Proficient Method towards Concurrency Control in Distributed Database

Anil Ahir
Sr. Lecturer, Babasaheb Gawde Institute of Technology, Mumbai, India

Abstract—Majority of the research in database management system focuses primarily on centralized database systems. However, with the demand for higher performance and higher availability, database systems have moved from centralized to distributed architectures. As the development and maturity of the popular centralized database system moves towards the distributed approach, the challenges and roles start becoming more complex and complicated. The discussion revolves around the variety of protocols, their working, their advantages and their disadvantages in a distributed environment. Concurrency control is an integral part of database systems. In literature various techniques have been discussed which are used to prevent, detect and resolve the deadlocks. In this report I have analyzed various concurrency control protocol and the deadlock detection and resolution techniques.

Keywords: Timestamp, classification, threshold, testing.

1. INTRODUCTION

The concurrency control in distributed DBMS’s is an important research problem. In a distributed database system, data in the system can be accessed by different users at the same time. If these concurrent accesses are not controlled properly, the database will become inconsistent during multiple concurrent updates to the database. Such inconsistencies may give rise to several problems. The term ‘Distributed’ in the concept of ‘Distributed Databases’ clearly means the environment of the database system that it is not at one place or location but exist in at physically autonomous locations with some interconnection in the logical plan. The ‘Distributed Database’ can be thought of as a collection of multiple, logically interrelated databases distributed over a computer network in such a way the distribution is clear to the users. The fact that the database is located at multiple sites obviously means that it promotes availability of data to multiple users concurrently without losing the integrity and accuracy of the database.

Distributed database as equated to the Centralized database varies mainly in the way the data is truly stored and located. Hence the data is not present at one place it shows the concept of Fragmentation and Replication[9]. Several issues could be generated if concurrency of transaction is not handle technically For example, it might lead to the lost update problem in a funds transfer transaction. It may be, however, almost impossible to execute a task, such as funds transfer usually consisting of a sequence of several atomic operations, without temporarily violating the integrity constraints. However these atomic operations can be grouped into time units consistency called transactions. Thus if a transaction start with a consistent database and the transaction is run to its completion, it is guaranteed to produce a consistent database.

2. DISTRIBUTED DATABASE SYSTEM

2.1 Summary on Distributed database Systems

A distributed database management system (DDBMS) manages the database as if it were all stored on the same computer. A distributed database is a database in which shares of the database are stored on multiple computers within a network. Users have admittance to the share of the database at their location so that they can access the data relevant to their tasks without interfering with the work of others. The DDBMS synchronizes all the data periodically and, in cases where multiple users must access the same data, ensures that updates and deletes performed on the data at one location will be automatically reflected in the data stored elsewhere.

FIG. 1 Distributed database conceptual diagram

2.2 Serializability

When more than one transaction is executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transaction are interleaved with some other transaction. Several issues could be generated if concurrency of transaction is not handled technically. For example, it might lead to the lost update problem in a funds transfer transaction. It may be, however, almost impossible to execute a task, such as funds transfer usually consisting of a sequence of several atomic operations, without temporarily violating the integrity constraints. However these atomic operations can be grouped into time units consistency called transactions. Thus if a transaction starts with a consistent database and the transaction is run to its completion, it is guaranteed to produce a consistent database.
2.3 Deadlock
In computer science, deadlock refers to a specific condition when two or more processes are each waiting for another to release a resource, or more than two processes are waiting for resources in a circular chain (see Necessary conditions). Deadlock is a common problem in multiprocessing where many processes share a specific type of mutually exclusive resource known as a software, or soft, lock. Computers intended for the time-sharing and/or real-time markets are often equipped with a hardware lock (or hard lock) which guarantees exclusive access to processes, forcing serialization. Deadlocks are particularly troubling because there is no general solution to avoid (soft) deadlocks.[9]

This situation may be understood by an analogy with two people who are drawing diagrams, with only one pencil and one ruler between them. If one person takes the pencil and the other takes the ruler, a deadlock occurs when the person with the pencil needs the ruler and the person with the ruler needs the pencil, before he can give up the ruler. Both requests can't be satisfied, so a deadlock occurs.[9]

2.3.1 Deadlock prevention
Deadlocks can be prevented by ensuring that at least one of the following four conditions occur:

- Removing the mutual exclusion condition means that no process may have exclusive access to a resource. This proves impossible for resources that cannot be spooled, and even with spooled resources deadlock could still occur. Algorithms that avoid mutual exclusion are called non-blocking synchronization algorithms.

