Parallel Execution of Blockchain Transactions with Sharding

Guo Chen[1], Jingjing Zhang[2], Weigang Wu[1], Jieying Zhou[1*]
[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]School of Cyber Security, Guangdong University of Foreign Studies, Guangzhou, China
Email: cheng93@mail2.sysu.edu.cn, 202210026@oamail.gdufs.edu.cn, {wuweig, isszjy}@mail.sysu.edu.cn
[*](Corresponding author)

***Abstract*—Scalability is one of the main problems limiting blockchain applications. Most recent research on blockchain scalability has focused on improving the consensus layer, but the most advanced consensus protocols can only reach a few thousand transactions per second(tps), which is far below the capacity of typical distributed databases. With the use of blockchain sharding technology, the time overhead corresponding to the execution layer and the consensus layer in the blockchain is gradually reduced. Additionally, improving the efficiency of transaction execution could incentivize nodes to actively verify transactions and enhance blockchain security. In this paper, we propose a Sharding Based Parallel Execution of Block Transactions (SPExTran) framework in blockchain, which transforms the sequential execution of intra-block transactions into parallel execution to improve the performance of blockchain. In addition, based on the characteristics of blockchain sharding transactions, we further propose a Blockchain Transaction Scheduling (BCTS) algorithm to improve the performance of blockchain by optimizing the parallel execution process of transactions. In the BCTS algorithm, the mechanism of local multi-processors sharing the Conflict Transactions cache Table (CTT) simplifies the verification process of conflicting transactions, thus improving the blockchain performance. Experimental results show that the throughput of blockchain using the SPExTran framework and BCTS algorithm is significantly higher than that of typical sharding technologies.**

***Index Terms*—Blockchain, Sharding, Parallel Execution of Transactions, Parallel Task Scheduling Algorithm**

I. Introduction

Scalability is one of the popular research fields in blockchain. There are two main categories of related research, on-chain and off-chain scaling [1]. On-chain Scaling includes proposing new consensus protocols [2], sharding [3] and DAG [4]. Off-chain scaling includes payment channel [5]- [7] and sidechain [8]. The proposed sharding technology greatly reduces the consensus size of network nodes, thus greatly reducing the overhead gap between the execution layer and the consensus layer. Therefore, improving the efficiency of the execution layer can also improve the performance of the blockchain with sharding. However, no related work considers the parallel execution of transactions within block for sharding. Moreover, the efficiency improvement of the execution layer and the consensus layer is not conflicting, but complementary. Therefore, these two layers are combined to improve the blockchain performance.

This paper presents a framework of Sharding Based Parallel Execution of Block Transactions (SPExTran) in Blockchain. In SPExTran framework, committee nodes are divided into leaders and followers. Leader sequentially verifies and executes transactions, generates Conflicting Transactions cache Table (CTT), packs them into block with Txs, and broadcasts block to followers. Followers verify and execute Txs in parallel according to CTT, determining correctness of the block and develop consensus on a result with other nodes.

In order to further improve the performance of blockchain, this paper designs a Blockchain Transactions Scheduling (BCTS) algorithm according to the characteristics of blockchain transactions in sharding. In BCTS algorithm, leader of sharding committee forms a group of transactions execution queues based on CTT. Followers verify and execute transactions in parallel on multiple processors. The mechanism of sharing CTT among local multi-processors simplifies the verification process of conflicting transactions, reduces the overhead of transaction verification, and further improves the efficiency of blockchain.

The SPExTran Framework and BCTS algorithm have a large impact on the overall performance of blockchain. The simulation results show that the SPExTran Framework and BCTS algorithm can increase blockchain performance by 14% to 68% against traditional sharding.

As far as we know, the SPExTran framework is the first sharding based framework, which converts the sequential execution of transactions into parallel execution to scale the blockchain. Our main contributions include:

1) Proposing a framework of Sharding Based Parallel Execution of Block Transactions (SPExTran) in Blockchain, which is a novel framework to reduce transaction execution time;
2) Designing Blockchain Transactions Scheduling (BCTS) algorithm in Blockchain;
3) Presenting a mechanism for local multiprocessors sharing Conflicting Transactions cache Table;
4) Implementing simulation experiments on SPExTran framework and BCTS algorithms and analyzing the results.

