CSCI235 Database Systems

Transaction Processing in Distributed Database Systems

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Transaction Processing in Distributed Database Systems

Outline

Principles

A distributed transaction accesses data stored at more than one location

Each transaction is divided into a number of subtransactions one for each site that has to be accessed

Apart from atomicity of a distributed transaction, atomicity of subtransactions must be ensured

Concurrency transparency and failure transparency must be enforced

Concurrency transparency means that the results of all concurrently processed transactions (distributed and non-distributed) must be the same as the results obtained by the same transactions processed in one of the serial orders

Failure transparency means that distributed system must provide recovery mechanisms that ensure that in the presence of failures transactions are atomic and durable

Transaction Processing in Distributed Database Systems

Outline

Distributed transaction management

In a centralized DBMS transaction manager coordinates transactions, scheduler implements a particular protocol processing of transactions, and recovery manager restores a database to a consistent state whenever it is necessary

In a distributed DBMS transaction manager, scheduler , and recovery manager exist in the local sites

Additionally each local site obtains transaction coordinator to coordinate processing of local and global transactions

A data communication component handles communications between the local sites

Distributed transaction management

Processing of global transactions is performed in the following way.

- A transaction coordinator at a site where a global transaction has been issued divides the transaction into subtransactions
- The subtransactions are sent to local sites
- A transaction coordinators at local sites manage the subtransactions
- The results from subtransactions are communicated to a transaction coordinator by data communication components

Transaction Processing in Distributed Database Systems

Outline

Distributed serializability

A concept of serializability can be extended on distributed transaction processing

A concurrent processing of distributed transactions is serializable if processing of subtransactions at local site is serializable and local serialization orders are the same

All subtransactions are processed in the same order in the equivalent serial schedule at all sites

Concurrency control in a distributed environment is based on locking or on timestamping protocol

If a distributed database is not replicated then there is only one copy of each data item then subtransactions do not need to be duplicated over many local sites

If distributed database is replicated then subtransaction must be replicated over many local sites and serialization of subtransactions must be the same in each local site

Transaction Processing in Distributed Database Systems

Outline

Locking protocols

There exists four locking protocols in distributed database systems

- Centralized 2PL
- Primary copy 2PL
- Distributed 2PL
- Majority locking

Centralized 2PL is based on the following principles

- A single site maintains all locking information, i.e. there is only one lock manager for entire distributed DBMS that can grant and release locks
- All replicated copies of data items require replication of subtransactions in different local sites
- Local transaction managers control processing of transactions at the local sites in the same way as in centralized 2PL
- Centralized lock manager checks if a request about lock on a data item is compatible with the locks already granted; if it is so lock manager grants a lock, otherwise a request about lock is put in a queue

Locking protocols

Primary copy 2PL protocol is based on the following principles

- Primary copy 2PL is an extension of centralized 2PL
- Primary copy 2PL distributes lock managers over the local sites
- For each replicated data item, one copy is chosen as the primary copy and the other copies are slave copies
- When a data item is to be updated transaction coordinator must determine where a primary copy is in order to send a lock request to a lock manager to appropriate local site
- It is necessary to put an exclusive lock on the primary copy
- While primary copy is updated a change can be propagated to slave copies
- Propagation must be done as soon as possible to prevent other transactions to read old slave copies
- However, the protocol guarantees that only the primary copy is current

Locking protocols

Distributed 2PL protocol is based on the following principles

- Distributed 2PL distributes lock managers to every local site
- Lock manager is responsible for managing locks in its own local site
- If data is not replicated the protocol is the same as primary copy 2PL
- Otherwise distributed 2PL implements read one write all replica control
- It means that any copy of a replicated data item can be used for read and all copies must be exclusively locked before an item can be updated

