Beyond FLP

Acknowledgement for presentation material

Chapter 8: Distributed Systems Principles and Paradigms: Tanenbaum and Van Steen

Paper trail blog: http://the-paper-trail.org/blog/consensus-protocols-paxos/
Distributed Commit

• Distributed database has the database replicated over multiple servers
• Commit operation must preserve the ACID properties
  – Atomicity, consistency, integrity, and durability
• On receiving a commit operation on a replica, all participants must be polled to reach a consensus on whether to commit or abort the transaction
• A consensus problem !!
Naïve Solution

• A coordinator process initiates the consensus
  – Coordinator is at the site where the request originate, and must communicate with other sites to reach consensus about commit/abort of the transaction

• Participants are the replicas which must be updated

• One-phase commit protocol
  – Coordinator tells the other processes whether or not to perform the operation locally

• Obvious drawback: what if any replica fails and cannot commit
2-phase Commit

• Coordinator sends VOTE_REQ to all participants
• On receiving VOTE_REQ, participant
  – Either, returns VOTE_COMMIT ➔ prepared to locally commit
  – Or, returns VOTE_ABORT ➔ cannot commit
• Coordinator collects all votes from participants
  – If all participants votes VOTE_COMMIT, then coordinator will commit, and sends COMMIT msg to all
  – Otherwise, ABORT msg to all
• Participants wait for msg from coordinator
  – COMMIT msg ➔ execute transaction locally
  – ABORT ➔ drop the transaction
2-phase Commit

Coordinator and participants have states where they block waiting for incoming msgs

1. Waiting for VOTE_REQ from coordinator
   1. Has not voted ➔ can ABORT
2. Coordinator is waiting for votes from (all) participants
   1. Coordinator can decide ABORT if even one vote is missing
3. Participant (after VOTE_COMMIT) waiting for msg from coordinator
   1. Cannot decide ➔ must have a way to terminate
2-phase Commit

- When a participant is waiting for msg from coordinator
  - Wait for coordinator to recover
  - Contact other participants to decide on the action

<table>
<thead>
<tr>
<th>State of Q</th>
<th>Action by P</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMIT</td>
<td>Make transition to COMMIT</td>
</tr>
<tr>
<td>ABORT</td>
<td>Make transition to ABORT</td>
</tr>
<tr>
<td>INIT</td>
<td>Make transition to ABORT</td>
</tr>
<tr>
<td>READY</td>
<td>Contact another participant</td>
</tr>
</tbody>
</table>

What if the coordinator crashes after receiving all VOTE_REQ replies, but before sending the response msg (COMMIT/ABORT) to any participant?
3-phase Commit

Coordinator State Machine

Participant State Machine

Avoids blocking in presence of fail-stop crashes

Phase-1: Same as 2PC
Phase-2: Coordinator sends a decision to all sites (PRECOMMIT)
Phase-3: sends COMMIT/ABORT msg to all sites

Does 3PC work in presence of network partition?
Paxos

- Paxos: a generalized consensus protocol for asynchronous network with crash failures
- FLP $\implies$ cannot guarantee both safety and liveness
  - Paxos maintains safety by sacrificing liveness over a period, if necessary
- 3PC is not resilient to fail-recover faults
  - If the network is partitioned, then two halves may reach different decision
Paxos Details

• Paxos agents:
  – Proposers: Proposes a value to be accepted as consensus value
    • Can be multiple proposers, but the protocol will terminate only when there is one proposer (multiple proposers must agree to one value)
  – Acceptors: provides majority voting
  – Learners: learns about the accepted value

• Important properties
  – Any proposal number is unique
  – Any two set of proposers have at least one acceptor in common
  – Value sent out by the proposer is the highest-numbered proposal of the ones accepted by acceptors
Paxos Details

- Proposals are ordered by some total ordering mechanism
- Use a majority voting instead of waiting for all participants to accept, as in 2PC
  - Half the nodes can fail to reply

- Every proposal has a unique sequence number
  - Used for total ordering of proposals
  - Acceptor determines the highest numbered proposal it has received, and returns a promise that no lower numbered proposal will be accepted

Source: http://the-paper-trail.org/blog/consensus-protocols-paxos/
Paxos: Protocol Steps

PROPOSERS:

1. Submit a proposal numbered $n$ to a majority of acceptors.
   i. Wait for a majority of acceptors to reply.

2. If majority reply ‘agree’
   i. Acceptors send back value of any proposals already accepted.
   ii. Pick one of these values, and send a ‘commit’ message with the proposal number and the value.
   iii. If no values have already been accepted, use your own.
   Else, majority reply ‘reject’, or fail to reply, abandon the proposal and start again.

3. If a majority reply to a commit request with an ‘accepted’ message, consider the protocol terminated. Otherwise, abandon the proposal and start again.

ACCEPTORS:

1. On proposal receive,
   1. Compare proposal number to the highest numbered proposal already agreed to.
   2. If the new proposal is higher, reply ‘agree’ with the value of any proposals you have already accepted.
   3. If it is lower, reply ‘reject’, along with the sequence number of the highest proposal.

2. On ‘commit’ receive,
   1. Accept it if,
      a) the value is the same as any previously accepted proposal and
      b) its sequence number is the highest proposal number you have agreed to.
   2. Otherwise, reject it.
Paxos: Failure Handling

When two proposers are active at the same time, they can alternately issue higher numbered proposal to override the other.

Paxos does not terminate till one proposer is unanimously chosen \(\Rightarrow\) one proposer lets other one go first.
Applications of Paxos

• Googles Chubby lock service
• Yahoo Zookeeper uses a variant of Paxos

• Several interesting papers on Paxos as a building block for implementing systems
  – Lampson – “How to Build a Highly Available System Using Consensus”
  – Chandra, Griesemer, Redstone – “Paxos Made Live: An Engineering Perspective”
  – Kirsch and Amir – “Paxos for System Builders”