This research is partially supported by the Key-Area Research and Development Program of Guangdong Province (2020B0101090005), Guangdong Provincial Natural Science Foundation of China (2018B030312002), and the Science and Technology Program of Guangzhou, China (No. 202002020045).

ICC 2023 - IEEE International Conference on Communications | 978-1-5386-7462-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICC45041.2023.10279242

The remainder of this paper is organized as follows. Section II describes the existing works on blockchain scalability. Section III introduces the overall design of the proposed SPExTran framework. We present BCTS algorithm in Section IV. Section V discusses the evaluation setups and results, and Section VI concludes the paper.

II. Related Work

As an approach to improve blockchain scalability, sharding divides the blockchain network into multiple committees for a given period of time. Each committee deals with a different set of transactions and uses consensus algorithms to reach consensus. It improves the efficiency of consensus layer and reduces the gap in time overhead between execution layer and consensus layer.

Elastico [9] is the first secure candidate sharding protocol that tolerates a percentage of failed Byzantine network nodes. RapidChain [10] is an adapted approach to improve throughput through an efficient cross-shard transaction approach. OmniLedger [11] uses the Byzantine sharding atomic commit protocol to process transactions atomically across committees. To increase network throughput as the network scales, [12] proposed a scalable blockchain protocol using sharding and PoS consensus. SharPer [13] improves the efficiency of cross-shard transactions by dividing different transactions sets to different committees.On the other hand, a good sharding mechanism should minimize the number of cross-sharding transactions. For the purpose, [14] proposed OptChain, a new sharding paradigm, which minimizes the number of cross-sharding transactions. In addition, Monoxide [15] proposed a new Chu-ko-nu mining mechanism that can ensure cross-zone atomicity, efficiency and security for blockchains with thousands of independent mining zones.

Blockchain transaction models affect the verification and execution of transactions, as well as the mechanism of blockchain sharding. Transaction models of permissionless blockchain can be divided into two types: Unspent Transaction Output(UTXO)-based and account-based model. The research [16] on data analysis of blockchain shows that some intra-block transactions are dependent on each other (conflicting) but the conflict rate is not high. For example, Bitcoin's [17] intra-block transactions conflict rate is 13% and Ethereum's [18] intra-block transactions conflict rate is 20%. It proves the possibility of parallel execution of blockchain transactions. In addition, improving transactions execution efficiency can strengthen the incentive mechanism and enhance the security of blockchain by reducing the incentive for rational nodes to skip transactions execution [19]. There are some discussion about transactions parallel execution, including [20] for permissioned blockchain systems, [21] for block miner, [22] for account-based model blockchain, but no for sharding with UTXO-based model.

To the best of our knowledge, SPExTran framework is the first work based on sharding to improve the blockchain scalability by verifying and executing intra-block transactions in parallel. The BCTS algorithm is the first work to simplify the validation of conflicting transactions. In addition, SPExTran framework and BCTS algorithm can be used with the existing sharding consensus.

III. A Framework of Sharding Based Parallel Execution of Block Transactions (SPExTran)

This section outlines the framework of Sharding Based Parallel Execution of Block Transactions (SPExTran) in blockchain and introduces two innovations of intra-committee consensus and cross-shard transactions.

A. SPExTran Architecture

Based on UTXO transaction model and blockchain sharding technology, a SPExTran architecture is proposed for blockchain, which includes committee construction phase, consensus phase, and committee reconstruction phase. As shown in Fig. 1, the committee construction phase runs only once, while the consensus phase and committee reconstruction phase occur in each epoch, and the consensus phase consists of several iterative processes. The main innovative idea of the framework is that the epoch leader sequentially validates and executes Txs, generates the Conflicting Transactions cache Table (CTT), packs both the CTTs and the Txs into a block, and then broadcasts the block to followers. CTT includes notations, introduced in session IV, such as id, st, ft, rt, inputs, outputs and so on. Followers verify and execute Txs in parallel against CTT, determine the correctness of the block, and agree on the execution results with other nodes.

