2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 979-8-3503-8199-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/TrustCom60117.2023.00059

IHFBF: A High-Performance Blockchain Framework for Improving Hyperledger Fabric Permissioned Chain

1st Min Xu
College of Computer and Cyber Security
Fujian Normal University
Fuzhou,Fujian,China
minxu996@qq.com

2nd Xiaoding Wang
College of Computer and Cyber Security
Fujian Normal University
Fuzhou,Fujian,China
wangdin1982@fjnu.edu.cn

3rd Hui Lin*
College of Computer and Cyber Security
Fujian Normal University
Fuzhou,Fujian,China
linhui@fjnu.edu.cn

***Abstract*—Permissioned blockchain frameworks typically employ efficient Byzantine fault-tolerant consensus protocols, making them appealing for the deployment of fast transaction applications among a large number of mutually distrustful participants. However, existing permissioned blockchain frameworks typically use sequential serial workflows to invoke the consensus protocol and execute transactions for the application, resulting in significantly lower performance for these applications when deployed in traditional systems. Therefore, a new permissioned blockchain framework is needed to improve transaction processing efficiency and enhance system performance for practical blockchain technology applications. We propose IHFBF (Improved Hyperledger Fabric Blockchain Framework), an improved permissioned blockchain framework that employs a predictive transaction sorting method by selecting a node within the consensus nodes to act as a sorter. This enables parallel execution of the consensus protocol and transactions, resulting in improved overall system performance. However, if the sorter is a malicious node, it can severely impact system performance. To address this, IHFBF uses a view-change method based on a deny-list approach, which effectively guides all participants and replaces or denies malicious participants. Compared to other three fast permissioned blockchain frameworks, IHFBF's parallel workflow framework reduces latency and exhibits better throughput in the presence of malicious participants, resulting in efficient system performance.**

***Index Terms*—Permissioned chain, Byzantine fault-tolerant consensus, blockchain, parallel workflow, IHFBF**

I. Introduction

Since the publication of a paper titled "Bitcoin: A Peer-to-Peer Electronic Cash System" by an author named Satoshi Nakamoto in 2008 [1], blockchain technology mentioned in the paper has undergone rapid development. Blockchain can be divided into permissionless blockchain and permissioned blockchain. Permissionless blockchain allows anyone to join the network, participate in the consensus process, and verify transactions. Typical examples include Bitcoin and Ethereum. Permissioned blockchain refers to allowing authorized nodes to join the network and view information based on permissions. The immutability and decentralization characteristics of permissioned blockchain have attracted the industry and academia to develop various permissioned blockchain frameworks for transaction applications, with well-known examples including Hyperledger Fabric.

*Corresponding author

However, even when deployed in data center networks, existing permissioned blockchains have significant performance gaps compared to traditional transaction systems. The relatively low performance of traditional permissioned blockchains may be due to their own serial workflow processes. According to different workflows, permissioned blockchains can be divided into two categories. The first category is the execute-consensus-verify workflow proposed by Hyperledger Fabric (HLF) (as shown in Figure 1). Nodes first concurrently execute transactions received from clients, then use the Byzantine fault-tolerant consensus protocol to agree on the order of transaction execution results, and finally, after reaching consensus among the nodes, verify the transaction results, and submit valid transactions while rejecting invalid ones. Fabric is suitable for building trustless blockchain applications for various enterprises. However, if a large number of transactions are submitted simultaneously in the network, these transactions may fail due to conflicts in reading and writing, resulting in significantly increased system latency.

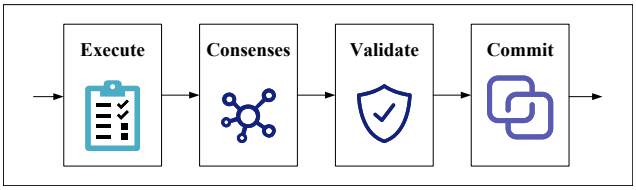

Fig. 1. Sequential workflow for permission chain HLF

The second type of permissioned blockchain framework (such as Quorum) uses a consensus-execute workflow [2]. In this workflow, nodes first use the BFT consensus protocol to sort transaction content, and then execute transactions separately in this order. However, to ensure that separately executed transactions obtain consistent outputs, it is necessary to rewrite all transactions using deterministic single-threaded

2324-9013/23/$31.00 ©2023 IEEE
DOI 10.1109/TrustCom60117.2023.00059

programming languages, which may reduce system efficiency.

