COMP4801

Bidl-HLF: Fast Distributed Consensus Algorithm Optimized for Datacenters

Project Plan

Supervisor: Dr. Cui Heming

Mentor: Qi ji

Member: Zhi Weichen, BEng (CompSc), 3035448550

Contents

1 Introduction

The Cross-enterprise trading systems hold a stringent requirement for its infrastructure to tackle an enormous amount of transactions among mutually untrusted parties. Such requirements include, but are not limited to, a throughput high enough to handle all the transactions, an almost instant time for ledger-recording and, the robustness against various kinds of system faults.

A Blockchain is a distributed database where the nodes are communicating using a p2p network. Blocks, each containing a batch of transaction records, are added tail-to-tail into the chain. As shown in Fig. 1, every block also includes a cryptographic hash of the previous block, and the entire chain is therefore protected from tampering its content.

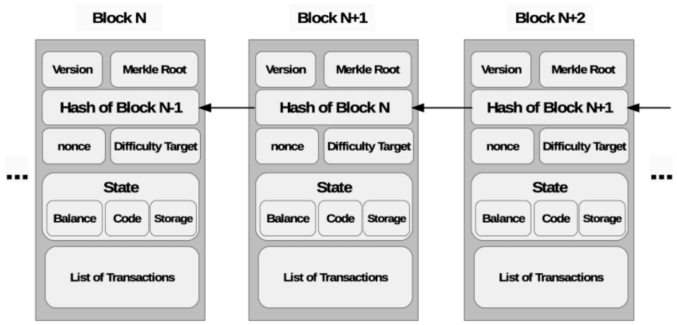

Fig. 1: Structure of a blockchain

Besides, blockchains can be categorized into either a *Permissioned* or *Permissionless* one. Permissionless blockchains are maintained totally under the p2p network without any entrance barriers or identification. Therefore, in order to reach an agreement of which node to append the latest block to the chain, a **consensus algorithm** must be conducted. One of the well-known consensus algorithms for permissionless blockchain is Proof-of-Work (PoW), where all nodes have to compete in solving cryptographic puzzles, i.e., computation intensive works, and the first one presenting the verified solution gets the privilege to append the block. However, the apparent long time for computation is obviously against the requirement for a low-latency financial system. For instance, Bitcoin is only capable of dealing with less than 10 transactions per second with a latency up to an hour (Vukolic, 2015), while a financial system like Visa will handle up to 24k transactions per second with a latency of only milliseconds (Hasbrouck & Saar, 2013).

On the other hand, Permissioned blockchains enforces a set of trusted nodes, and uses a Byzantine fault tolerant consensus protocol which will guarantee the correctness of the system even when some nodes in the network became malicious. However, there are limitations to the existing permissioned blockchains, since the consensus protocol now used is of such low a performance that it became the performance bottleneck of the entire blockchain. Also, the

sequential workflow (shown in Fig. 2) also degrades the performance, in that the part of the phases cannot run in parallel, which leads to a considerable latency.

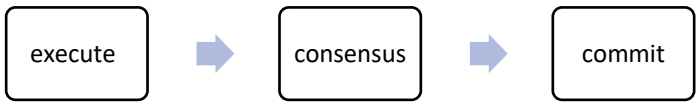

Fig. 2. Sequential workflow of HLF

Considering the points to improve stated above, we propose our optimization of blockchain as Bidl, which first focusing on the optimization of datacenter network ordering to increase throughput, then a new parallel workflow is presented to ease the workflow problem issued before. Also, a view change protocol based on blacklist is implemented to guarantee the performance even with malicious nodes within the network. Detailed principle and implementation will be discussed in 2.2 Bidl part.

2 Project Background

2.1 HLF

Hyperledger Fabric (HLF) is an open source, permissioned blockchain project under the Hyperledger umbrella project. Fig. 3 shows the general protocol of HLF for general transaction processing, and its principle is described as follows:

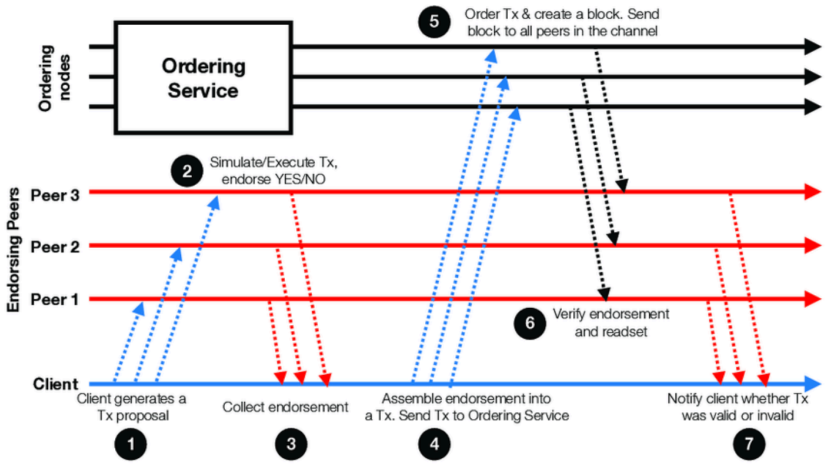

Fig. 3. HLF General transaction processing protocol

a. Client who wants to make a transaction first generate a transaction proposal tx=<ClientID, chaincodeID, payload>with its signature. Then he/she will send tx to all the endorsing peers for reviewing and voting.