- The "hold and wait" conditions may be removed by requiring processes to request all the resources they will need before starting up (or before embarking upon a particular set of operations); this advance knowledge is frequently difficult to satisfy and, in any case, is an inefficient use of resources. Another way is to require processes to release all their resources before requesting all the resources they will need. This too is often impractical. (Such algorithms, such as serializing tokens, are known as the all-or-none algorithms.)

- A "no preemption" (lockout) condition may also be difficult or impossible to avoid as a process has to be able to have a resource for a certain amount of time, or the processing outcome may be inconsistent or thrashing may occur. However, inability to enforce preemption may interfere with a priority algorithm. (Note: Preemption of a "locked out" resource generally implies a rollback, and is to be avoided, since it is very costly in overhead.) Algorithms that allow preemption include lock-free and wait-free algorithms and optimistic concurrency control.

- The circular wait condition: Algorithms that avoid circular waits include "disable interrupts during critical time units", and "use a hierarchy to determine a partial ordering of resources" (where no obvious hierarchy exists, even the memory address of resources has been used to determine ordering) and Dijkstra’s solution.[9]

3. Concurrency Control Techniques For Distributed Database

3.1 Priority Based Locking
In the proposed protocol we assume that the database is fully replicated. The system consists of N number of nodes consisting of different transactions. Each node sends its read/write request to a global node. Global node assigns the requested lock to the transaction having the higher priority. A waiting queue is maintained in which the transactions waiting for a lock are placed. First, we find the total execution time of the transactions performing read/write operation on the basis of timestamp based priority. Then we find the total execution time of transactions on the basis of read transactions being assigned a higher priority. Then we compare the results from both the cases.
and the values of the write set are written in the database at the global node. The values of the write set are transmitted by the global node to all other nodes. Each node receives the new write set and updates the database; then an acknowledgment is sent back to the global node. When the global node receives acknowledgments from all other nodes in the system, it knows that transaction T1 has been completed at all nodes. The global node releases the locks and starts processing the next transaction [7].

3.2 Threshold and Classification Based Locking
- All the transaction is classified according to its execution time.
- Classification is done on basis of Threshold Time value (T)
- Those transactions satisfy the T will group into once.
- Groups are classified on the basis of T value, every next group will follow 2 Times the T value of its successor group
- All transaction will be grouped according to its execution time.
- Classification will be on until all transaction is grouped.
- Once the classification is done the scheduler now keep track on ET, An ET is an threshold time given to any group for execution.
- After expiry of ET a next group is selected.
- If no transaction left, ET restart and next group is selected. Until no transaction left in any group

4. Computing Execution Time on the Basis of Threshold and Classification.
Consider an Example to test this algorithm

T1(r)=2time_unit,
T2(r)=2time_unit,T3(w)=3time_unit,T4(r)=3time_unit,
T5(r)=4time_unit, T6(w)=5time_unit, T7(r)=6time_unit,
T8(r)=6time_unit,T9(w)=2time_unit,T10(r)=3time_unit

According to algorithm
We need to select T value
Let Value of T=2
- Scheduler will now classify the above transaction according to T, where T=2time_unit.
Hence all above transaction will now follow the below group
- First Group=T1,T2,T9.........less than 2 time_unit’s (T)
- Time_unitond Group=T3,T4,T5,T10-----more than 2 and less than 4time_unit’s (2T)
- Third Group=T6,T7,T8---------more than 4 and less than 8time_unit’s (4T)

Now ET need to be select, Let ET time is 6 time unit
Therefore now for execution, each group will be executed not more than 6 Time_units.
Within ET time limit if no transaction left in a group new transaction from higher adjacent group will be selected.
While executing Read transaction, the read process can run parallel hence more than one read transaction can be executed, hence when multiple read instruction are executed , the transaction with higher execution time is taken under consideration.