To improve the performance of permissioned chains, transaction sorting and distribution can be moved to the routing layer. Specifically, we can designate a node as a dedicated sorter, which assigns a sequence number to all transactions and routes all transactions to all nodes through a routing-aware multicast. If the sorter is functioning normally, all nodes can receive all transactions in the same order, which eliminates the consensus process, and nodes can independently execute and submit transactions, greatly improving block generation efficiency and reducing latency.

In this paper, we propose an improved high-performance permissioned blockchain framework. The parallel workflow diagram of this new framework is shown in Figure 2: the consensus nodes first sort transactions in order, then call the Byzantine fault-tolerant consensus protocol to achieve consensus on the transaction order. In parallel, ordinary nodes predictively execute transactions from the sorter. Then the nodes securely submit the execution results if they are consistent with the consensus result, and re-execute the transaction if they are inconsistent.

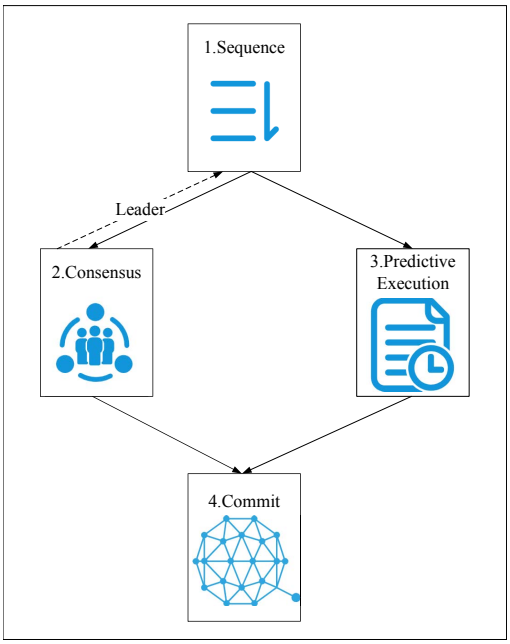

Fig. 2. New Parallel Workflow.

Correctly executing transactions in the presence of malicious nodes is a key issue in the parallel workflow. Although BFT consensus protocols can be used to detect malicious nodes and recover performance, there may still be some cunning malicious nodes that guide other nodes to execute incorrect transactions by broadcasting carefully crafted transactions, which can result in reduced system performance. To address this challenge, the new parallel workflow employs a deny-list-based view-change protocol to guide participants in the permissioned blockchain. If the Leader node behaves maliciously, the system uses random view switching to replace the Leader and adds the malicious node to the blacklist.

Our main contribution is the design of a high-throughput, low-latency blockchain framework with a parallel workflow (IHFBF), which introduces a sequence sorting node to achieve efficient transaction sorting. With this method, Peer nodes can directly perform predictive transaction execution without waiting for consensus results, achieving efficient parallel workflow. To address the potential existence of a malicious Leader node, we introduce a mechanism for randomly selecting a new Leader node for view switching using a random function. We also introduce a node blacklist system to prevent malicious nodes from launching attacks by broadcasting incorrect transactions.

The remaining sections of this paper are as follows: Section II discusses related work; Section III outlines the parallel workflow; Section IV presents the protocols of IHFBF; Section V shows correctness and performance analysis of IHFBF; Section VI conducts experimental analysis, and Section VII concludes.

II. Related Work

Permissioned blockchains [3]are typically maintained by mutually untrusted organizations, so running BFT protocols on many consensus nodes to tolerate many malicious nodes is essential. The consensus algorithm plays a critical role in the blockchain and directly affects its performance. During consensus processing, nodes need to verify pending transactions and sort them into new blocks, which requires verifying application-specific data encapsulated in transactions. To address this, Wanxin Li [4] proposed P-CFT, a zero-knowledge and crash-fault-tolerant consensus algorithm for permissioned blockchains. The algorithm can directly provide data privacy protection to the consensus layer while still providing crash fault-tolerance guarantees.

