Distributed Data Storage and Management Part IV

Saptarshi Pyne
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Jodhpur, Rajasthan, India 342030

CSL4030 Data Engineering Lecture 14
September 1st, 2023

What we discussed in the last class

- Data transparencies
- Distributed/global transactions
    - The ACID properties

A thought-provoking question

Q. In the “aliasing” scheme for providing the local transparency to users, what happens if the query server crashes?

Execution of a global/distributed transaction

Each site has a log file and two computer programmes – a transaction manager (TM) and a transaction coordinator (TC).

SBI	**ICICI**
SBI initiates transaction T_i.	
TC$_{SBI}$ starts the execution.	
TC$_{SBI}$ breaks the transaction into two sub-transactions and distributes them to appropriate sites.	
TM$_{SBI}$ executes the following sub-transaction:	TM$_{ICICI}$ executes the following sub-transaction:
lock(A); read(A);	lock(B); read(B);
A = A – 50;	B = B + 50;
write(A); unlock(A);	write(B); unlock(B);
TM$_{SBI}$ maintains a log for recovery purposes.	TM$_{ICICI}$ maintains a log for recovery purposes.
TM$_{SBI}$ informs TC$_{SBI}$ that it has completed its task.	TM$_{ICICI}$ informs TC$_{SBI}$ that it has completed its task.
TC$_{SBI}$ sends a “commit T_i” message to all TMs.	
TM$_{SBI}$ adds <commit T_i> to its log.	TM$_{ICICI}$ adds <commit T_i> to its log.

What could go wrong?

- Site failures
- Loss of messages
- Link failures and ‘network partitions’

Resolution: The two-phase commit protocol (2PC)

The two-phase commit protocol (2PC)

SBI	**ICICI**
Phase 1: SBI initiates transaction T_i and TC_{SBI} starts the execution. TC_{SBI} breaks the transaction into two sub-transactions and distributes them to appropriate sites along with a "prepare T_i" message.	

TM_{SBI} adds <prepare T_i> to its log and executes the following sub-transaction T_{i1}:
lock(A); read(A);
A = A – 50;
write(A);
TM_{SBI} logs <ready T_i> and sends a "ready T_i" message to TC_{SBI}. If T_{i1} fails, TM_{SBI} logs <no T_i> and sends an "abort T_i" message to TC_{SBI}.
	TM_{ICICI} logs <prepare T_i> and executes the following sub-transaction T_{i2}:
	lock(B); read(B);
	B = B + 50;
	write(B);
	TM_{ICICI} logs <ready T_i> and sends a "ready T_i" message to TC_{ICICI}. If T_{i1} fails, TM_{ICICI} logs <no T_i> and sends an "abort T_i" message to TC_{ICICI}.

Phase 2: If and only if TC_{SBI} receives a "ready T_i" message from every TM before the timeout (*ready state*), TC_{SBI} sends a "commit T_i" message to all TMs. Otherwise, TC_{SBI} sends an "abort T_i" message to all TMs.	
TM_{SBI} adds <commit T_i> or <abort T_i> to its log, and commits/rolls back its T_{i1}.	TM_{ICICI} adds <commit T_i> or <abort T_i> to its log, and commits/rolls back its T_{i2}.

[1] Chap 19, Korth.
[2] https://www.geeksforgeeks.org/two-phase-commit-protocol-distributed-transaction-management/

2PC (contd.)

SBI	ICICI
TM_{SBI} sends an “acknowledge T_i” message to TC_{SBI}. unlock(A); If TC_{SBI} receives the “acknowledge T_i” messages from all TMs before timeout, it logs <complete T_i>. SBI sends the “Payment successful” or “Payment failed” message to John.	TM_{ICICI} sends an “acknowledge T_i” message to TC_{SBI}. unlock(B);

[1] Chap 19, Korth.
[2] https://www.geeksforgeeks.org/two-phase-commit-protocol-distributed-transaction-management/

2PC: Handling of failures and limitations

- **Site failures:** Nothing happens to their log files since the log files are stored in local secondary storages.
  - See ‘in-doubt transactions’ in Section 19.4.1.3.

- **Network partitions:** Similar to site failures.

- **Coordinator failures:** Data items A and B remain locked until the coordinator recovers. Even other transactions involving A and B get blocked. This is the infamous ‘**Blocking problem**’.
  - **Proposed solutions:** 3PC and persistent messaging protocols.

The three-phase commit protocol (3PC)

SBI

Phase 1:

Same as that of 2PC.

Phase 2:

If and only if TC_{SBI} receives a “ready T_i” message from every TM before the timeout (*ready state*), TC_{SBI} sends a **“prepare_to_commit T_i”** message to all TMs. Otherwise, TC_{SBI} sends an “abort T_i” message to all TMs.

TC_{SBI} crashes in the process of sending the “prepare_to_commit T_i” or “abort T_i” messages to the TMs.

(If TC_{SBI} does not crash, Phase 3 will be similar to the remaining steps of 2PC.)

Phase 3:

If some of the TMs do not receive the “prepare_to_commit T_i” or “abort T_i” messages from TC_{SBI} before timeout, their TCs contact other available TCs. If at least a pre-specified number of TCs is up, together they elect a new TC for this transaction (using an ‘election algorithm’).

TC_{new} checks whether at least one of the TMs have received a “prepare_to_commit T_i” message or not. If one of them did, TC_{new} sends a “commit T_i” message to all TMs. Otherwise, TC_{new} sends an “abort T_i” message to all TMs. Thus everything gets back on track.

[1] Chap 19, Korth.

Today we discussed

- Commit protocols for distributed/global transactions ensure that a global transaction **either commits at all sites or aborts at all sites**.
  - The two-phase commit protocol (2PC)
  - The three-phase commit protocol (3PC)

Remaining sub-topics for distributed databases

- Concurrency control with locking protocols
- Availability
    - High availability at the cost of consistency: The Cloud
- Multi-database systems for heterogeneous distributed databases
- Distributed directory systems for managing data
    - The lightweight directory access protocol (LDAP)

References

- A. SILBERSCHATZ, H.F. KORTH, S. SUDARSHAN (2011), Database System Concepts, McGraw Hill Publications, 6th Edition.
  - Chapter 19. Distributed Databases
- Paper: Bronson et al., “TAO: Facebook’s Distributed Data Store for the Social Graph”, 2013 USENIX Annual Technical Conference (USENIX ATC ‘13).
  - Video: https://www.usenix.org/conference/atc13/technical-sessions/presentation/bronson

Thank you