ANALYSIS FOR DEADLOCK DETECTION AND RESOLUTION TECHNIQUES IN DISTRIBUTED DATABASE

Rohan yadav, Kimmi makkar
Dronacharya college of Engineering
Dept. Information technology
Gurgaon, Haryana, India

Abstract: In a distributed database environment, where the data is spread across several sites there are many concerns to deal with such as concurrency control, deadlock. Deadlocks impact the overall performance of the system. A deadlock is a condition in a system where a process cannot proceed because it needs to obtain a resource held by another process which it itself is holding a resource that the other process needs. In literature various techniques have been discussed which are used to prevent, detect and resolve the deadlocks. In this paper we have analyzed the deadlock detection and resolution techniques that are used. We have reviewed in detail the algorithms presented by B. M. Alom[2] for deadlock detection and resolution in distributed environment and found that when we rearrange the order of the transactions pair in the algorithm by Aloms[2] then it completely fails to detect the deadlock.

INTRODUCTION

Distributed database systems (DDBS) consist of different number of sites which are interconnected by a communication network. In such a resource sharing environment the database activities can be performed both at the local and global level so if the allocation of the resource is not properly controlled than it may lead to a situation that is referred to as deadlock. In Distributed database system model, the database is considered to be distributed over several interconnected computer systems. Users interact with the database via transactions. A transaction is a sequence of activities such as read, write, lock, or unlock operations. If the actions of a transaction involve data at a single site, the transaction is said to be local, on the other hand a global transaction involve resources at several sites. A deadlock may occur when a transaction enters into wait state, i.e. when request is not granted due to non-availability of the resources as the requested resource is being held by another waiting transaction. In such a situation, waiting transaction may never get a chance to change its state. Deadlock representation techniques for their easy detection have been discussed widely in the literature and graphical representation has been found to be most suitable and effective technique. A deadlock can be indicated by a cycle in the directed graph called Wait-for-Graph (WFG) [4] that represents the dependencies among the processes. A node in the graph G represents a transaction and a directed edge from vertex i to vertex j exist in G, if Ti (Transaction i) needs a resource, which is being held by Tj (Transaction j). For example, in Fig 1 a transaction T1 has locked data item P and needs to lock item Q, T2 has locked item Q and needs to lock item P.
avoidance algorithm decides if the requesting transaction can wait or if one of the waiting transactions need to be aborted.

**Deadlock Prevention:** It is an approach that prevents the system from committing an allocation of locks that will eventually lead to a deadlock. This technique requires pre-acquisition of all locks. The transactions are required to lock the entire data item that they need before execution. Deadlock prevention deals with deadlock ahead of time.

**Deadlock Detection:** In this approach, deadlock may have already occurred and the deadlock detection technique tries to detect it and gives the process by which it can be resolved. Thus the system periodically checks for them. The existence of a directed cycle in the Wait-for-Graph indicates a deadlock. One transaction in the cycle called victim is aborted, thereby breaking the deadlock. We have analyzed in detail the algorithms presented by B. M. Alom [2] for detecting and resolving deadlocks in distributed environment. In section 2, we discuss about the transaction model. In section 3, we take up the recent work and analyze the contributions made by several researchers dealing with prevention, detection and resolving of the deadlocks. In section 4, we take an example to compare the working of techniques by B. M Alom[2] in details. In section 5, we give the concluding remark

**DISTRIBUTED TRANSACTION MODEL**

We next take up a distributed transaction model [1, 3] its general structure is shown in Fig 2. In this each node has the following modules: a Transaction Manager (TM), a Data Manager (DM), a scheduler (S), and a Transaction Process (T). The Transaction Manager (TM) present at each distributed site controls the execution of each transaction process (T). The transactions communicate with TMs, and in turn TMs communicate with Data Managers (DMs), the Data Manager, manages the actual data at each distributed site. A single TM supervises each transaction executed in the DDBMS. The transaction issues all of its database operations to its particular transaction manage

**RELATED WORK**

Different distributed deadlock detection and resolution algorithms have been proposed in the literature. In the paper we discuss the contributions of other researchers and the algorithms they have used for dealing with deadlocks. Chandy et. al. [4] used a Transaction Wait-for-Graph (TWFG) to represent the status of transaction at the local sites and probes to detect global deadlock.

**CONCLUSIONS**

Deadlocks in a distributed system drastically reduce the performance of the system and therefore have to be detected and resolved as soon as possible for the efficiency of the systems. After analyzing various techniques we have found that the technique presented by B. M. Alom (section 4.1)[2] is detecting deadlocks correctly if the priorities are taken in the same order as taken by him in his paper but if we take any other order then it detects false deadlocks. It means that there is a complete dependency on directed edges of wait for graph in the B.M Alom technique [2]. The concept of time stamping could be used to abort the younger transaction in our future work.

**References**


[13]. Nacer Farajzadeh , Mehdi Hashemzadeh, Morteza Mousakhani, Abolfazl T. Haghighat, An Efficient Generalized Deadlock Detection and Resolution Algorithm in Distributed Systems, Fifth International Conference on Computer and Information Technology (CIT'05), pp.303-309, 2005