JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 15, No. 3, September 2021, pp. 125-133

Leverage Sidechains to Reduce the Workload of Smart Contracts through Parallelization

Magne Saetran, Jungwon Seo, and Sooyong Park*
Department of Computer Science and Engineering, Sogang University, Seoul, Korea
magnesaet@gmail.com, jungwonrs@gmail.com, sypark@sogang.ac.kr

Abstract
Recently, blockchain has been evolving rapidly with new innovations, coins, and use cases every day. The platform is getting more congested due to the increase of interest and mainstream adoption, and because most of this activity is on the widely used Ethereum blockchain. This also further increases the costs of using the platform, as the gas prices become higher. Smart contracts are deployed on the Ethereum blockchain and the high usage of these smart contracts is one of the main reasons behind the congestion. We propose a new scheme using sidechains and a middleware to reduce the workload for certain smart contracts, which allows the execution of smart contract transactions in parallel through sidechains. In this way, the sidechains could be leveraged to decrease the congestion on the main blockchain and increase the rate of transactions per second. Furthermore, the sidechains could have their settings, like block time and block gas limit, adjusted to give more optimal results. We implemented a modified version of the ballot contract from the solidity documentation, and our results demonstrated that through the use of two sidechains, the transactions processed per second could be increased from 1.8× to 13.0×, depending on the sidechains settings.

Category: Smart and Intelligent Computing

Keywords: Blockchain; Smart contracts; Scalability; Sidechains; Parallelization

I. INTRODUCTION

Blockchain technology was first introduced in 2009 under the pseudonym of Satoshi Nakamoto in the Bitcoin context [1], a cryptocurrency made with the purpose of being decentralized and digitally transferable. A blockchain can be described as a decentralized, distributed, append-only ledger, and by design is transparent, immutable, and secure. In layman's terms, it is a growing list of blocks with recorded information that are connected through cryptography [2].

Ethereum was introduced in 2015 by Buterin [3], and was the first blockchain with smart contract functionality. A smart contract is a self-executing program, which executes according to terms and conditions programmed in the contract. It enables the execution of programmable applications in a trustless blockchain-based environment.

The attributes of blockchain technology being seen as promising, along with the potential of smart contracts, this technology has continued to evolve with mainstream adoption and usability in solutions for everyday scenarios. However, currently, the industry is expanding faster than the technology can sustain. In the Ethereum blockchain, the most widely used decentralized blockchain network and smart contract platform, the frequency of daily transactions has almost doubled in a year, i.e., from about 700,000 transactions to about 1.4 million transactions from April 2020 to April 2021 [4]. On the other hand, the

Open Access http://dx.doi.org/10.5626/JCSE.2021.15.3.125 http://jcse.kiise.org

Received 02 June 2021; **Accepted** 02 September 2021
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

Ethereum blockchain has a limit of about 15–18 transactions per second [5] and it remained unchanged despite the increased rate of transactions. This is a bottleneck considering today's demand and hinders further growth of the technology. Furthermore, the excessive number of transactions the blockchain cannot handle congest the network, which subsequently increases the gas price [6]. Moreover, a few of the popular smart contracts seems to be responsible for the majority of network traffic [7].

With DeFi, NFT, and more potential applications being developed daily, the need for scalability is essential for the sustainability of the technology. Applications on the Ethereum blockchain are deployed as smart contracts, hence with the current transaction volume and network congestion, users would have to pay more money for each interaction with these applications along with longer wait times for their transactions to be confirmed. Methods, such as, reducing the block generation time period or increasing the block gas limit, have been proposed with potential benefits when it comes to throughput. However, these have drawbacks which can make the public blockchain network security more vulnerable [8].

There are various researches on improving the scalability of blockchain technology [9-14]. Some works proposed the use of speculative parallel execution for smart contracts by borrowing techniques from software transactional memory. While these solutions seem promising, due to the state-of-the-art of Ethereum and Ethereum Virtual Machine (EVM, the EVM is single-thread), the solutions cannot be implemented in practice. Other works tried to implement sharding, a technique to spread the load over several shards, enabling parallel execution. However, sharding is still in the very early stage, as the required changes, like changing the consensus protocol, are very difficult to implement in practice. Similarly, some works tried to implement sidechains and atomic cross-chain transactions, allowing transactions between sidechains and all data from the different sidechains to be used in an agnostic manner. However, several changes would have to be implemented for this to work in practice, and further theoretical and practical testing would be needed.

