Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 8; April-June, 2015 pp. 63-65
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/acsit.html

A Review on Deadlock Detection in Distributed Database

Parul Tomar[1] **and Mohit Bhardwaj**[2]

[1,2](YMCA University Faridabad Haryana/ India)

Abstract—*These days most ofthe organizations are using distributed databases. But there are many concerns attached with these databases, for example concurrency control, deadlock, multiuser access etc. So in this paper we give a detailed study of deadlocks including detection, prevention and their removal from the system. This paper also discusses various detection schemes available in literature.*

Keywords*: Deadlock Detection, Distributed databases*

1. INTRODUCTION

Database, a collection of information is most important part of this software industry. All the big organization relies on it. These databases are managed by various database management tools such as Hierarchical DBMS, Network DBMS, Relational DBMS, and Object-oriented DBMS. If we divide databases in two categories then it could be, first distributed databases and second non-distributed databases. Non distributed databases are on a single site and can be accessed from there only. But in case of distributed databases, they are distributed over various sites and accessed from anywhere.

When it comes to distributed database [2], deadlock is the biggest concern these days, as databases are accessed from various sites and demand and allocation of resources may lead to a deadlock in distributed database. When we make some changes in data of one site we are required to make same changes on other site where same data is present. For this one site sends message to other sites.Also, the request for data resources from different sites at the same time can lead to a situation where none of the request can be fulfilled. This Situation is known as deadlock. Further Sections discussed the deadlock and its phases in detail Section 2.0 will give brief detail about deadlock situation in a system. In section 2.1 we discussed about deadlock prevention in a system. Section 2.2 deals with deadlock detection technique in distributed and non distributed databases. Section 2.3 will tell you how to remove a deadlock it occurs in a system. In section 3 various deadlock detection algorithms has discussed along with their advantages and disadvantages. In section 5 conclusion of this paper is given. And in section 5 we have references.

2. DEADLOCK

A deadlock is a state where some processes request for some resources but those resources are held by some other processes.For example, if there are 4 Process A, B, C and D. Process A is waiting for Process B to get completed and Process B is waiting for Process C to get completed, Process C is waiting for Process D to get completed and Process D is waiting for Process A to get completed. This situation created a cyclic wait for the four transactions and resulted in a Deadlock.

If we see this scenario in a Transactional Database, suppose there are two processes A and B. and Process A wants to update Row 1 and then row 2. On the same time Process B wants to update Row 2 then Row 1, it creates a cyclic wait for the Process A and Process B and results in a deadlock.

There are three main techniques generally used to handle a deadlock. Deadlock prevention, deadlock detection and deadlock removal. We are discussing these three techniques below

2.1 Deadlock Prevention

This technique prevents the system from making any deadlock. In this technique information of the all resources which are allocated to some process is recorded. Now if a process requests for some resources, the system will grant only when all the resources are available. System will make sure that not a single resource which is requested is required or hold by some other process. In Simple words we can say that the resources are reserved in advance.

As deadlock Prevention technique prevents the system from making any deadlock so there is no overhead of any rollback. For the systems which as no restoring option this is the only solution to handle deadlock. There are two main problems associated with the technique first is, it reduces the concurrency the system, as the resources will only be allocated to a process only when all the resources requested are free. Second is, Overhead of the record containing information about the allocated and free resources with the system.

2.2 Deadlock Detection

This technique is used to detect existing deadlock in the system. When Resource allocation is fair and processes holding and waiting for resources results in deadlock. When a deadlock occurs it should be detected and resolved as soon as possible for good efficiency of the system. A deadlock can be distributed deadlock, when a process holds or wait for the resources on some other sites and undergo in a deadlock situation such deadlocks are known as distributed deadlock, these deadlock not only affects the home site of the process but also other sites so these deadlocks must be detected and resolved as soon as possible. There are various deadlock detection algorithms which are discussed in this paper.

2.3 Deadlock Removal

When a deadlock is detected in a system, it must be removed by terminating some process. During removal we must provide the roll back facility to the terminated process. There are variousstrategies for deadlock removal such as time stamping, youngest process removal, priority based removal etc.

3. DEADLOCK DETECTION IN DISTRIBUTED DATABASE

Various deadlock detection algorithms published in last few years. For being a good deadlock detection algorithm it must satisfy two main criteria: first is no false deadlock detection and second is no undetected deadlock, all the deadlocks must be detected in a finite time. Literature review of various papers is given below.