Yin et al. [5]proposed a leader-based Byzantine fault-tolerant replication partially synchronous model protocol, Hot-Stuff, which allows the correct leader to push the protocol to achieve consensus at the speed of actual network delays. Rashid et al. [6] proposed a multi-layer secure network model for IoT networks based on blockchain technology, which uses genetic algorithms and particle swarm optimization algorithms to divide the network into K unknown clusters, and then selects cluster leaders to form a public chain to synchronize information. However, as the network scale increases, the delay of the public chain will also increase. Chai et al. [7]also proposed a lightweight proof of knowledge (PoK) consensus mechanism. The consensus models the knowledge sharing process as a game between the leader group and the ordinary group in the transaction market. Although it reduces the computational cost, the communication cost does not decrease.

The concept of parallel execution has been widely used in existing systems, and now some people are introducing parallel thinking into blockchain systems. To improve transaction speed and scalability in blockchain systems, SS et al. proposed a parallel-based, rather than individually mining

Bitcoin to speed up the proof of work process, which improved the scalability of the proof of work by 24% compared to the current system. Xu et al. [8] proposed a time-space scheduling algorithm to optimize transaction parallelism and redundancy with a multi-transaction processing unit. The transaction processing unit achieves transaction optimization by asynchronously parallel execution and scheduling in the spatial dimension and fine-grained data and instructions in the time dimension. Saraph et al. [9] proposed parallel execution of transactions within each block of the Ethereum blockchain, and using this method, gas costs can be reduced by 2 times.

III. Overview

A. System Model

The blockchain nodes [10]–[12] in the system model are divided into two types: Normal Nodes (NN) and Consensus Nodes (CN). Normal nodes perform transaction execution, and consensus nodes are responsible for BFT consensus on submitted transactions. Nodes are grouped into organizations, each of which runs several consensus nodes and normal nodes. A server can have both consensus nodes and normal nodes at the same time.The system model is given in Figure 3.

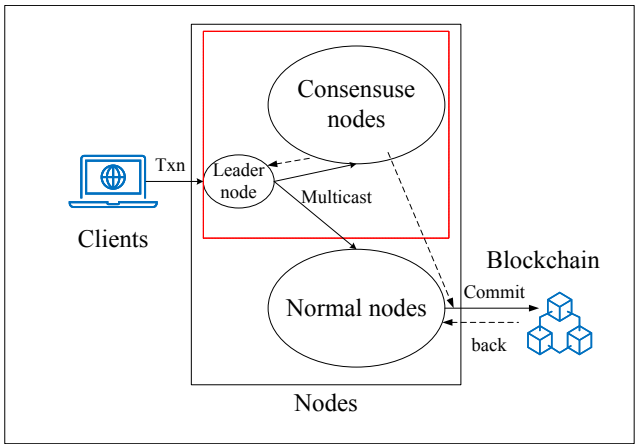

Fig. 3. New Parallel Workflow.

The blockchain used in this paper adopts BFT consensus, so there are at most $f = \frac{(|N-1|)}{3}$ malicious nodes among the consensus nodes. Similar to the typical HFL permissioned blockchain [11], nodes within the same organization trust each other, and nodes across different organizations do not trust each other. Clients may also be malicious and may collude with malicious nodes. We can refer to them as adversaries (denoted as $\mathcal{A}$).

B. Workflow Overview

The consensus process of IHFBF is divided into five phases. Figure 4 shows the workflow of the five phases:

The first phase is the submission phase, where the client submits signed transactions to the leader of the consensus nodes via a TLS-enabled connection. If the client sends a transaction with an incorrect format, the leader will disconnect to avoid a DoS attack from the client.