In this paper, we propose a method for certain smart contracts to be executed in parallel through the usage of sidechains, a secondary blockchain connected to the main blockchain. The sidechains are connected and communicate with the main chain through a middleware. The middleware is used to communicate and relay information between the sidechains and the main chain. Executing the smart contracts on the sidechains, rather than on the main chain, allows a reduction of the workload for the main chain, in addition to enabling higher throughput of transactions through the usage of parallelism.

This paper contributed in the following:

- a way to reduce the workload for smart contracts on a blockchain by leveraging sidechains;
- a way to increase the throughput of transactions by using sidechains for parallel execution, and also obtain greater results through adjusting the settings of the sidechains;
- a prototype implementation using Hyperledger Besu, with two sidechains and a middleware. An evaluation shows that using a slightly modified version of the ballot smart contract from solidity's documentation yields speed up ranging from 2× to 13× depending on the settings of the sidechains.

The rest of the paper is structured as follows: The background for the research is described in Section II; Section III describes related works and studies that also tried to solve scalability problems of the Ethereum blockchain; our approach is presented in Section III; Section V describes the implementation of our approach and our evaluation of the experimental results; Section VI presents our conclusion.

II. BACKGROUND

In this section, we introduce blockchain technology, along with Ethereum and EVM. Additionally, we introduce how smart contracts work and what are sidechains.

A. Blockchain Technology

A blockchain can be described as a decentralized, distributed, digital, and append-only ledger. It consists of blocks which are identified by their cryptographic hash and are linked like a chain, as each block contains the cryptographic hash of the previous block. Furthermore, each block holds a timestamp and a set of transactions represented as a Merkle tree root hash in the block. By design, the blockchain is immutable as retroactively modifying transaction data of a block would involve altering all subsequent blocks as well. Typically, a blockchain operates as a peer-to-peer system, where the nodes which hold a copy of the blockchain find a consensus through a consensus protocol. Consensus protocol refers to the sets of rules that ensure all participants agree on what is added to the blockchain network. In both Bitcoin and Ethereum's cases, the consensus protocol is called proof-of-work (PoW) and the nodes use computation power to compete for solving a hash puzzle, where the winner is given a reward in the form of the native currency of the blockchain.

There exist public and private blockchains, in the public blockchain everyone can access and take a part in the consensus, while in the private blockchain, one needs the authority to be able to interact with the blockchain. In permissioned blockchains, consensus protocols, other than PoW, are often used, as it is deemed secure even in a permissioned blockchain environment but provide other

benefits like faster transactions per second (TPS). Depending on the purpose of the blockchain, the permission level can vary as well as the consensus protocol.

B. Ethereum, EVM, and Smart Contracts

Distinct from the Bitcoin blockchain, the Ethereum blockchain manages a long-lived state. The Ethereum blockchain is a single state that is shared given the blockchain's lack of a central authority. The Ethereum blockchain currently uses a PoW consensus protocol, with miners and validators. Miners are responsible to package transactions into blocks, and through the consensus protocol, agree on which block to append next onto the blockchain. Validators ensure that the blockchain state remains correct. The Ethereum blockchain has a block time, the time between each generated and proposed block, of approximately 15 seconds, while the gas limit, the maximum amount of gas that can be processed per block, is 12.5 million. This gives Ethereum an average TPS ranging from 15 to 18.

In Ethereum there are two types of accounts: externally owned account (EOA) and contract account (CA). EOAs represent users who interact with the blockchain through transactions, and CAs represent smart contracts. Transactions in Ethereum refer to messages that are sent from an EOA to another account on the Ethereum blockchain. An EOA can send Ether to another EOA, which is a transfer of asset. If the receiving account of a transaction is a CA, the smart contract is executed according to the function and conditions specified in the message. A CA can also execute a transaction to another CA, if programmed so.

