Non-Blocking Two Phase Commit (2PC) Using Blockchain

***Paul Ezhilchelvan**, Amjad Aldweesh and Aad van Moorsel*

School of Computing,

Newcastle University, UK

Contributions

- Non-blocking 2PC in a synchronous database system
  - Impossible when the coordinator crashes – Skeen, 1981
  - Possible if C lets a synchronous blockchain coordinate 2PC – CryBlock, 2018
  - Cost of (Ethereum) coordination

- Non-blocking 2PC in an asynchronous database system
  - Not possible if an asynchronous blockchain coordinates 2PC
  - Consistent with known impossibility result e.g., Fischer-Lynch-Patterson, 1985

- Synchronous vs Asynchronous blockchains
  - Distinction is significant for offering application level guarantees

Synchrony vs Asynchrony

- Upper bounds on completion time for important primitive activities
  - If Known: Synchronous
  - If unknown: Asynchronous
- Typical upper bounds
  - Bound on message transmission delay (d),
  - Bound on clock synchronisation accuracy (a),
  - Bound (β) on duration between
    - launch of a valid transaction and the first confirmed entry of a block containing that transaction
  - Bound (α) on duration in which distinct observers get aware of a transaction entry
- Known bounds $\Rightarrow$ reliable timeouts for accurate deduction of
  - Non-occurrence of events
  - Occurrence of Crash

Synch vs Asynch Ledgers

- In a synchronous DB cluster, *d* and *a* are known
- In a synchronous ledger, β and α are known

- Permissioned Ledgers can qualify to be synchronous
  - E.g., Hyperledger

- Permissionless ledgers may be asynchronous in nature
  - E.g., Ethereum relying on incentives for block composition, pos/pow
  - β can be unbounded with some small probability – *Weber et.al., 2017*

Background & Motivation

- In an ACID transaction,
  - Coordinator C distributes *work* to n servers
  - a server
    - performs allotted work, or
    - finds it not ‘safe’ to do the work, or
    - crashes
  - 2PC is the mandatory *finale* for completing an ACID transaction
  - C starts 2PC and also ends it
- 2PC Blocking
  - If C crashes between 2PC start and end, servers must wait until C recovers
  - Data of blocked transactions cannot be accessed by other ACID transactions

2 Phase Commit – rationale for the finale

Aim (either all or none):

- If *all* servers performed the allotted work, *all* decide commit
- Otherwise, servers decide to abort any work they had done
- The “otherwise” scenarios: some servers crashed or found doing work unsafe
- Harder to handle in an asynchronous cluster is the best-case scenario:
  - Commit must be the decision if every server is operative **and** claims to have completed the work allotted to it.

2PC outline

- Coordinator forms a global picture, decides and enforces decision

- Coordinator:
  - Ask each server ‘R U good?’
  - Wait on a timer
  - No response from a server within timeout ⇒ ‘I am no good’ received

  ----------*phase 1 -> 2*

  - ‘I am good’ from every server → *decision* = commit;
  - Otherwise, *decision* = abort;
  - Broadcast *decision*

- A Server:
  - If work can’t be done → abort; quit;
  - Await ‘R U good?’ on timeout
  - Timeout → abort; quit
  - Else send ‘I am good’

  ----------*phase 1 -> 2*

  - Await decision;
  - Periodically request C if decision not received // ***blocking*** *if C crashed*

2PC with Synch Blockchain

- C *requests* a smart contract instantiation for 2PC execution
  - Supplies server ids, timeout to be used etc.
- On observing an instantiation, each server *votes*
  - Vote = 1 to announce ‘I am good’;
  - Vote = 0 to announce ‘I am no good’;
- After all servers vote, decision is displayed
- If decision not displayed until timeout,
  - Servers that *voted* 1 invoke *verdict* function of the smart contract
- No blocking if C crashes
- Timeout measured using block timestamps as ‘wall’ clock

Cost

Functions	Cost in GAS	Cost in USD
Deploy	845,550	1.403
Request	190,226	0.315
Vote	75,472	0.125
Verdict	55,102	0.091

- 1 GAS = 830.61 USD
- All servers invoke Vote()
- Each working server may end up invoking Verdict()

Summary

Database	Ledger	Non-Blocking 2PC	Reason
Synchronous	Synchronous	Yes	Ledger as a replicated state machine
Asynchronous	Asynchronous	Nope	Impossibility of perfect detection
Synchronous	Asynchronous	Appears no (?)	Impossibility of perfect detection

- Synchronous vs asynchronous Ledger significant for non-blocking 2PC
- What if synch/synch solution used for Asynch/Asynch?
  - Blocking will not happen
  - BUT, small prob that commit does not occur where it must occur

It always seems impossible until it’s done

Nelson Mandela

Questions?