For First ET cycle.
From Group A & B T1+T2+T9+T3=5time_unit

<table>
<thead>
<tr>
<th>T class (&lt; =2 Time_unit)</th>
<th>2T class (&lt;=4 and &gt;2 Time_unit)</th>
<th>4T class (&lt; =8 and &gt;4 Time_unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP A</td>
<td>GROUP B</td>
<td>GROUP C</td>
</tr>
<tr>
<td>T1 Read 2tu</td>
<td>T3 Write 3tu</td>
<td>T6 Write 5tu</td>
</tr>
<tr>
<td>T2 Read 2tu</td>
<td>T4 Read 3tu</td>
<td>T7 Read 6tu</td>
</tr>
<tr>
<td>T9 Read 2tu</td>
<td>T5 Read 4tu</td>
<td>T8 Read 6tu</td>
</tr>
<tr>
<td>T10 Read 3tu</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

At first ET cycle T1,T2,T9 are read instruction hence they are executed parallel with higher time unit that is 2, hence all 3 transaction will execute in 2time_unit hence, still time_unit remain in ET1 therefore higher adjacent group is selected and T3 which is write instruction all fall below remain ET1 time. Hence T3 is executed separately. Later 1 time_unit remain and next instruction in group B is of 3 time_unit hence ET1 is terminated

For second ET cycle.
From Group B, T4+T5+T10=4time_unit

<table>
<thead>
<tr>
<th>T class (&lt; =2 Time_unit)</th>
<th>2T class (&lt;=4 and &gt;2 Time_unit)</th>
<th>4T class (&lt; =8 and &gt;4 Time_unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP A</td>
<td>GROUP B</td>
<td>GROUP C</td>
</tr>
<tr>
<td>T1 Read 2tu</td>
<td>T3 Write 3tu</td>
<td>T6 Write 5tu</td>
</tr>
<tr>
<td>T2 Read 2tu</td>
<td>T4 Read 3tu</td>
<td>T7 Read 6tu</td>
</tr>
<tr>
<td>T9 Read 2tu</td>
<td>T5 Read 4tu</td>
<td>T8 Read 6tu</td>
</tr>
<tr>
<td>T10 Read 3tu</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

At this cycle T4, T5 and T10 are executed parallel because they are read instruction ant T5 has higher tu value among them of 4tu. Therefore no other transaction left in same group and 2 time_unit left in ET2 hence higher adjacent group is selected but no transaction available less than 2 time_unit hence ET2 is terminated

For third ET cycle.
From Group C, T6=5time_unit

<table>
<thead>
<tr>
<th>T class (&lt; =2 Time_unit)</th>
<th>2T class (&lt;=4 and &gt;2 Time_unit)</th>
<th>4T class (&lt; =8 and &gt;4 Time_unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP A</td>
<td>GROUP B</td>
<td>GROUP C</td>
</tr>
<tr>
<td>T1 Read 2tu</td>
<td>T3 Write 3tu</td>
<td>T6 Write 5tu</td>
</tr>
<tr>
<td>T2 Read 2tu</td>
<td>T4 Read 3tu</td>
<td>T7 Read 6tu</td>
</tr>
<tr>
<td>T9 Read 2tu</td>
<td>T5 Read 4tu</td>
<td>T8 Read 6tu</td>
</tr>
<tr>
<td>T10 Read 3tu</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
At this Cycle of ET only T6 which is write instruction is executed isolated, hence no other transaction falls in remaining time of ET3

For Fourth ET cycle
From Group C, T7+T8=6 time_unit

<table>
<thead>
<tr>
<th>T class (&lt;=2 Time_unit)</th>
<th>2T class (&lt;=4 and &gt;2 Time_unit)</th>
<th>4T class (&lt;=8 and &gt;4 Time_unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP A</td>
<td>GROUP B</td>
<td>GROUP C</td>
</tr>
<tr>
<td>T1 Read 2tu</td>
<td>T3 Write 3tu</td>
<td>T6 Write 5tu</td>
</tr>
<tr>
<td>T2 Read 2tu</td>
<td>T4 Read 3tu</td>
<td>T7 Read 6tu</td>
</tr>
<tr>
<td>T9 Read 2tu</td>
<td>T5 Read 4tu</td>
<td>T8 Read 6tu</td>
</tr>
<tr>
<td>T10 Read 3tu</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All 10 transaction is executed in ET1+ET2+ET3+ET4=5+4+5+6= 20 Time_unit.

CONCLUSION
If the same example is to executed on basis of time priority 3.1 above shown it would take 25 Time_unit to finish

REFERENCES