b. Each endorsing peer will simulate the transaction and send back endorsements to the client saying whether or not it agrees with the transaction. They verify the client's authorization on best-effort, but make no ledger updates at this stage.
c. Client will receive all the endorsements and evaluate whether the transaction meets all the endorsement policies. With all things set, client will send a signed transaction proposal containing the peers' read/write sets, endorsement packs and the Channel ID.
d. Client will broadcast the proposal to Ordering service.
e. All proposals from all channels and clients will be ordered, then fitted into newly created chain blocks.
f. Ordering service send the ordered blocks to peers who will again verify the endorsement and the read/write set conditions. Decisions marks of either valid or invalid will then be made.
g. Peers will append the blocks, *both valid and invalid*, to the chain of the channel. And whenever the transaction is valid, the write sets are formally committed to the peer, and the client will receive a notice about the result of the Tx.

2.2 Bidl

This part will analyze the workflow applied by Bidl.

Workflow

As mentioned in the introduction part, Bidl possesses a parallel workflow where consensus and execution module are working asynchronously, as shown in Fig. 4. The full workflow is described as below:

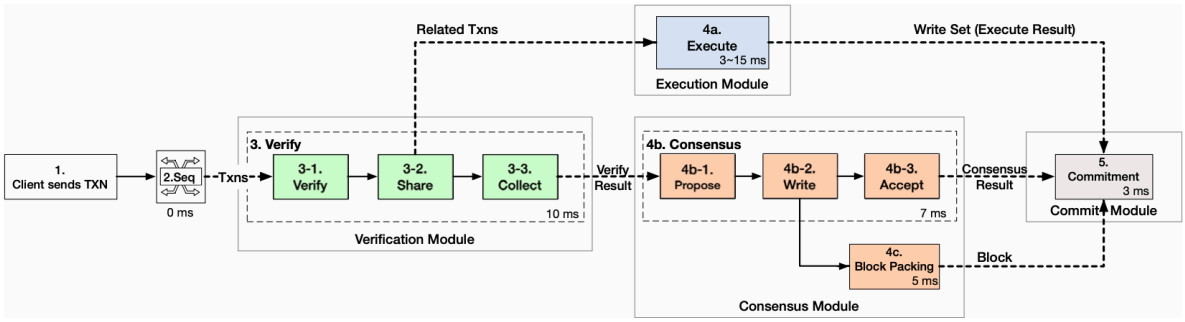

Fig. 4 Workflow of Bidl

1. Client sends his/her transaction to Sequencer module in Bidl group in the format of *tx:<<TXN, Payload, hash, privateFor, t⟩δ,s>*
2. A sequence number including a view number and a series number is added to each transaction in the sequencer module before broadcasted to verification module.

3-1. Nodes in verification module collect and verify incoming transactions. They conduct Blacklist check, Sequence check[1], Relevance check, Replay check, Signature check and Specific verification logic check respectively,

3-2. Relevant nodes exchange their verification decision.

3-3. Consensus leader collects the result in a straggler protocol to deal with long-time verifications. We will discuss about it in the methodology part.

4a. Execution module executes related transactions after verification completion asynchronously, without waiting for consensus to complete. This step will be discussed in methodology part.

4b. The job of consensus module is to guarantee the correctness and order in each node.

4b-1. Consensus leader will propose a result to other nodes

4b-2. On receiving *propose* message, the node verifies <*propose, v, b, h, H, R, nL*>, send <*write, v, b, h, ni*> to all other nodes if verification success or, request the entire block from the leader otherwise.

4b-3. Therefore, once the number of *write* message each node receives reached 2*f*, an <accept, v, b, h, ni> is sent to all other nodes.

4c. Each block then packing up the block as stated by leader node in *propose* message.

5. Each node who receives *2f+1 accept* messages will commit the write-set to the state and add the block to the chain.

3 Objective

The main objectives of the project are 1) the implementation of Bidl sequencer onto HLF framework and 2) expand Bidl's support for general transactions. Each objective should be fitted into the two development phases as stated in **5 timeline** part. We plan to achieve the main objectives in the following steps.

