Concept Map

CSE 707/CSE449
Group Task 3

22241167 - Fahmida Ahmed
22273001 - Sushana Islam Mim
22273008 - Mirza Ahmad Shayer

Principles of Distributed Database Systems, Third Edition
M Tamer Oszu and Patrick Valduriez

Chapter 11
Distributed Concurrency Control

Chapter 11 Distributed Concurrency Control

In this chapter two assumptions are made, the distributed system is fully reliable and no failures are experienced. Though these are unrealistic they allow us to solve the issues involving management of concurrency. Serializability is the most widely accepted corectness criterieon for concurrency control algorithms. Other types are discussed as well.

11.1 Serializability Theory

The Concurrency Control algorithms most popular accuracy criteria is serializability theory which provides the formula framework to reason about and analyze serializability and it's framework execution. Execution of transactions leaves the database in a state that can be achieved by their is exactly the point of the serializability argument which addresses serializability issues more formally.

11.2 Taxonomy of Concurrency Control Mechanisms

The most common classification criteria is the mode of batabase distribution. Optimistic algorithm synchronize the concurrent execution of transactions early in the execution of their execution life cycle, while pessimistic algorithms delay the synchronization of transactions until their termination; the currency control

11.3 Locking-Based Concurrency Control Algorithms

It is to ensure that a data item that is shared by conflicting operations is accessed by only one operation at a time. This is done by connecting a lock with each lock unit. It is set by a transaction before access and is reset by the end. A lock request is granted if a connected lock is not being held by another. The distributed DBMS manages both the locks and lock management responsibilities. The scheduler is the lock manager. The two phase states that no transaction should request a lock after it releases one of its locks. 2PL algorithms execute transactions in two phases. Locking based algorithms can cause deadlocks.

11.3.1 Centralized 2PL

The 2PL algorithm can be extended to the distributed DBMS environment. By delegating the lock management to one site only. This is known as the primary site 2PL algorithm. This is also a Centralized 2PL (C2PL) algorithm. There is also the centralized 2PL transaction management (C2PL-TM) and centralized 2PL lock management (C2PL-LM) algorithms. The one important data structure used in the algorithms is a 5-tuple. One issue with C2PL algorithms is a bottleneck may form around the central site.

11.3.2 Distributed 2PL

Distributed 2PL (D2PL) requires the availability of lock managers at each site. This algorithm is similar to C2PL-TM, with two changes. The messages sent to the central site lock manager are sent to the lock managers at all sites. The second difference, the operations are not passed to the data processors by the coordinating transaction manager but by the participating lock managers.

11.4 Timestamp-Based Concurrency Control Algorithms

A timestamp is a simple identifier that serves to identify each transaction uniquely and is used for ordering. Uniqueness is only one of the properties of timestamp generation. Two timestamps generated by the same transaction manager should be monotonically increasing. Architecturally , the transaction manager is responsible for assigning a timestamp to each new transaction and attaching this timestamp to each database operation that it passes on to the scheduler.

11.4.1 Basic TO Algorithm

The same data structures and assumptions we used for centralized 2PL algorithms apply to these algorithms as well. When an accepted operation is passed on to the data processor, the scheduler needs to refrain from sending another conflicting, but the acceptable operation to the data processor until the first is processed and acknowledged. In one sense the queue that the TO scheduler maintains may be thought of as a lock.

11.4.2 Conservative TO Algorithm

TO scheduler restarts a transaction if a younger conflicting transaction is already scheduled or has been executed. The receiving transaction managers can then compare their own counter values with that of the incoming operation. If a transaction manager does not have a transaction to process, it needs to send dummy messages periodically to every scheduler in the system, informing them that the operations that it will send in the future will have timestamps greater than that of the dummy message.

11.4.3 Multiversion TO Algorithm

Multiversion TO is another attempt at eliminating the restart overhead cost of transactions. Most of the work on multi-version TO has concentrated on centralized databases, so we present only a brief overview. However, we should indicate that the multi-version TO algorithm would be a suitable concurrency control mechanism for DBMSs that are designed to support applications that inherently have a notion of versions of database objects .

11.5 Optimistic Concurrency Control Algorithms

The previous algorithms are pessimistic in nature and assumes conflicts happen often. Optimistic algorithms delay the validation phase just before the write phase. Each transaction initially makes it updates on local copies of data items. The original optimistic proposals were based on timestamp ordering. Unfortunately, there is no optimistic algorithm for validating the mutual consistency rule globally. An advantage of these algorithms is their potential for higher levels of concurrency. However one issue is starvation though it is possible to solve this problem.

11.6 Deadlock Management

In deadlocks, since there is mutual exclusion of access to shared resources (data) and transactions may wait on locks. Algorithm that acquire the waiting of transactions may also cause deadlocks. The distributed DBMS requires special procedures to handle them. A deadlock can occur because of transactions wait for one another.

11.6.1 Deadlock Prevention

Deadlock prevention methods guarantee that deadlocks cannot occur in the first place. To perform this check, it is required that all of the data items that will be accessed by a transaction be predeclared. Access to certain data items may depend on other conditions that may not be resolved until run time.

11.6.3 Deadlock Detection and Resolution

The simplest means of avoiding deadlocks is to order the resources and insist that each process request access to them in that order. Another alternative is to make use of transaction timestamps to prioritize transactions and resolve deadlocks by aborting transactions with higher (or lower) priorities.

11.6.2 Deadlock Avoidance

There are three fundamental methods of detecting distributed deadlocks. Deadlock detection and resolution is the most popular and best-studied method. The problem of selecting the minimum total-cost set for breaking the deadlock cycle has been shown to be a difficult (NP-complete) problem.

11.7 “Relaxed” Concurrency Control

There have been studies that relax serializability in arguing for the correctness of concurrent execution, and consider other transaction models, primarily nested ones.

11.7.1 Non-Serializable Histories

The solution is to observe and exploit the «semantics» of these transactions. The concept of compatibility is refined by Lynch and several levels of compatibility among transactions are defined. Intuitively, a relatively consistent history is equivalent to a historythat is stepwise serial , and in which a step of transaction Ti interleaves two consecutive steps of transaction Tj only if transactions of Ti’s type are allowed to interleave Tjm at its breakpoint.

11.7.2 Nested Distributed Transactions

The concurrent execution of nested transactions is interesting, especially since they are good candidates for distributed execution. From the perspective of ACID properties, closed nested transactions relax durability since the effects of successfully completed subtransactions can be erased if an ancestor transaction aborts.

11.8 Conclusion

This chapter discussed about distributed concurrency control algorithms that provide the isolation and consistency properties of transactions. It is one of the fundamental components of distributed DBMS. This chapter discussed both hardware and software components of the computer systems.

11.9 Bibliographic Notes

There are a number of papers that discuss the results of performance evaluation studies on distributed concurrency control algorithms. The research activity on this topic has slowed down in the last years. Some of the recent relevant papers are [Yeung and Hung, 1995; Hofri, 1994; Lee and Kim, 1995; Kshemkalyani and Singhal, 1994; Chen et al., 1996; Park et al., 1995] and [Makki and Pissinou, 1995].