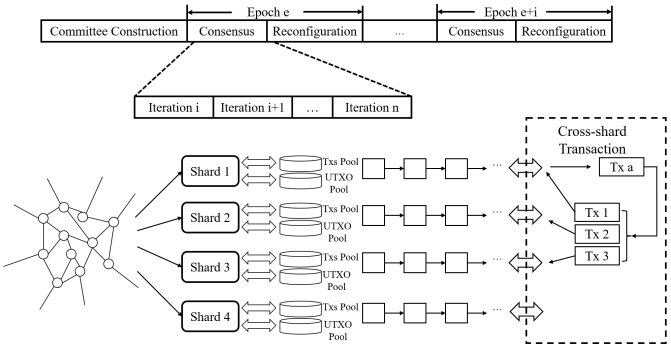

Fig. 1. SPExTran Architecture

B. Internal Committee Consensus

Internal committee consensus can be divided into two parts: the client and the sharding system, in which the sharding system has two role types, namely the leader and the follower, as shown in Fig. 2. The client submits the transaction to the sharding system.Committee nodes store transactions in local Txs pools, which are different from each other due to network latency and other issues.The leader verifies and executes the transactions in the local Tx pool, and then analyzes the transactions in the local UTXO pool to generate a CTT. Since all blockchain nodes in the same committee need to guarantee consistency, all UTXO pools of all nodes within the committee

are consistent.The leader then collects relevant information, creates a block and sends to all the followers via internal routing. Followers verify and execute transactions in parallel according to local UTXO pool and CTT. The follower receives the block and decides whether to add the block to the local blockchain according to the block header and other received verification results.

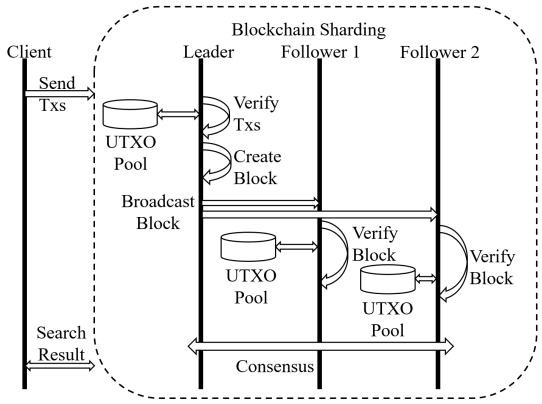

Fig. 2. Internal Committee Consensus

C. Cross-Shard Transactions

This paper uses a transaction splitting method to deal with cross-shard transactions. The process of CTT generation is optimized according to its characteristics. Transactions are assigned to the corresponding output committee based on the output of the cross-shard Tx. Similarly, a transaction may have multiple inputs from different committees, which are called input committees. As shown in Fig. 3, there is a transaction Tx, with two inputs $In1$ and $In2$ and one output Out. These two inputs and output belong to different committees $S1$, $S2$ and $S3$ respectively. $In1$ uses output $row1$ of Tx_a in $S1$. $In2$ uses the output $row1$ of Tx_b in $S2$. Tx is in $S3$. Leader of $S3$ creates three new transactions Tx_1, Tx_2, Tx_3 and sends Tx_1, Tx_2 to $S1$, $S2$ respectively, as shown in Fig. 3.

From the above process, we can find that no other Txs uses $Tx_a.row_1$ and $Tx_b.row_1$, and these two outputs are not stored in UTXO pool. Therefore, they can only be the output of end Tx in CTT. Since inputs of Tx_3 comes from other committees, the in-degree of Tx_3 is zero if Tx_3 is in CTT. The above cross-shard transaction feature reduces the CTT generation time overhead of these transactions by about half. This further improves the performance of blockchain.

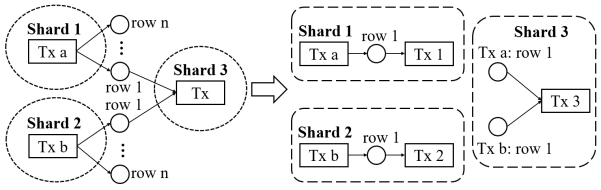

Fig. 3. Cross-shard Transaction

IV. Blockchain Transactions Scheduling (BCTS) Algorithm

This section presents Blockchain Transactions Scheduling (BCTS) algorithm for the SPExTran framework, which consists of several phases, including creating transactions priority queue, generating a group of transactions execution queues, and verifying and executing transaction in parallel.

