Distributed Systems
236351
Tutorial 12 - Hyperledger-Fabric (HLF)

Yehontan Buchnik

Technion

January 24, 2019
HLF

Overview

HLF is a Distributed Operating System for Permissioned Blockchains, recently developed in IBM

- Blockchain can be viewed as an immutable ledger for recording transactions, maintained within a distributed network of mutually untrusting peers.
- Blockchains may execute arbitrary, programmable transaction logic in the form of **smart contracts**
- Prior permissioned blockchains suffer from many limitations due to their **order-execute** architecture
- HLF address those limitation by offering the **execute-order-validate** architecture
- These days HLF is used in more than 400 prototypes, proofs-of-concept, and in production distributed ledger systems, across different industries and use cases.
Order-Execute Architecture for Blockchains

Overview

In the order-execute architecture a node in the network typically performs the followed:

- The node assembles a block containing (possibly) valid transactions
- Then the node disseminates its block through an ordering service (i.e., paxos based atomic broadcast)
- The node collects blocks that were received from other nodes, then executes and validates the transactions sequentially by a pre-defined deterministic order
- Finally, if the node update the state and persist the transaction in the blockchain
Order-Execute Architecture for Blockchains

Primary Limitations

**Sequential execution** Executing the transactions sequentially on all peers limits the effective throughput that can be achieved by the blockchain. For example, if an execution of a transaction is 200 ms, it means that the throughput is bounded by 5 transactions/sec. This may become a performance bottleneck for all but the simplest smart contracts.

**DoS attacks** Even in the benign case, the benign behaviour may be assumed only on servers. Thus, clients may behave maliciously and submit a transaction that may take a long time to determinate.
Execute-Order-Validate Architecture

Overview

- Execute
  - Simulate trans. and endorse
  - Create rw-set
  - Collect endorsements

- Order
  - Order rw-sets
  - Atomic broadcast (consensus)
  - Stateless ordering service

- Validate
  - Validate endorsements & rw-sets
  - Eliminate invalid and conflicting trans.

- Update state
  - Persist state on all peers
Execute-Order-Validate Architecture

Overview

To cope with the order-execute limitation, HLF offers a new scheme named **Execute-Order-Validate**

- First, a node simulate the transaction output according to the current state of its blockchain
- Then, it orders the new state with an ordering service (i.e paxos based **atomic broadcast**)
- The node collects the blocks that were received by the ordering service and validates the states in a sequentially pre-defined deterministic order
- If the state is valid it get persistent in its local blockchain replica, else it is ignored and aborted
HLF Transaction flow

Entities

**Client**  Client submits transaction proposals for execution, helps orchestrate the execution phase, and, finally, broadcasts transactions for ordering

**Endorser**  Endorser executes transaction proposals and validate transactions. Not all peers execute all transaction proposals, only a subset of them called **endorsers** does

**Orderer**  Orderers are the nodes that collectively form the ordering service. Orderers are entirely unaware of the application state, and do not participate in the execution nor in the validation of transactions.
HLF Transaction flow

Overview
HLF Transaction flow

Execution Phase

- Clients sign and send the transaction proposal to one or more endorsers for execution according to the application policy
- The endorsers simulate the proposal
  - A proposal is simulated against the endorser’s local blockchain state, without synchronization with other peers. Moreover, endorsers do not persist the results of the simulation to the ledger state
- As a result of the simulation, each endorser produces a value **writeset**, consisting of the state updates produced by simulation, as well as a **readset**, representing the version dependencies of the proposal simulation
After the simulation, the endorser cryptographically signs a message called *endorsement* and sends it back to the client.

The client collects endorsements until they satisfy the endorsement policy of the application. In particular, this requires all endorsers as determined by the policy to produce the *same execution* result.

Then, the client proceeds to create the transaction and passes it to the *ordering service*.
HLF Transaction flow

Ordering Phase

- When a client has collected enough endorsements on a proposal, it assembles a **transaction** and submits this to the ordering service.

- The ordering phase establishes a total order on all submitted transactions per application.

- To improve performance the ordering service batches multiple transactions into blocks and outputs a hash-chained sequence of blocks containing transactions.

- HLF was designed such that its ordering service is highly modular, and can be replaced easily.
HLF Transaction flow

Validation Phase

Blocks are delivered to peers either directly by the ordering service or through gossip. A new block then enters the validation phase which consists of three sequential steps:

1. The application policy evaluation occurs in **parallel** for all transactions within the block.

2. A read-write conflict check is done for all transactions in the block **sequentially**.

3. The ledger update phase runs last, in which the block is appended to the locally stored ledger and the blockchain state is updated.