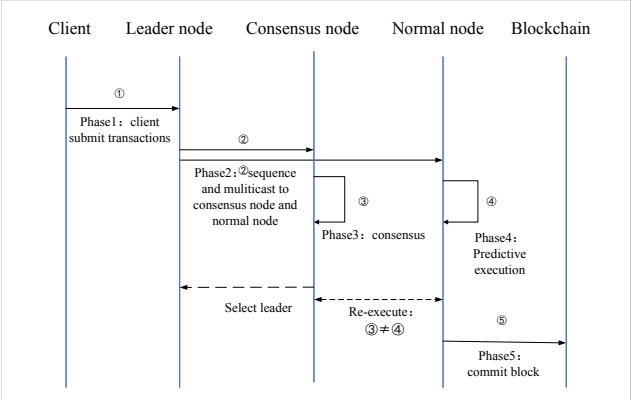

Fig. 4. Detailed workflow of IHFBF.

The second phase is the multicast phase, where we select the leader node as the sorting node in BFT consensus. The sorting node sorts the received transactions in order and broadcasts the sorting results to all consensus nodes and normal nodes.

The third phase is the consensus phase, where consensus nodes run the BFT protocol and, in the absence of malicious nodes, agree on the same order as the received transaction sequence and the transaction hash sequence broadcast by the leader.

The fourth phase is the predictive execution phase, which is concurrent with the third phase. Normal nodes speculatively execute the transactions sent by clients based on the sequence number sorted by the sorting node. If the node produces inconsistent results for the transaction, the transaction is aborted to ensure consistency of the state. Because the execution delay in the fourth phase is usually smaller than the consensus protocol delay in the third phase, the system's performance can be greatly improved. If there are malicious nodes in the nodes, the transaction order predicted by the predictive execution phase may be different from the order agreed upon in the consensus phase. In the fifth phase, if inconsistency is detected between the two, normal nodes will execute transactions according to the order agreed upon in the consensus phase, ensuring system security.

The fifth phase is the submission phase, where normal nodes of IHFBF submit transactions only after receiving matching committed transactions in the third phase and persistently executed results in the fourth phase. This ensures safe and reasonable liveness and reasonable high performance even in the presence of non-deterministic transactions.

The new workflow of IHFBF achieves security when there are non-deterministic transactions in the consensus workflow and zero interruption rate while meeting workload requirements, which is critical for improving performance in the consensus execution workflow. The parallelization of the consensus and execution phases greatly reduces the accumulated delay compared to the traditional serial workflow.

IV. Protocol Description

IHFBF uses a series of views with consecutive view numbers, where each view has a consensus node (n_L) as the leader. Each view in IHFBF has its own state, which is maintained by IHFBF clients and node caches.

A. Transaction Submission

In the IHFBF blockchain system, the first phase is for the client to submit transactions to the leader node(n_L)of the current view. The client (CL)sends a transaction $t(txn, p, n, v, pk)_{\sigma s}$, where txn represents the transaction, p is the transaction payload, n is the organization or node related to the transaction, v is the view number, pk represents the client's public key, and σ_s represents the transaction signed with the client's private key. The relevant nodes of the transaction refer to all normal nodes in the transaction-related organization in HLF, which should execute the transaction.

The second phase of IHFBF is multicast, where the leader node n_L multicasts the received transactions to all IHFBF nodes.n_L assigns a sequence number s to each transaction in order and broadcasts the marked transaction T ((Txn, p, n, v, pk), s) to all nodes (consensus nodes and normal nodes). When a node n_i receives the transaction T, it verifies whether the transaction already exists. If a transaction with the same sequence number has already been received, T will be discarded.

B. Consensus

The third phase, the consensus phase, is to determine the order of transactions. The leader node(n_L)sends transactions to the consensus nodes. When a consensus node receives a certain number of valid transactions or a specified time limit has elapsed, n_L hashes the transaction payload using SHA-256 and begins BFT consensus by sending the message $M(n_L, v_n, id, \mathcal{H})\sigma_s$ to all consensus nodes. Here, v_n represents the current view number, id represents the block ID, and $\mathcal{H}$ represents the list of transaction hashes. After receiving the M message, the consensus node CN checks the hash values of the transactions in$\mathcal{H}$. If there is any inconsistency, the consensus node requests the leader to retransmit the missing transactions. After receiving all the transactions in the M message, the consensus node CN begins the consensus protocol on the hash values of the transactions. Each consensus node combines the transactions into a block and delivers the block to the normal nodes according to $\mathcal{H}$.