1 A comprehensive understanding of the principle of Bidl and HLF's Ordering service should be reached before any kind of implementation.

2 The HLF client would be re-implemented for the support of Bidl Sequencer.

3 [Important] A new execution protocol for Bidl workflow in support of general public transactions would be designed.

4 Based on the design, the new protocol would be implemented

5 As the previous evaluating progress, the new Protocol would then be evaluated with carefully designed steps.

[1] Sequence check: if current number is 1) in range of current block: skip 2) less than the current block: discard the transaction 3) larger than the current block: cache the transaction 4) of conflict hash with other transactions in the same block: put both side in local black-list as malicious

4 Methodology

This part will give detailed description about the problem mentioned above.

- **Straggler protocol**

This straggler protocol is introduced to handle the situation when the verification module takes too long to verify, i.e., to access external databases. The protocol first introduces priority level into verification results (Accept:1, Pending: 2, Reject:3, from low to high respectively) to make the final verification decision. Pending message is sent when the time spent in verification exceeds the time constraint as <*RESULT, s, h, Pending, ni*>

When final result is pending, all other nodes send back their verification with signature and ask client to re-submit the transaction with all the verifications, since those verifications will be skipped in the next run. Intuitively, Consensus leader under such circumstances will not add the pending transactions into the block.

- **Executing transaction**

As mentioned in 4a step of Bidl workflow, the execution on transaction tx(s) is initiated by n(i) when the following three conditions are met:

1) N(i) received all verification results of tx(s) from related nodes;
2) Previous transactions, i.e., with smaller serial number have been executed, and
3) No missing or duplicate sequence number in related transactions.

Besides, Bidl introduces dependency on transactions to decide whether they can be executed concurrently. In a nutshell, Bidl constructs non-circular dependency graphs with policies enforcing that transactions cannot be executed when there are unexecuted prerequisite transactions. As for non-dependent transactions, they get the permission to be executed in parallel.

- **Implementing network ordering**

Usually, network within a datacenter is implemented with software-defined networks, a high-level controller. Bidl nodes are implemented on the application layer of SDN, above control layer (network controller devices) and infrastructure layer (routers). Bidl sequencer is implemented using software-stimulated programmable switch, in order to reach the best performance.

- **Evaluation**

To evaluate the performance of our implementation of Bidl sequencer on HLF, we will run Bidl-HLF and HLF with several Byzantine consensus protocols. Each protocol will be evaluated with n = 3f+1 (f from 1 to 32) nodes. The main criteria are 1) the performance of Bidl-HLF compared with HLF alone 2) the robustness of Bidl sequencer on malicious sequences.

5 Timeline

SCHEDULED TIME	SCHEDULED WORK
SEP. 1 – OCT. 4, 2020	Background research
OCT. 4, 2020	First deliverable • Project plan • Project webpage
OCT. 5, 2020 – JAN. 10, 2021	Phase 1 • Careful review of the existing work as basis • implementation of Bidl sequencer on HLF
JAN. 15, 2021	First presentation
JAN. 24, 2021	Second Deliverable • Interim report • Implementation on HLF
JAN. 25 – APR. 17, 2021	Phase 2 • Focus on implementing public transactions • Evaluation and improvement of performance
APR. 18, 2021	Third Deliverable • Final Report • Final implementation
APR. 19, 2021	Final Presentation
MAY 4, 2021	Exhibition of the project
JUN 2, 2021	Completion of the project

Reference

1. C. Wang et al. (2018). PLOVER: Fast, Multi-core Scalable Virtual Machine Fault-tolerance, *15th USENIX Symposium on Networked Systems Design and Implementation.*
2. E. Androulaki et al. (2018). Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains, *EuroSys '18: Thirteenth EuroSys Conference*, Porto Portugal, pp. 1-15.
3. J. Hasbrouck & G. Saar. (2013). Low-latency trading. Journal of Financial Markets 16, 4 (2013), pp. 646–679.
4. J. Qi et al. (2020). Bidl: A High-Throughput, Low-Latency Permissioned Blockchain Framework in Datacenter Networks, *ACM Trans. Internet Technol.*, Vol. 1, No. 1.
5. J. Sousa, A. Bessani & M. Vukolic. (2017). A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform, *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Luxembourg City, pp. 51-58.
6. M. Castro & B. Liskov. (1999). Practical Byzantine Fault Tolerance*, OSDI '99: Proceedings of the third symposium on Operating systems design and implementationFebruary*, pp. 173–186
7. M. Vukolić . (20150. The quest for scalable blockchain fab- ric: Proof-of-work vs. BFT replication. In *Open Prob- lems in Network Security - IFIP WG 11.4 International Workshop, iNetSec 2015, Zurich, Switzerland, October 29,* pp. 112–125, Zurich, Switzerland.