Network partitions

- Groups of nodes may be isolated or nodes may be slow in responding

No data replication

- If data unavailable, we are stuck...
- A problem: partition during commit protocol

Quorums

\[ C_1 = \{\{a,b,c,d\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}\} \]
\[ A_1 = \{\{a,b\}, \{a,c\}, \{a,d\}, \{b,c\}, \{b,d\}, \{c,d\}\} \]

Important property: \( X \in C_1 \Rightarrow \forall Y \in A_1 \ X \cap Y \neq \emptyset \)
\[ Y \in A_1 \Rightarrow \forall X \in C_1 \ X \cap Y \neq \emptyset \]
• Quorums can be implemented with vote assignments
  
  To commit ≥ 3
  To abort ≥ 2

Voters to commit + voters to abort > total votes

• Commit protocol must enforce quorum. b
  a. c
d. e

If node knows transaction could have committed (aborted), it cannot abort (commit) even if abort (commit) quorum available

All commit protocols are blocking (with network partitions)
  • can we do anything about it?

3PC Example
• To make commit decision: commit quorum
• To make abort decision: abort quorum

Example:
- votes for commit VC = 3
- votes for abort VA = 3

Note: need to go to Prepare to Abort state (PA), analogous to Prepare to Commit state (PC)

What if new group has following states?
• What if new group has following states?

\[
\begin{align*}
1 &. \text{ w} \\
1 &. \text{ PC} \\
1 &. \text{ PA}
\end{align*}
\]

• Can this happen? How?
• Could transaction have aborted?
• Could transaction have committed?
• What do we do? Block?

Another 3PC Example

\[
\begin{align*}
1 &. p \\
\text{ old coordinator} &. 2 \\
1 &. w \\
\text{ new coordinator} &. 1
\end{align*}
\]

\[VC = 3; VA = 3\]

• Coordinator could not have aborted
• Have commit quorum
  \(\Rightarrow\) Try to commit!

Yet Another 3PC Example

\[
\begin{align*}
1 &. p \\
\text{ old coordinator} &. 2 \\
1 &. w \\
\text{ new coordinator} &. 1
\end{align*}
\]

\[VC = 3; VA = 3\]

• Not enough votes!
  \(\Rightarrow\) Block!

Not all quorums can be implemented via votes

\[C_2 = \{\{a,b\}, \{c,d\}\}\]
\[A_2 = \{\{a,c\}, \{a,d\}, \{b,c\}, \{b,d\}\}\]
Partitions and data replication

Options:
(1) all copies required for updates
(2) Group may update, but at most one (at a time)
(3) Any group may update

Reading replicated data

Coterie

\[ C_1 = \{a, b, c, \} \{a, b, d, \} \{a, c, d, \} \{b, c, d, \} \]
\[ C_2 = \{a, b, \} \{a, c, \} \{a, d, \} \{b, c, d, \} \]

\[ X_1 = \{a, b, \} \{c, d, \} \] not valid

Important property:

\[ S \subseteq C \Rightarrow \forall G \in C, S \cap G \neq \emptyset \]

Reading replicated data - Votes

C1 = \{a, b, c, \} \{a, b, d, \} \{a, c, d, \} \{b, c, d, \}
R1 = \{a, b, \} \{a, c, \} \{a, d, \} \{b, c, \} \{b, d, \} \{c, d, \}

C2 = R2 = \{a, b, \} \{a, c, \} \{a, d, \} \{b, c, d, \}

Incidentally, which one is “better”?

C1 = \{a, b, c, \} \{a, b, d, \} \{a, c, d, \} \{b, c, d, \}
R1 = \{a, b, \} \{a, c, \} \{a, d, \} \{b, c, \} \{b, d, \} \{c, d, \}

C2 = R2 = \{a, b, \} \{a, c, \} \{a, d, \} \{b, c, d, \}
Note
• Note: not all coteries have vote assignments

Nodes = \{a, b, c, d, e, f\}
C = \{\{a, b\}, \{a, c, d\}, \{a, c, e\}, \{a, d, f\}, \{a, e, f\}, \{b, c, f\}, \{b, d, e\}\}

A Problem
Example: a 3 node system, 1 vote each node, replicated data
Now:
T_1 \cdot a
T_1 \cdot b
T_1 \cdot c

T_1 is committed at a, b

Later:
T_2 reads at c (not seeing T_1);
then writes and commits at a, c

Solution
• each node keeps list of committed transactions
• compare list at read site with those at write sites
• update sites that missed transactions

Example Revisited
Initially

T_0, T_1 \cdot a
T_0, T_1 \cdot b
T_0

\cdot c
Each node must keep updates for transactions until all nodes have seen them
- interesting problem...

Separate Operational Groups

For integration
(1) Compensate transactions to make schedules match
(2) Data-patch: semantic fix
Example: compensation

- Say \( T_1 \) commutes with \( T_3 \) and \( T_4 \)
  
  E.g.: no conflicting operations

- At DB:
  
  Schedule = \( T_0, T_3, T_4, T_1 \)
  
  (equivalent to \( T_0, T_1, T_3, T_4 \))

• At DB3:

  Schedule = \( T_0, T_1, T_2, T_2^{-1}, T_3, T_4 \)
  
  (equivalent to \( T_0, T_1, T_3, T_4 \))

• In general:

  Based on characteristics of transactions, can “merge” schedules

Example: Data Patch

- forget schedules!
- integrate differing values via “rules”

Simple rules

- For X: site 1 wins
- For Y: latest timestamp wins
- For Z: add increments
Network Partitions: Summary

- No replicated data
  - abort, commit quorums
  - commit protocols
- Replicated data
  - at most one operational group
    - coterie
    - propagation of updates
  - multiple operational groups