A smart contract is an executable program which is run on an emulated computer, wherein the Ethereum's case is the EVM. The EVM is responsible for compiling high-level smart contract code to EVM bytecode before executing the code. Solidity [15] is the most popular high-level programming language for smart contracts. Smart contracts manage a long-lived state, which is recorded on the blockchain, and this state is managed through smart contract functions. We can classify smart contract functions into two types: write functions, which refer to functions that modify the state, and read-only functions, which only read the state values and thus no state changes are carried out. To execute a smart contract, a fee must be paid in the form of gas, which is the necessary costs to perform a transaction. Only the transactions that modify the states incur a gas cost, and the more complicated the smart contract transaction is, the higher the cost is. In the Ethereum blockchain, the gas fee is paid through the use of the blockchain's native token, Ether (ETH). This functionality refers to the EVM as a quasi-Turing-complete state machine, as there is a gas limit to prevent delays caused by huge transactions cost, also known as the halting problem. The EVM has no scheduling capabilities, thus making it behave as single-threaded since the execution order is organized externally through the nodes.

C. Sidechains

Sidechains are secondary blockchains which are connected to the main blockchain, also known as the main chain. Commonly, sidechains are connected to the main chain through two-way pegs, a mechanism which allows the bidirectional transfer of native digital currencies between the chains [16]. Sidechains can have a different protocol or consensus algorithm from the main chain, which can provide benefits when it comes to flexibility and functionalities.

III. RELATED WORK

This section presents other studies which have tried to solve the current scalability problem of the Ethereum blockchain.

Dickerson et al. [9] proposed a speculative execution model allowing smart contracts to be executed in parallel by adapting techniques from software transactional memory. Miners execute smart contract transactions in parallel and record a fork-join schedule that allows validators to re-execute the transactions following the same schedule to ensure a correct final state. Abstract locks and inverse logs are used to prevent conflict of transactions that try to modify the same shared states, with rollback used in the case of a conflict. Anjana et al. [10] further extended this approach, which do not require locks and rollback, by using basic time stamp ordering and multi-version time stamp ordering. However, as the current EVM is single-threaded, the prototypes were implemented in Java virtual machine to run simulations for experimental results. Therefore, it would not be possible to implement the solution in practice with the current state-of-the-art of the Ethereum blockchain and EVM.

Zamani et al. [11] have proposed RapidChain, a way to incorporate sharding into blockchain technology. Sharding is a technique to split a database in order to spread the load. In a blockchain context, sharding divides the participants of a blockchain network into subgroups called shards, which enable transactions to process in parallel. However, given that Ethereum uses a PoW consensus protocol, sharding reduces the security of the blockchain as the shards need to communicate with one another. In RapidChain, they propose a new consensus protocol that enables high throughput of transactions while keeping the security level high, but the system is limited to cryptocurrency applications, and so cannot be used in a smart contract environment. Furthermore, changing the consensus protocol of Ethereum is extremely difficult to implement due to its decentralized nature. Ethereum is also currently working on incorporating sharding [12],

but is in the very early phases of a years-long project.

It is generally perceived that Back et al. [13] proposed the first sidechain technology with pegged sidechains, where transactions can move coins from one chain to another. However, when transferring from one chain to another it incurs high latencies which could become a bottleneck. Robinson et al. [14] proposed atomic cross-chain transactions, allowing transactions to be executed atomically across the sidechains, including smart contract transactions. This enabled data in one sidechain to be used by other sidechains. They introduced a new transaction locking mechanism and a new mechanism for proving values across sidechains. For this solution to be properly implemented, both application designers and developers need to adjust given the programming model and guidelines, and further testing is needed to determine the theoretical and practical performance.

IV. APPROACH

We propose a method of leveraging sidechains to reduce the workload of certain smart contracts. Our aim was to showcase a method where throughput benefits could be gained given the current state-of-art of the Ethereum blockchain.

Our architecture consists of several sidechains and a middleware (Fig. 1). As the EVM is characteristically single-threaded, transactions must be executed serially, which consequently limits the number of transactions which can be processed per second. In our approach, we took advantage sidechains to execute smart contract transactions in parallel, as sidechains are separate blockchains with inherent EVMs.