C. Transaction Execution and Submission

In the fourth phase of IHFBF, after the leader node nL sorts the transactions, normal nodes speculatively execute the transactions. In the multicast phase, because the leader node in the consensus nodes has already numbered the transactions, normal nodes can directly execute the transactions in the order they received without waiting for the consensus result. Executing and consensus synchronously can improve the performance of the blockchain architecture. The execution result of the transaction will be submitted in the fifth phase. If the order of transaction hashes executed is different from the order of transaction hashes in the consensus phase, the system will re-consensus and execute.

In the fifth phase, normal nodes submit valid blocks to the blockchain ledger. After receiving the consensus results from more than $2f+1$ consensus nodes, normal nodes compare the hash value of their own executed transaction results with the consensus results. If they are consistent, the execution results of the transactions are uploaded to the blockchain ledger. If an adversary $\mathcal{A}$ pretends to be a leader node and publishes different transaction sequence numbers, it may cause transaction retransmission and speculative execution failures in the consensus phase, which will reduce the system's performance.

Based on the above description of the protocol process, the system workflow can be obtained, as shown in Algorithm 1:

Algorithm 1 System Workflow

Input: Initialize transaction hash list $\mathcal{H}$ to be empty. Consensus and normal nodes receive transactions T from clients.
Output: Normal node (NN) submits a successful block or re-executes the ordering.

if T is a new T and n_i is leader node **then**
  $s \leftarrow sequence(T), add\ \mathcal{H} \leftarrow \mathcal{H} \cup (s, hash(T))$
end if
while $T(txn, p, n, v, pk)$ is enough for a block or a timeout elapsed **do**
  Consensus nodes start agreement of $M(n_L, v_n, id, \mathcal{H})$ σ_s
  if normal nodes receive a block containing $2f + 1$ valid signatures from different consensus nodes and its executed transactions are consistent with the transaction **then**
    commit(T)
  else
    Normal nodes re-execute all transactions
  end if
end while

D. Leader Node

Malicious leader nodes can adopt various methods to reduce the performance of IHFBF, such as sending inconsistent transactions to nodes, deleting client transactions, or sending different sequence numbers. To address these issues, IHFBF uses a view change protocol similar to PBFT.

IHFBF's view change protocol is similar to PBFT, but there are differences. The rotation of leader nodes in IHFBF is unpredictable, unlike in PBFT where leader nodes rotate cyclically among consensus nodes.

Consensus nodes trigger IHFBF view changes in three situations. First, to ensure high performance of the blockchain architecture, consensus nodes trigger view changes when a significant drop in throughput is detected in phase 3. Second, if a client cannot receive transactions after a timeout, consensus nodes will initiate a view change. Third, the correct leader node will proactively invoke a view change to detect

suspicious and inappropriate behavior, while malicious leader nodes will act arbitrarily.

When an adversary pretends to be a leader node, broadcasts carefully crafted transactions, and causes conflicts in the sequence numbers of other nodes. These conflicts may cause transaction retransmissions in phase 3 and speculative execution failures in phase 4, reducing the performance of IHFBF.

E. Deny-List Protocol

For the problem of malicious leader nodes in the blockchain system, it can be detected and replaced by the BFT consensus protocol [13]–[15]. For the problem of malicious clients, we use the deny-list protocol [16]–[18] to solve it.

The specific steps of the deny-list protocol are as follows: First, if a node n_i receives two different transaction sequence numbers, it considers the transaction received in the multicast phase to be malicious and can add the client(CL) that sent the transaction to the suspect list $\mathcal{S}$. Second, during the process of switching different views in the consensus phase, if n_i detects a conflict in $f+1$ views of different leaders, it determines that the client(CL)in the suspect list S is malicious. Finally, n_i' carries CL when switching views. If $f+1$ consensus nodes detect transaction conflicts at this time, it is determined that the client(CL)is malicious and (CL) is added to the deny list $\mathbb{D}$.

