Distributed Database Systems and Data Warehouses

Dr. Volodymyr Sokol
(vlad.sokol@gmail.com)

LECTION 4

Distributed Concurrency Control - Serializability

- If the schedule of transaction execution at each site is serializable, then the **global schedule** (the union of all local schedules) is also serializable provided local serialization orders are identical. This requires that all subtransactions appear in the same order in the equivalent serial schedule at all sites
- Thus, if the subtransaction of T_i at site S_1 is denoted T_i^1 , we must ensure that if $T_i^1 < T_j^1$ then:
- $T_i^x < T_j^x$ for all sites S_x at which T_i and T_j have subtransactions

Distributed Concurrency Control - Serializability

- The solutions to concurrency control in a distributed environment are based on the two main approaches of locking and timestamping
- Thus, given a set of transactions to be executed concurrently, then:
  - locking guarantees that the concurrent execution is equivalent to *some* (unpredictable) serial execution of those transactions
  - timestamping guarantees that the concurrent execution is equivalent to a *specific* serial execution of those transactions, corresponding to the order of the timestamps

Locking Protocols

- We present the following protocols based on two-phase locking (2PL) that can be employed to ensure serializability for distributed DBMSs:
  - centralized 2PL
  - primary copy 2PL
  - distributed 2PL
  - majority locking

Centralized 2PL

- With the centralized 2PL protocol there is a single site that maintains all locking information. There is only one scheduler, or *lock manager*, for the whole of the distributed DBMS that can grant and release locks.
- The centralized 2PL protocol for a global transaction initiated at site S_1 works as follows:
  - (1) The transaction coordinator at site S1 divides the transaction into a number of subtransactions, using information held in the global system catalog. The coordinator has responsibility for ensuring that consistency is maintained. If the transaction involves an update of a data item that is replicated, the coordinator must ensure that all copies of the data item are updated. Thus, the coordinator requests exclusive locks on all copies before updating each copy and releasing the locks. The coordinator can elect to use any copy of the data item for reads, generally the copy at its site, if one exists
  - (2) The local transaction managers involved in the global transaction request and release locks from the centralized lock manager using the normal rules for two-phase locking
  - (3) The centralized lock manager checks that a request for a lock on a data item is compatible with the locks that currently exist. If it is, the lock manager sends a message back to the originating site acknowledging that the lock has been granted. Otherwise, it puts the request in a queue until the lock can be granted.

Primary copy 2PL

- This protocol attempts to overcome the disadvantages of centralized 2PL by distributing the lock managers to a number of sites
- Each lock manager is then responsible for managing the locks for a set of data items
- For each replicated data item, one copy is chosen as the **primary copy**; the other copies are called **slave copies**
- The choice of which site to choose as the primary site is flexible, and the site that is chosen to manage the locks for a primary copy need not hold the primary copy of that item

Distributed 2PL

- This protocol again attempts to overcome the disadvantages of centralized 2PL, this time by distributing the lock managers to every site
- Each lock manager is then responsible for managing the locks for the data at that site
- If the data is not replicated, this protocol is equivalent to primary copy 2PL
- Otherwise, distributed 2PL implements a Read-One-Write-All (ROWA) replica control protocol

Majority locking

- This protocol is an extension of distributed 2PL to overcome having to lock all copies of a replicated item before an update
- Again, the system maintains a lock manager at each site to manage the locks for all data at that site
- When a transaction wishes to read or write a data item that is replicated at *n* sites, it must send a lock request to more than half of the *n* sites where the item is stored
- The transaction cannot proceed until it obtains locks on a majority of the copies
- If the transaction does not receive a majority within a certain timeout period, it cancels its request and informs all sites of the cancellation. If it receives a majority, it informs all sites that it has the lock.

Timestamp Protocols

- The objective of timestamping is to order transactions globally in such a way that older transactions – transactions with *smaller* timestamps – get priority in the event of conflict. In a distributed environment, we still need to generate unique timestamps both locally and globally
- Clearly, using the system clock or an incremental event counter at each site would be unsuitable. Clocks at different sites would not be synchronized; equally well, if an event counter were used, it would be possible for different sites to generate the same value for the counter