A. Creating transactions priority queue

This phase aims to create a transaction priority queue (PQ) based on conflicting transactions information table (CTT) in the committee. Conflicting transactions are recorded in the CTT, visualized as Directed Acyclic Graph(DAG), which uses points to denote transactions and edges to denote conflicts of transactions. The main idea is to connect DAGs with virtual nodes and then find the static critical path (SCP) for the whole graph. According to SCP, each transaction in turn recursively invokes the algorithm for the subgraph composed of its ancestor transactions, and is placed in PQ. When SCP traversal is completed, this phase of the algorithm is finished. The notations used in this phase are shown in TABLE I. And the process of this phase is as follows:

TABLE I
NOTATION OF CREATING TRANSACTIONS PRIORITY QUEUE PHASE

Notation	Explanation
n_i	Transaction i in CTT
rt_i	Execution and validation time of transaction i
EST	Transaction Earliest Start Time
STL_i	Transaction i Static Top-level
SCP	Static Critical Path

1) Add a common virtual parent Tx to those with null entry (without father Tx) in CTT. Figuratively, it is represented as a transformation of DAG from multiple connected graphs to a connected graph;
2) For each Tx $\forall n_i$ in CTT, calculate STL_i. Compare STL of transactions with each other, and determine the SCP of DAG in CTT according to the static critical path algorithm;
3) Take the first Tx on SCP as current Tx n_i;
4) If the number of the entry item of n_i is zero, insert n_i into PQ and minus 1 from the entry number of child Tx of n_i. If entry number of n_i is not zero, sort all parent Tx n_j, which has not entered PQ, by size of STL_j+rt_j. Parent Tx and its ancestor Txs, which has not entered PQ, form a DAG subgraph. Recursively call this phase of BCTS algorithm to create a queue for this subgraph, and add it to PQ. Finally, add n_i to PQ and minus 1 from the entry number of child Tx of n_i;
5) If n_i is the last Tx of SCP, this phase ends. Otherwise, take the next Tx of SCP as current Tx n_i and go to 4).

B. Generating a group of transactions execution queues

This phase of the BCTS algorithm converts the transactions priority queue into a group of transactions execution queues.

The main idea is to generate multiple processor idle queues and find the most suitable processor for the transactions in the transaction priority queue in order. After each transaction is scheduled, the CTT and processor idle queue information is updated. The symbols used in this phase are shown in TABLE II, and the process of this phase is as follows:

TABLE II
NOTATION OF GENERATING A GROUP OF TRANSACTIONS EXECUTION QUEUES PHASE

Notation	Explanation
$st(n_i, p)$	Start time of n_i in p processor
$ft(n_i, p)$	Finish time of n_i in p processor
$ft(n_i)$	Run time of n_i
rt_i	Execution and validation time of transaction i
DRT	Data Ready Time
$idle_p(k)$	A certain idle interval on p processor
$st(idle_p(k))$	Starting position of an idle interval on the p processor
$ft(idle_p(k))$	Finishing position of an idle interval on the p processor
$p_{best}(n_i)$	A certain idle interval on p processor
makespan	Scheduling completion time
EST(n_i)	Transaction n_i earliest start time

1) Generate m processor idle queues;
2) Take the first transaction in transaction priority queue as the current transaction n_i;
3) According to the end time of the parent transaction of n_i in CTT, calculate $DRT(n_i)$. n_i can be scheduled to run on $idle_p(k)$ when (1) is satisfied.

$$max\{st(idle_p(k)), DRT(n_i, p)\} + rt(n_i) \leq ft(idle_p(k)) \quad (1)$$

The earliest running time on p processor is calculated according to equation (2);

$$st(n_i, p) = max\{st(idle_p(k)), DRT\} \quad (2)$$

4) When multiple processors are available to schedule the execution of n_i, the best processor is determined according to the priority matching minimum time;

$$p_{best}(n_i) = min\{st(n_i, p)\} \quad (3)$$

5) Update information of processor idle queues;

$$st(idle_{(p_{best})}(k)) = st(n_i, p_{best}) \quad (4)$$

$$ft(idle_{(p_{best})}(k)) = st(n_i, p_{best}) + rt(n_i) \quad (5)$$

6) Calculate EST of n_i and store it in CTT for next phase;

$$EST(n_i) = st(n_i, p_{best}) \quad (6)$$

7) If n_i is the last transaction in PQ, return and end. Otherwise, take the next transaction in PQ as n_i and go to 3).

