Hyperledger fabric: a distributed operating system for permissioned blockchains

Androulaki et al.

Fabric

- *Passive* versus *active* replication
  - *Active*: the proposal/command is sent to the entire RSM and each node executes the command locally (each replica is “active”) ~ similarity to op-based CRDT
  - *Passive*: the command is executed at a **single** node (primary), the side-effects (updates) of the execution are sent to the each RSM replica. The replicas install the new state that they receive from the primary (“passive”) ~ similarity to state-based CRDT
- Fabric hybrid:
  - Active in the sense that multiple endorsers execute the same chain code
  - Passive in the sense that matching endorser side-effects get distributed to rest of system (but only if they match!)

Fabric

- A “distributed OS” for permissioned blockchains: general and modular
    - More a framework than an OS? Much higher level than a distributed OS
        - Modularity: different consensus pieces, different PLs that can be used for smart contracts, different endorsement policies
- Alternative approach: *execute-order-verify*
- Separate trust in application from trust in consensus/ ordering

Why *execute-order-verify?*

- (In contrast to *order-[execute-verify])*
- Throughput: Wasteful for everyone to execute chaincode = smart contract. Constrain set of nodes that execute, then use passive replication on side-effects to KVS to distribute exec. results.
    - Trad. blockchain combine trust with consensus: delegating nodes for execution adds complexity
    - execute-verify stage in BitCoin is cheap: simple DSL, and txns are easy to “execute” (e.g., validate)
- Non-determinism = bad = bugs = good to discover early. Execute first means you get to fail fast! You don’t want non-determinism to be discovered late in the processing of a txn/invocation.
    - Public bchain:
        - Problem with scaling to a large network: endorsement doesn’t scale.
        - Who is trusted to provide endorsement? Someone could provide incorrect endorsement.
        - How do you incentivize other nodes to execute and endorse — in a public bchain this will need some fee
- Perhaps this solution to non-determinism is overkill: emulate the execution (multiple times) and compare the output to check for non-det.
- Non-det checking modularity: **endorsement policy** for non-det checking (default: all endorsements should match)
- Trade-off: high contention of ops **to same keys** => client may not be able to satisfy *endorsement policy*
    - Mismatch because of non-det in code (good to catch)
    - Mismatch because of inconsistent state at the endorsement nodes (BAD!) ~ systemic race condition (by design)
    - High load is a problem for consistency checking of my non-det element (NO PROGRESS GUARANTEE)

Fabric design

- Networking protocols:
    - BitCoin: use gossip txns and blocks (trust no one; no privacy)
    - Fabric: use gossip distribution of blocks (public state of the chain)
        - Use point-to-point for endorser set (execute stage) based on policy: only target the endorsers you need — this provide privacy for invocations).
- Ordering:
    - BitCoin: *Any full node can be an ordering node: if they are first to a PoW + head of chain + other nodes believe them.*
        - *Ordering is algorithmic: [longest chain wins + chain is a chain + …]; and everyone agrees on this algo*
    - Fabric: Uses orderer nodes (OSN): stateless, can be swapped out (for different variants), centralized (multiple for redundancy)
        - Ordering is in a single (trusted) place + application-unaware

Fabric evaluation

- FabCoin ~ BitCoin => transactions are light-weight; trivial smart contracts (chaincode)
- What is a fair Fabric comparison system?
    - *Comparison with an order-execute system! (Previous systems are the expected baseline)*
- Lacking: Experiment with high load to same key, varying the time between txns to the key: from 1ms between access to 1s — measure txns throughput or endorsement failure (for a strict policy); Could also evaluate different policies
- No evaluation of key trade-offs being “sold” in the paper: No focus on modularity + other design elements
- WAN deployment compared against LAN (1 DC) ~ good proxy for multiple orgs that coordinate
- Strange to see SSD vs. RAM eval?

Closing discussion

- Breakout discussion:
  - Consider the papers we read in the course, which paper/topic was your favourite and why?

Next: project presentations!

- **Project presentations** schedule finalized
  - 12m talk + 5m Q/A
  - Time must be split evenly between all group members

- **Project report+code** due December 11th by 6PM PT
  - **Report** as pdf via email. Instructions on homepage:
    - *https://www.cs.ubc.ca/~bestchai/teaching/cs538b_2020w1/final-report.html*
  - **Code** as link to a public repository, or a private repo shared with my GitHub id *bestchai*