The deny-list protocol can effectively prevent malicious clients from disrupting the consensus process and ensure the security and performance of the blockchain system.

V. IHFBF Performance Analysis

A. Effectiveness of the Deny-List Protocol in IHFBF

The deny-list protocol in IHFBF has a low false-positive rate, effectively detecting malicious clients. For the false-positive rate, IHFBF can add malicious clients to the deny list, making them detected in $f+1$ suspected views. To evade the deny list, an adversary$\mathcal{A}$ is best to only cause conflicts with a specific leader in a view, but due to IHFBF's proactive view changes and random leader rotation mechanism, this will inevitably lead to conflicts with different leaders in subsequent views. Therefore, adversary $\mathcal{A}$ is better off using a different client(CL) to reduce the probability of being suspected. Since in IHFBF adversary A cannot arbitrarily increase its collusive client list in the permitted blockchain, IHFBF usually has a low false-positive rate.

B. High Performance of IHFBF

The high performance of IHFBF is reflected in two aspects. First, for views with malicious leaders, consensus nodes can call view changes by detecting re-execution rates or throughput drops to replace leaders, ensuring the continuity of parallel workflow. Second, due to IHFBF's low false-positive rate, the malicious client used by the malicious node $\mathcal{A}$ will continuously cause conflicts in the sequence number space of normal nodes and be added to the deny list of IHFBF. Since $\mathcal{A}$ cannot arbitrarily increase its collusive client list, IHFBF can effectively ensure the high performance of the network model.

VI. Experimental Analysis

All experiments were conducted in an environment consisting of a cluster of three laptops, each equipped with an Intel Core i5 1.60GHz processor, 32 GB of memory, a 64-bit Win11 operating system, and VMware Workstation 20pro, as well as an Ubuntu system with 16 GB of memory and two processors. To verify the performance of the IHFBF framework, it was compared with three other blockchain frameworks: Hyperledger Fabric (HLF) [17], StreamChain [18], and FastFabric [19].

We evaluated the blockchain frameworks using the SmallBank [20] workload. The performance metrics were throughput and latency.

A. Workflow Performance

Figure 5 shows the workflow performance of IHFBF, FastFabric, and StreamChain without malicious nodes. It can be seen that the IHFBF framework performs significantly better than the other frameworks in terms of both throughput and latency. The experimental results demonstrate that IHFBF has high performance and scalability, making it suitable for various applications in the blockchain domain.

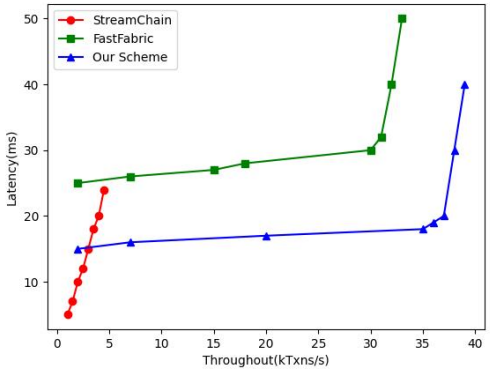

Fig. 5. Throughput and Latency without Malicious Nodes

StreamChain processes transactions in a streaming manner by sequentially processing each transaction in a pipeline form, thereby achieving relatively low transaction latency. Compared with other blockchain platforms that process transactions in parallel, StreamChain sacrifices some throughput but significantly improves latency performance. Its peak throughput is around 5k transactions.IHFBF performs better than FastFabric in terms of both throughput and latency. The throughput of IHFBF is higher than FastFabric because in IHFBF, ordinary nodes execute relevant transactions according to their numbers (Phase 4). Therefore, IHFBF does not need to perform heavy MVCC contention checks like the consensus execution workflow in FastFabric. So the peak throughput of IHFBF can exceed 30k transactions. The relatively lower latency of IHFBF is mainly due to the parallelization of the execution and consensus phases in IHFBF.