3.1 Obermarck's algorithm

In this algorithm one site gets wait for graph information from other site, information received by the site is combined with the local wait for graph and global deadlock presence is checked. And it breaks the only cycle which does not contain "External node" (External node is at the local site which represents TWFG which is on some other site.) For Each Cycle which contains External nodex ->A->B->C->x sends a "string" message to the site T_n if transaction id of T1 is greater than $T_{n.}$

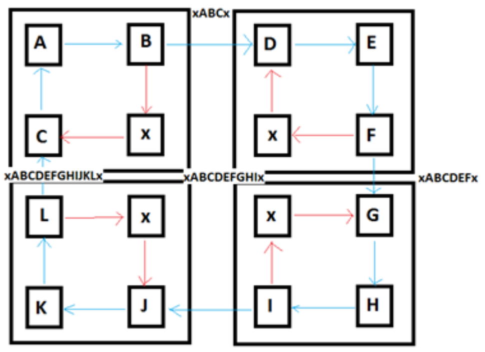

Figure 1 : Flase Deadlock Detection in Obermarck's algorithm

.In this way it reduces the number of messages sent; this is the main advantage of this algorithm. But sometimes this algorithm may detect false deadlock as TWFG constructed do not represent a snap shot of global TWFG at any instant.

3.2 ChandyMisraHaas algorithm. [5]

In this algorithm a probe message is used which sent from one site to another for deadlock detection. Suppose deadlock detection for Process P1. Home site of P1 has P2, P3, and P4 in WFG as P1->P2->P3->P4. And P4 is waiting for P5 which is at site 2. Now the probe message (P1, P4, P5) will be sent to site 2. Now site 2 has its local WFG as P5 - > P6 -> P7 -> P8 and P5 -> P6 -> P9. And P8 is waiting for P10 (at site 3) and P9 is waiting for P11 (at site 3). Messages sent at site 3 by site 2 are (P1, P8, P10) and (P1, P9, P11). Now at site 3 Local Wait for graphs are P10 -> P12 and P11 -> P13 -> P14. Suppose P12 is waiting for P1 (at site 1) and P14 is not waiting for any other process to get completed so this probe will lead no deadlock but When site 3 will send other message i.e. (P1, P12, P1) it will result in a deadlock. Main advantage of this algorithm is: it not only gives the information about the deadlock initiator but also about the deadlock path and number of messages is reduced.

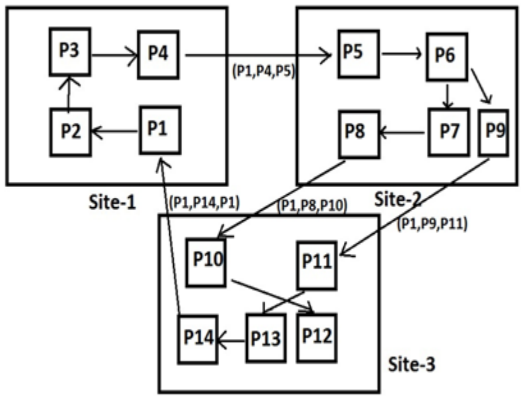

Figure 2 : Chandy Misra Haas algorithm Example

3.3 Michael's algorithm. [6]

In this algorithm we have LTS and DTS which stores local transaction information and distributed transaction information respectively. For a local deadlock we scan LTS and for global deadlock we scan DTS. If any transaction requests a resource held by another transaction on the same site then this data is storedin the linear transaction structure (LTS). Distributed Transaction Structure (DTS) containsthose transactions which are interconnected from one site to another site. Intra requests DTS is managed by Data Manager (DM).To detect a local deadlock the linear transaction structure (LTS) of that local site is checked. For a global deadlock DTS is checked. If a deadlock cycle is detected. Transaction having low priority is aborted or the youngest transaction is aborted. This algorithm detects local and global both types of deadlocks. And it is very simple to implement.

3.4 Ho’s Algorithm. [7]

In is algorithm transaction table at each site has a resource tablewhere information regarding the transactions holding and waiting for local resources is maintained. We choose a central site where periodic controller scans the resource table periodically and detects deadlock if exists. Steps involved in deadlock detection by periodic controller are:

1) Controller requests all sites to send their resource table.
2) Using the data of all tables received by the controller, it creates a wait for graph.
3) If a cycle is detected, it reports a deadlock.
4) Now the steps of deadlock removal are taken.