After analyzing this phase of the BCTS algorithm, it is found that the validation and execution time of the transaction can be calculated using formula (7).

$$makespan = \max_{n_i \in V}\{ft(n_i, p)\} \quad (7)$$

Before running this phase of the BCTS algorithm, the more suitable number of processors, m, can be calculated using formula (8).

$$m_{best} = \frac{\sum rt(n_i)}{max(SCP)} \quad (8)$$

C. Verifying and executing transaction in parallel

In this phase, based on the group of generated transaction execution queues and CTT, a more reasonable and efficient scheduling is carried out for parallel execution. Each processor of the committee node at this phase schedules transactions according to the execution queues group and validates and execute transactions in parallel in conjunction with the CTT. The symbols used in this phase include $input_i$ (the transaction input of n_i in CTT), R (right of transaction), Req (other transactions requesting the result of this transaction). And the specific process of this phase is as follows:

1) Assign execution queues to processors. Each processor validates and executes transactions according to a queue in the execution queue group;
2) Take the first transaction in the queue as the current transaction n_i;
3) Determine whether n_i exists in the CTT: If n_i does not exist in CTT, validate and execute normally and go to 8). Otherwise, go to 4);
4) Take the first Input of n_i as the current Input $input_i$;
5) Search for the position of transaction corresponding to $input_i$ in CTT and determine whether R is 1: If R is 1, $input_i$ is correct and R is set to 0. If R is 0, Req is set to 1 and this processor sleep. Until Req is 0, execute 5);
6) If there is a next input, setting it to the current input and go to 5). if $input_i$ is the last input, go to 7);
7) Verify the remaining inputs of $input_i$, which is not in CTT: if both are successful, n_i is correct, R in Output of n_i are set to 1 and Req in Output of n_i are set to 0. if there is an incorrect input, return error;
8) If n_i is the last transaction in execution queue, return correct and end; Otherwise, take the next transaction as n_i and go to 3);
9) If all processors return correct, the verification result is right. Otherwise, the verification result is error.

According to the transaction verification process of BCTS algorithm, transactions with dependencies are quickly verified by CTT instead of entering the UTXO pool for verification, which saves a lot of time. Transactions without dependencies normally enter the UTXO pool for validation. Block transactions are executed in parallel by multiple processors of committee member nodes. The Right and Req values are set in the CTT table shared by multiple processors and a locking

mechanism is used to ensure the correctness of the transaction validation process.

The CTT mechanism greatly reduces the overhead of the validation process for conflicting transactions. Because UTXO pools are historical data on the blockchain, the lookup and verification process takes a lot of time. However, the processing overhead with the CTT mechanism is much smaller and negligible. Transaction sets without dependencies can be executed in parallel on different processors directly, and the CTT mechanism reduces the verification overhead of transaction sets with dependencies, thus achieving dynamic performance improvement.

V. Evaluations

A. Settings

Experiments were conducted to evaluate the performance of SPExTran framework and BCTS algorithm. The communication latency between nodes with different committee sizes and the reconfiguration latency for committees with different network sizes are set as shown in TABLE III.

TABLE III
COMMUNICATION OVERHEAD WITH DIFFERENT COMMITTEE SIZES AND NETWORK SIZES

Committee size in number of nodes	latency (ms)	Network size in number of nodes	latency (ms)
10	275	100	135
50	275	500	200
100	300	1000	240
200	325	2000	250

The effect of 4 sharding parameters on blockchain throughput are evaluated, including Txs conflict rate, parallelism, block size(the number of transactions per block), committee size(the number of nodes). The default values are shown in TABLE IV.

TABLE IV
DEFAULT VALUES

Parameter	Value
Txs conflict rate	30%
parallelism	12
block size	200Txs/Block
committee size	10*10[a]

[a]10 committees, each one consist of 10 nodes.

The experiment is divided into two phases, pre-processing phase and execution phase. And execution phase consists of multiple consensus phases and committee reconfiguration phases. In the pre-processing phase, enough transactions will be generated to simulate stabilized blockchain. Firstly,100,000 wallet addresses are randomly generated and assigned to 10,000 users. And 10,000 Txs are created based on the wallet address.Then, the corresponding unused UTXOs are created and stored in the UTXOs pool of each committee.In execution phase, a total of 10,000 blocks are generated by the committees. Cross shard Txs are assigned according to the rules proposed in session III.