Figure 6 shows the workflow performance of IHFBF, Hyperledger Fabric, and FastFabric in the presence of 30% malicious nodes. From the figure, it can be seen that in the presence of a large number of malicious nodes in the system, our proposed IHFBF framework outperforms the other frameworks in terms of both throughput and latency metrics.

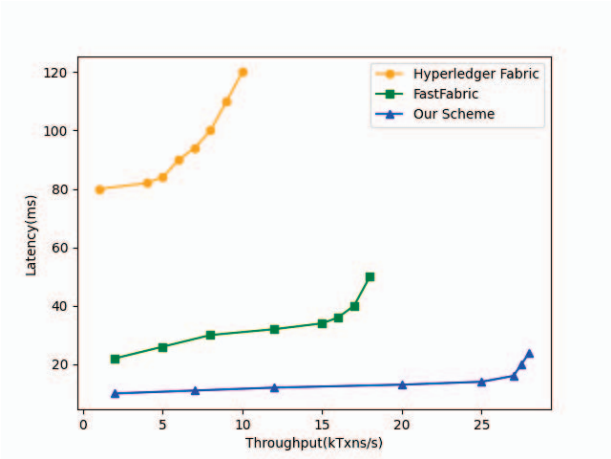

Fig. 6. Work Performance in the Presence of 30% Malicious Nodes

To evaluate the scalability of IHFBF, we tested the latency with different numbers of organizations. Each organization had one consensus node and one normal node. As shown in Figure 7, as the number of organizations increased, the latency of IHFBF on four BFTs rapidly decreased and slowly increased. We can analyze that when the number of organizations is small, the time of executing transactions affects the performance of the workflow. When the number of organizations gradually exceeds 25, the number of transactions executed by each organization decreases, and the time for consensus among organizations becomes the main factor that affects the delay of the workflow.

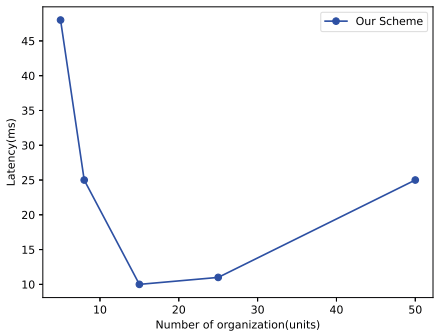

Fig. 7. Scalability of the System Framework

Table 1 shows the performance of IHFBF in the presence of malicious nodes. Er represents that the framework does not support the experimental conditions, F1: all nodes are running normally, F2: there exist malicious leader nodes. For each experiment, the worst-case performance of IHFBF and the relevant frameworks is recorded when the system is in a stable state after being attacked by malicious nodes. All blockchain frameworks run four BFT consensus nodes and ten normal nodes. The effective throughput of each system (i.e., the number of valid client transactions submitted per second, kTxns/s) is recorded after the system is stabilized following the attack of malicious nodes. As can be seen from the table, IHFBF has good throughput both in the absence and presence of malicious nodes, outperforming other blockchain frameworks.

TABLE I
OBSERVED THROUGHPUT OF BLOCKCHAIN FRAMEWORKS IN DIFFERENT SCENARIOS

Blockchain framework	F1:All nodes normal	F2:Malicious leader node
StreamChain	2.55	Er
HLF	8.25	8.25
FastFabric	26.21	Er
IHFBF	40.33	40.33

VII. CONCLUSION

To solve the problem of efficient and timely data storage, we propose a high-performance permissioned blockchain framework IHFBF, which is a new type of led parallel workflow that uses network ordering to realize the parallelization of transaction execution and consensus execution. By comparing and evaluating with other excellent permissioned blockchains, the study shows that IHFBF has low latency and high throughput, and also performs well with the participation of malicious nodes.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China under Grant No. 61702103 and U1905211, Natural Science Foundation of Fujian Province under Grant No. 2020J01167 and 2020J01169.