If there are M Number of sites then 4m messages will be required to detect a deadlock.

3.5 B M Alom Algorithm [8]

This is a deadlock detection and deadlock removal algorithm using the concept of priorities. In a table list of all transactions are maintained with their priority. Deadlock is detected by making Wait for graph by the list of transactions in the table using their priority. To make system deadlock free, the transaction with the least priority is aborted and resources held by this transaction becomes free and allotted to other waiting transactions. But there is a problem associated with this algorithm, if there is any change in the priority of the deadlock then there is a chance that it may fails to detect any deadlock.

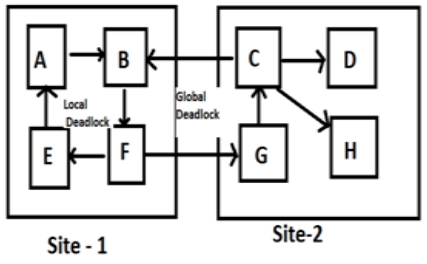

Figure 3 : B M Alom Deadlock Detection Algorithm

3.6 Sinha and Natrajan Algorithm [9]

It is an extension of Chandy’s Algorithm. Here a priority scheme is added to each transaction.With this algorithm we can easily determine the number of messages required to be sent in the best and worst. As in Chandy’s Algorithm a probe is sent to detect a deadlock, here the number of probes sent during the deadlock detection are reduced using the priority. A probe is only sent if the priority of the resource holder is greater than that of the resource requestor. The probe is only be propagated if the priority of the holder of the data item is greater than that of the initiator. After propagation if the Data manager receives the probe which was initiated by it, reports a deadlock. A youngest transaction is aborted to resolve the deadlock. The main advantage of this algorithm is, number of messages is reduced using the priority scheme.

4. CONCLUSION

Deadlock is very harmful for any system, it not only reduces the performance of the system but also affects the ongoing processes, so it must be detected as soon as possible and removed if detected. In this paper we have discussed various deadlock detection algorithms their techniques, advantages and drawbacks. In Obermarck’s algorithm we found that it reduces the number of messages sent during deadlock check but it sometimes give false deadlock detection. ChandyMisraHaas algorithm not only gives the information about deadlock but also about deadlock path and number of messages are also reduced in this algorithm. Sinha and Natrajan Algorithm is the extension of ChandyMisraHaas algorithm in this priority scheme is used to reduce the number of messages. B M Alom Algorithm works on LTS and DTS and deadlock is removed on the bases of the priority of the transaction. Ho’salgorithm have a table at each site which contains the transaction details, one site is periodically selected and controller requests the transaction data from all sites and checks for deadlock.

REFERENCES

[1] Wikipedia (http://en.wikipedia.org/wiki/Database)

[2] University of waterloo Lecture on Distributed Database (https://cs.uwaterloo.ca/~tozsu/courses/cs856/F02/lecture-1-ho.pdf)

[3] Distributed Deadlock DetectionBy JoAnne L. Holliday and Amr El AbbadiUniversity of California at Santa Barbara (http://www.cse.scu.edu/~jholliday/dd_9_16.htm)

[4] R. Obermarck, "Distributed Deadlock Detection Algorithm," ACM Transaction on Database Systems, vol. 7:2, pp. 187-208, 1982

[5] K.M. Chandy, J. Misra, and L.M. Haas, “Distributed Deadlock Detection,” ACM Trans. Computer Systems, May 1983, pp.144-156.

[6] Mitchell D.P. and Merritt M.J., "A Distributed Algorithm for Deadlock Detection and Resolution", AT&TBell Labs, Murray Hill, NJ 07974.

[7] Ho G.S. and Ramamoorthy C.V., "Protocols for Deadlock Detection in Distributed Database Systems", IEEETransactions on Software Engineering, Vol SE-8 No. 6, November 1982. 554-557.

[8] Alom B.M. Monjurul, Frans Alexander Henskens, Michael Richard Hannaford, Optimization of Detected Deadlock Views of Distributed Database, International Conference on Data Storage and Data Engineering , pp.44-48, ISBN: 978-0-7695-3958-4, 2010

[9] 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.