal of Recent Sciences Vol. 1 (ISC-2011), 422-425 2012.

[25] Md. Abdur Razzaque, Md. Mamun-Or-Rashid, Choong Seon Hong, "MC2DR: Multi-cycle Deadlock Detection and Recovery Algorithm for

Distributed Systems," R. Perrott et al. (Eds.): HPCC 2007, LNCS 4782, pp. 554–565, 2007.

[26] https://en.wikipedia.org/wiki/Edge_chasing

[27] R. Obermarck, "Distributed Deadlock Detection Algorithm," ACM Transaction on Database Systems,

vol. 7:2, pp. 187-208, 1982.

[28] Chandy X. M. and Mishra J., "A Distributed Algorithm for Detecting Resource Deadlocks in Distributed Systems " in ACM, 1982.

[29] 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.