B. Experimental Results

The performance of sharding blockchain is evaluated through traditional sequential transaction execution and parallel transaction execution based on the SPExTran framework and BCTS algorithm.The throughput of the two methods is compared against the default values of the parameters. As shown as Fig. 4, based on the SPExTran framework and BCTS algorithm, the throughput of the blockchain has been significantly improved by about 46%. The reduction in throughput is due to the fact that the process of generating blocks creates transactions that make the UTXO pool increase relative to the initial state. It is inevitable when the experimental data are not large enough in simulation experiments.

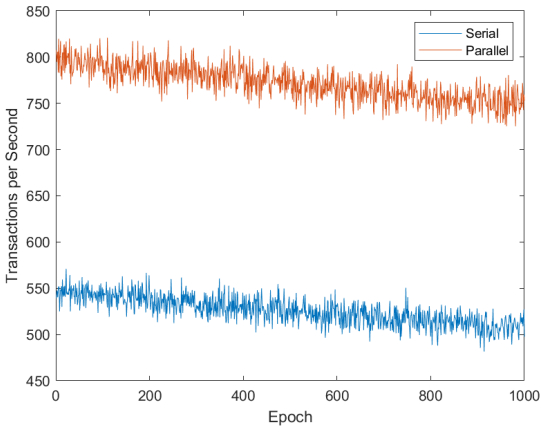

Fig. 4. Throughput with default parameters

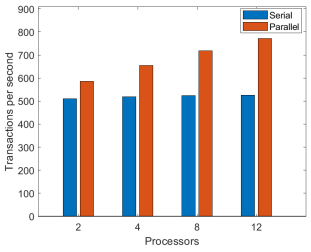

Fig. 5. Throughput for various processors in blockchain with serial and parallel execution

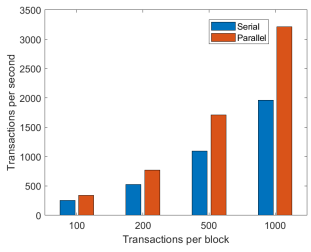

Fig. 6. Throughput for various transactions in blockchain with serial and parallel execution

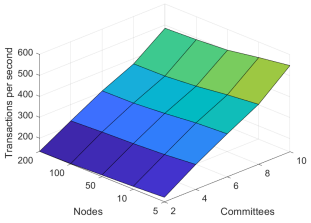

Fig. 7. Throughput for various committees and nodes in blockchain with serial execution

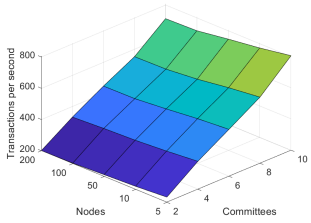

Fig. 8. Throughput for various committees and nodes in blockchain with parallel execution

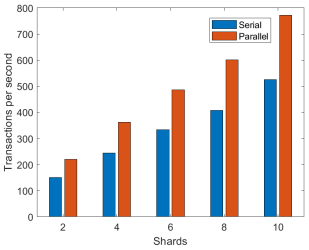

Fig. 9. Throughput for various shards in blockchain with serial and parallel execution

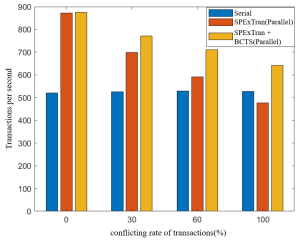

Fig. 10. Throughput for various conflicting rate in blockchain with different method

In addition, the effect of the parameters on the proposed parallel scheme is evaluated. Fig. 5 shows the results for different degree of parallelism. It can be observed that the number of processors has little effect on sequential execution, while for parallel execution, the throughput increases as the number of processors increases. With the increase of processors, the proposed approach provides more significant performance improvements. Fig. 6 shows the results of different block size. It can be observed that the throughput of both methods increases with the increase of block size, but the proposed method has a higher growth rate. Fig. 7-8 shows the results of different committee sizes. It is observed that throughput increases as the number of committees increases. However, throughput is not sensitive to changes in the number of nodes per committee. Therefore, the number of nodes per committee is set to 10 and the impact of the number of shards is explored, as shown in Fig. 9. Although the throughput of both models increases with the number of committees, all performances are improved by about 46% with the proposed method. Fig. 10 shows the contribution of BCTS algorithm under CTT mechanism. At any conflicting rate, the parallel execution of the proposed SPExTran+BCTS scheme is more effective than the traditional serial execution, which is also impossible to achieve using the parallel execution of the SPExTran framework only. Since when the conflicting rate is 100%, there is no parallelism, but more operations need to be done.