A. Sidechains and Middleware

The sidechains are responsible for executing the smart contract functions. Normally, users of decentralized applications on the Ethereum blockchain would interact with the blockchain by sending transactions to the smart contracts. As the blockchain can only process a limited number of transactions per block, the higher the transaction volume, the more the network is congested. Thus, by leveraging the sidechains to execute the smart contracts, the congestion of the network and a further increase in gas prices could be prevented. The same smart contract was deployed on all the sidechains and the main chain. By utilizing read-only smart contract functions, we could extract data from the smart contracts on the sidechains cost-free. The extracted data could be sent to the middleware which could then send it to the main chain.

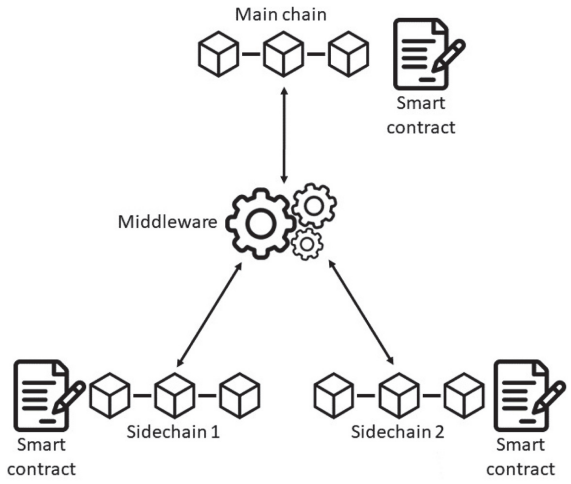

Fig. 1. Architecture of approach.

The middleware is responsible for communicating between the sidechains and the main chain. Depending on the type of smart contract and purpose, the middleware has to adapt and be programmed differently. Furthermore, the smart contracts need to be changed according to the usage of the middleware as well. The middleware can extract data from sidechains, manipulate the data if needed, and send the data to the main chain. However, sending data to a blockchain is done through write functions and so requires a transaction with a gas fee. Therefore, designing the middleware so as to use less write function transactions could be beneficial. Given that the middleware is somewhat centralized, one may wonder if it is secure to use. However, as blockchains are transparent, it is easy to check what is extracted and sent from the middleware.

The first step in our approach was to modify the smart contracts so that they are compatible with our approach. We modified the smart contracts to involve commutative functions if they do not already contain so. Commutative functions refer to functions that are not dependent on other functions. The reason for this is twofold. First, commutative functions are needed to increase the number of transactions involving the same smart contract per block. If two transactions tried to modify the same variable in the same block, then the miner who packaged the transactions into the block was the determiner of which of the transactions was executed last, and thus what final value the variable will be modified to. Fig. 2 showcases a scenario where two users tried to set the same variable x to different values. However, it was the miner who was the one responsible for the transactions

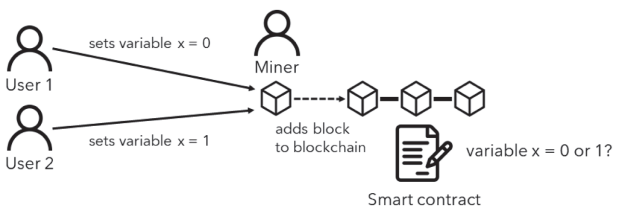

Fig. 2. Example of a non-commutative function.

which were executed last, and so it was not possible to know beforehand which values the variable would end up being set to. By utilizing commutative functions, smart contract functions could be called without this worry of data conflict. Second, to enable the use of sidechains, the smart contract functions should be commutative, so that the values from the sidechains could be combined and not conflict with one another.

Next was to deploy the smart contract to all of the blockchains. After deployment, users could interact with all of the smart contracts. To reduce the workload on the mainchain, users were encouraged to utilize the contracts on the sidechains. After users have submitted transactions to the sidechains, the middleware was responsible to extract the data from the sidechains to the main chain. Finally, the results could be seen in the main chain.