Locking protocols

Majority locking protocol is based on the following principles

- Majority locking is an extension of distributed 2PL that avoids to lock all copies of a replicated item before an update
- The protocol maintains a lock manager at each site to manage locks of all data at the site
- When transaction wishes to read or write a data item replicated in `n` sites then it must send a lock request to more than half of `n` sites where the item is stored
- A transaction cannot proceed until it obtains locks on a majority of the copies
- If a transaction does not receive majority of locks after certain period of time it informs all sites about its cancellation
- Otherwise it informs the sites about successful attempt to lock a majority of items
- Any number of transactions can simultaneously hold a shared lock on a majority of copies
- Only one transaction can hold an exclusive lock on a majority of copies

Transaction Processing in Distributed Database Systems

Outline

Distributed database recovery

Distributed recovery maintains atomicity and durability of distributed transactions

Recovery in a distributed DBMS is more complicated than in a centralized DBMS because atomicity is required for local and global transactions

Global transaction cannot commit until all its subtransactions are committed or aborted

Recovery protocol must ensure that the failures in one site do not affect processing in the other sites, i.e. it must be nonblocking protocol

Every global transaction has one site that acts as a coordinator (transaction manager)

Local sites where a global transaction has agents are called as participants (resource managers)

Transaction Processing in Distributed Database Systems

Outline

Two-phase commit protocol (2PC)

Global `COMMIT` or global `ROLLBACK` is performed in two phases: voting phase and decision phase

PHASE 1

- All participating systems inform a coordinator that a transaction at a local system is completed
- A coordinator sends a message can commit ? to local systems
- All participating systems force-write all log records and information needed for recovery and send ready to commit message to a coordinator
- If a participating system cannot force-write all log records then it sends cannot commit message to a coordinator

Two-phase commit protocol (2PC)

Global `COMMIT` or global `ROLLBACK` is performed in two phases: voting phase and decision phase

PHASE 2

- If all participating systems reply with ready to commit message then a coordinator sends commit message to all participating systems
- Each participating systems complete the transactions by writing `COMMIT` to a transaction log and optionally permanently updating a database
- If at least one of participating systems reply with cannot commit message then a coordinator sends rollback message to all participating systems

Two-phase commit protocol (2PC)

Problems with 2PC protocol:

- 2PC protocol is a blocking protocol
- Blocking protocol means that if a coordinator fails then all participating sites must wait until a coordinator recovers
- If a coordinator and one of participating transactions fails together then the distributed transaction becomes nondeterministic
- It means that it is impossible to ensure that all participants got commit message in the second phase
- Then some of participants may commit independently on the other participants

Transaction Processing in Distributed Database Systems

Outline

Three-phase commit protocol (3PC)

In 3PC the first phase is the same as in 2PC

The second phase is divided into PREPARE-TO-COMMIT and COMMIT phases

PHASE 1

- All participating systems inform a coordinator that a transaction at a local system is completed
- A coordinator sends a message can commit ? to local systems
- All participating systems send yes message to a coordinator
- If a participating system send a message no then a coordinator sends abort message

Three-phase commit protocol (3PC)

In 3PC the first phase is the same as in 2PC

The second phase is divided into PREPARE-TO-COMMIT and COMMIT phases

PHASE 2

- If all participating systems reply with yes message then a coordinator sends pre commit message to all participating systems and waits for "acknowledgement" message
- Each participating system replies with acknowledgement that it is ready to commit
- At this point each participating system is aware that global commit is possible
- If a participating system is not able to reply with acknowledgement message the transactions are aborted by a coordinator

Three-phase commit protocol (3PC)

In 3PC the first phase is the same as in 2PC

The second phase is divided into PREPARE-TO-COMMIT and COMMIT phases

PHASE 3

- A coordinator sends do commit message to all participating systems
- Each participating system replies with has committed message after `COMMIT` operation was successful

References

T. Connoly, C. Begg, Database Systems, A Practical Approach to Design, Implementation, and Management, Chapter 25 Distributed DBMSs - Advanced Concepts, Pearson Education Ltd, 2015