VI. Conclusions

This paper proposes a framework of Sharding based Parallel Execution of block Transactions in Blockchain (SPExTran) and an associated Blockchain Transactions Scheduling (BCTS) algorithm to improve transaction execution efficiency by converting sequential execution into parallel execution. Extensive experimental results show that SPExTran and BCTS substantially improve scalability, significantly reduce the time overhead at the execution layer, and improve the verification of conflicting transactions with CTT mechanism. SPExTran and BCTS can be applied to any existing sharding consensus protocol in a complementary manner.

References

[1] Hafid A, Hafid A S, Samih M. Scaling blockchains: A comprehensive survey[J]. IEEE Access, 2020, 8:125244-125262.

[2] Yin M, Malkhi D, Reiter M K, et al. Hotstuff: Bft consensus with linearity and responsiveness[C]//Proceedings of the 2019 ACM Symposium on Principles of Distrib-uted Computing. 2019:347-356.

[3] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. Rapidchain: Scaling blockchain via full sharding. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 931–948.

[4] Silvano W F, Marcelino R. Iota tangle: A cryptocurrency to communicate internet-of-things data[J]. Future Generation Computer Systems, 2020, 112:307-319.

[5] Zhang J, Ye Y, Wu W, et al. Boros: Secure and Efficient Off-Blockchain Transactions via Payment Channel Hub[J]. IEEE Transactions on Dependable and Secure Computing, 2021.

[6] Ye Y, Ren Z, Luo X, et al. Garou: An efficient and secure off-blockchain multi-party payment hub[J]. IEEE Transactions on Network and Service Management, 2021, 18(4): 4450-4461.

[7] Lin S, Zhang J, Wu W. FSTR: funds skewness aware transaction routing for payme-nt channel networks[C]//2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2020: 464-475.

[8] Nick J, Poelstra A, Sanders G. Liquid: A bitcoin sidechain[J]. Liquid white paper. URL https://blockstream.com/assets/downloads/pdf/liquid-whitepaper. pdf, 2020.

[9] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A secure sharding protocol for open blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 17–30.

[10] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. Rapidchain: Scaling blockchain via full sharding. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 931–948.

[11] Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., and Ford, B. Omniledger: A secure, scale-out, decentralized ledger via sharding. In 2018 IEEE Symposium on Security and Privacy (SP) (2018), IEEE, pp. 583–598.

[12] Gao Y, Kawai S, Nobuhara H. Scalable blockchain protocol based on proof of stake and sharding[J]. Journal of Advanced Computational Intelligence and Intelligent Informatics, 2019, 23(5): 856-863.

[13] Amiri M J, Agrawal D, El Abbadi A. Sharper: Sharding permissioned blockchains over network clusters[C]//Proceedings of the 2021 International Conference on Management of Data. 2021: 76-88.

[14] Nguyen L N, Nguyen T D T, Dinh T N, et al. Optchain: optimal transactions placement for scalable blockchain sharding[C]//2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019: 525-535.

[15] Wang J, Wang H. Monoxide: Scale out blockchains with asynchronous consensus z-ones[C]//16th USENIX symposium on networked systems design and implementation (NSDI 19). 2019: 95-112.

[16] Reijsbergen D, Dinh T T A. On exploiting transaction concurrency to speed up blockchains[C]//2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2020: 1044-1054.

[17] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. Decentralized Busi-ness Review, 2008: 21260.

[18] Wood G. Ethereum: A secure decentralised generalised transaction ledger[J]. Ether-eum project yellow paper, 2014, 151(2014): 1-32.

[19] Luu L, Teutsch J, Kulkarni R, et al. Demystifying incentives in the consensus computer[C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 2015: 706-719.

[20] Amiri M J, Agrawal D, El Abbadi A. Parblockchain: Leveraging transaction parallelism in permissioned blockchain systems[C]//2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019: 1337-1347.

[21] Baheti S, Anjana P S, Peri S, et al. DiPETrans: A framework for distributed parallel execution of transactions of blocks in blockchains[J]. Concurrency and Computation: Practice and Experience, 2022, 34(10): e6804.

[22] Li Y, Liu H, Chen Y, et al. FASTBLOCK: Accelerating Blockchains via Hardware Transactional Memory[C]//2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS). IEEE, 2021: 250-260.