As aforementioned, due to security risks, the block generation time period or the block gas limit of public blockchain should not be meddled with. However, for permissioned blockchains, all the participating nodes in the network are known, making the security tighter. Thus, permissioned sidechains could be utilized to further increase the potential throughput by adjusting the sidechains' settings for optimal results. For example, the block generation time period could be lowered and the block gas limit could be increased, which would increase the number of transactions executed per second.

C. Ballot Contract Example

The approach will be demonstrably explained in the context of the ballot contract from Solidity's documentation [17].

Our architecture consists of a main chain, several sidechains, and a middleware. To exemplify our approach, we used two sidechains. Fig. 3 shows the workflow of our approach and Algorithm 1 shows the pseudo code. The ballot smart contract was deployed to all the blockchains. Rather than voting on the main chain, the voting transactions were carried out on the sidechains in parallel, as the voting function was commutative. After a set condition or a certain number of votes, the chairperson could extract the votes from the sidechains to the middleware using a read-only function. The middleware adds up the proposals' votes from each smart contract and the chairperson could send transactions to the main chain, where the number of votes would be added to the corresponding proposal. In the case of the voting contract, the number of transactions to the main chain was proportional to the number of proposals, thus with two proposals, only two transactions to the main chain were needed. In other applications the middleware's task might vary, therefore it should be implemented depending on the smart contract.

Algorithm 1 [Ballot contract approach] Workflow

```
for voters do
    if delegated voting rights & not voted then
        vote on proposal on sidechain
    end if
end for
for proposals on sidechains do
    if sender = chairperson then
        return vote count for proposal
    end if
end for
for votes in proposals do
    add votes to proposal's total vote count
end for
for proposals on mainchain do
    if sender = chairperson then
        send total vote count
    end if
end for
```

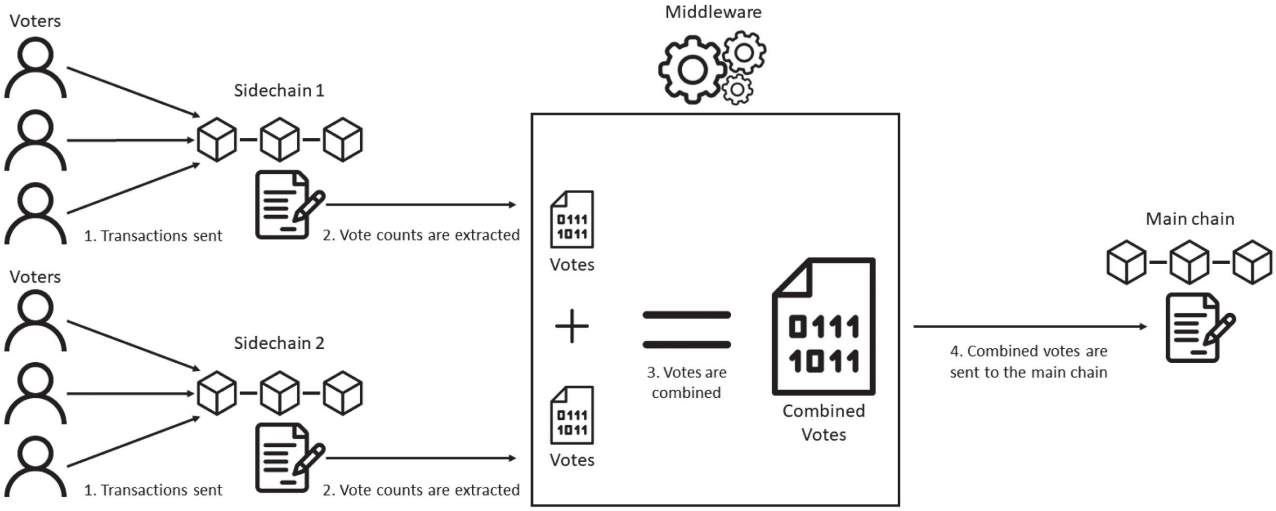

Fig. 3. Workflow of the usage of sidechains and middleware to execute the ballot smart contract.