REFERENCES

[1] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system[EB/OL].2008.
[2] Luo, Haoxiang, et al. "ESIA: An Efficient and Stable Identity Authentication for Internet of Vehicles."arXiv preprint arXiv:2305.17377(2023).
[3] Wang, Zhe, et al. "A credibility-aware swarm-federated deep learning framework in internet of vehicles." Digital Communications and Networks(2023).
[4] Mollah, Muhammad Baqer, et al. "Blockchain for the internet of vehicles towards intelligent transportation systems: A survey."IEEE Internet of Things Journal 8.6 (2020): 4157-4185.
[5] Yin, Maofan, et al. "HotStuff: BFT consensus in the lens of blockchain." arXiv preprint arXiv:1803.05069(2018).
[6] Rashid, Mohammed A., and Houshyar Honar Pajooh. "A security framework for IoT authentication and authorization based on blockchain technology."2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, 2019.

[7] Chai, Haoye, et al. "A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles."IEEE Transactions on Intelligent Transportation Systems 22.7 (2020): 3975-3986.
[8] Hazari, Shihab Shahriar, and Qusay H. Mahmoud. "A parallel proof of work to improve transaction speed and scalability in blockchain systems."2019 IEEE 9th annual computing and communication workshop and conference (CCWC). IEEE, 2019.
[9] Pan, Rui, et al. "An Algorithm and Architecture Co-design for Accelerating Smart Contracts in Blockchain." Proceedings of the 50th Annual International Symposium on Computer Architecture. 2023.
[10] Saraph, Vikram, and Maurice Herlihy. "An empirical study of speculative concurrency in ethereum smart contracts."arXiv preprint arXiv:1901.01376(2019).
[11] Androulaki, Elli, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains."Proceedings of the thirteenth EuroSys conference. 2018
[12] Bessani, Alysson, João Sousa, and Eduardo EP Alchieri. "State machine replication for the masses with BFT-SMART."2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE, 2014.
[13] Qi J, Chen X, Jiang Y, et al. Bidl: A high-throughput, low-latency permissioned blockchain framework for datacenter networks[C]//Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles. 2021: 18-34.
[14] Golan Gueta, Guy, et al. "SBFT: A scalable and decentralized trust infrastructure."arXiv e-prints(2018): arXiv-1804.
[15] Kotla, Ramakrishna, et al. "Zyzzyva: speculative byzantine fault tolerance." Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles. 2007.
[16] Yin, Maofan, et al. "HotStuff: BFT consensus in the lens of blockchain." arXiv preprint arXiv:1803.05069(2018).
[17] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, Mobility-Aware Proactive Edge Caching for Connected Vehicles using Federated Learning, IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 8, pp. 5341-5351, 2021.
[18] J. Mills, J. Hu, G. Min, Communication-Efficient Federated Learning for Wireless Edge Intelligence in IoT, IEEE Internet of Things Journal, vol. 7, no. 7, 5986 - 5994, 2020.
[19] J. Mills, J. Hu, G. Min, Multi-Task Federated Learning for Personalised Deep Neural Networks in Edge Computing, IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 3, pp. 630-641, 2022.
[20] Z. Chen, J. Hu, G. Min, A. Zomaya, T. El-Ghazawi, Towards Accurate Prediction for High-Dimensional and Highly-Variable Cloud Workloads with Deep Learning, IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 4, 923 - 934, 2020.
[21] R. Jin,J. Hu, G. Min, J. Mills, Lightweight Blockchain-empowered Secure and Efficient Federated Edge Learning,IEEE Transactions on Computers,DOI: 10.1109/TC.2023.3293731, 2023.
[22] Androulaki, Elli, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." Proceedings of the thirteenth EuroSys conference. 2018
[23] Kuhring, Lucas, et al. "StreamChain: Building a Low-Latency Permissioned Blockchain For Enterprise Use-Cases." 2021 IEEE International Conference on Blockchain (Blockchain). IEEE, 2021
[24] Gorenflo, Christian, et al. "FastFabric: Scaling hyperledger fabric to 20 000 transactions per second." International Journal of Network Management 30.5 (2020): e2099.
[25] Choi, Wonseok, and James Won-Ki Hong. "Performance evaluation of ethereum private and testnet networks using hyperledger caliper."2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2021.