V. IMPLEMENTATION AND EVALUATION

In this section, we present our experiment results designed to evaluate our approach. For evaluating the approach, we established the three research questions below and carried out three experiments in response to the research questions.

- RQ1: How much can the throughput be increased using two sidechains compared to a single blockchain?
- RQ2: How much can the throughput be increased using two sidechains, where the sidechains have lower block time periods?
- RQ3: How much can the throughput be increased using two sidechains, where the sidechains have a higher block gas limit, and how does this change depend on the different block time?

A. Ballot Smart Contract

The ballot contract implements a simple voting system with automatic vote counting and delegation of voting rights. Voters can vote between proposals that are set at the creation of the contract and the smart contract ensures that no voter can vote twice. The voting function in the smart contract is commutative, allowing several transactions invoking the voting function to be executed in the same block, provided that the transactions come from different accounts. The creator of the smart contract is referred to as the chairperson and is responsible for delegating the rights to vote. Additionally, there is a function in the smart contract that returns the winning proposal after counting the votes. We modified the contract by adding two functions: a function that enables the chairperson to add a number of votes to a proposal, and a read-only function that enables the chairperson to extract a proposal vote count.

B. Experimental Setup

We implemented a simple prototype with two sidechains and one main chain using Hyperledger Besu [18]. Hyperledger Besu is an Ethereum-based blockchain designed for both public and private permissioned network use cases. It was run with the consensus protocol IBFT 2.0. The experiments were run on a Microsoft Azure Virtual Machine with 64 GiB ram and 16 vCPUs. We implemented the smart contract in solidity and used a modified version of the ballot contract from Solidity's documentation.

Several experiments were conducted. The main chain ran constantly with a 15s block generation period time and a 12.5 million block gas limit. These settings were chosen to simulate the current settings of the Ethereum blockchain. The sidechains were tested with different block generation time periods and block gas limits. The block generation time period setting varied between 15s, 10s, 5s, and 2s, while the block gas limit setting was between 12.5 million and 25 million. All of the experiments were performed for 100 rounds and the average results were calculated. A total of 10,000 accounts were created randomly and given enough ether to send transactions, allowing 10,000 transactions from 10,000 different accounts to be sent.

Transactions were continually sent, so that the transaction pool would fill up, allowing transactions to be taken directly from the transaction pool to be packed into the next block for execution. For voting, we set up the ballot contract with two proposals, and then we made the accounts randomly vote for one of the proposals when sending a voting transaction. Prior to the start of the experiment, the rights to vote were delegated to the accounts. After executing each round of the experiments, we verified that the winning proposal was correct given the vote counts.

C. Benchmarks

For comparison, we analyzed a single blockchain to see how long it would take to execute 10,000 voting transactions. The results in Fig. 4 showed that executing 10,000 transactions on average took about 520.6 seconds, thus giving us a TPS of 19.2. This gave us a TPS which was similar to the Ethereum blockchain and served as a good indicator for how our approach would work. This was referred to as the current scheme.

D. RQ1: How much can the throughput be increased using two sidechains compared to a single blockchain?

For RQ1, we conducted an experiment to see how our approach using two sidechains and a middleware, while keeping the block time and block gas limit same as

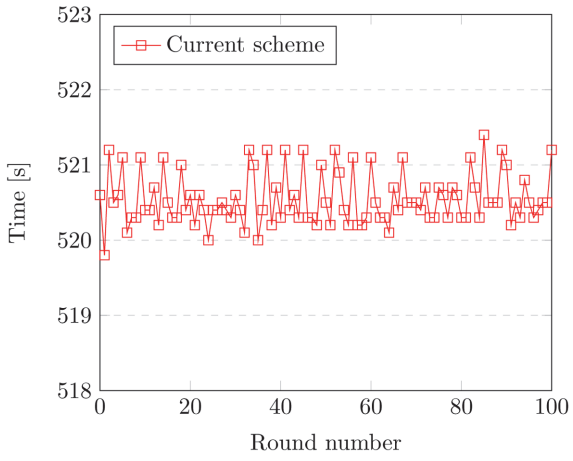

Fig. 4. Average execution time for 10,000 voting transactions on a single blockchain, conducted 100 rounds.

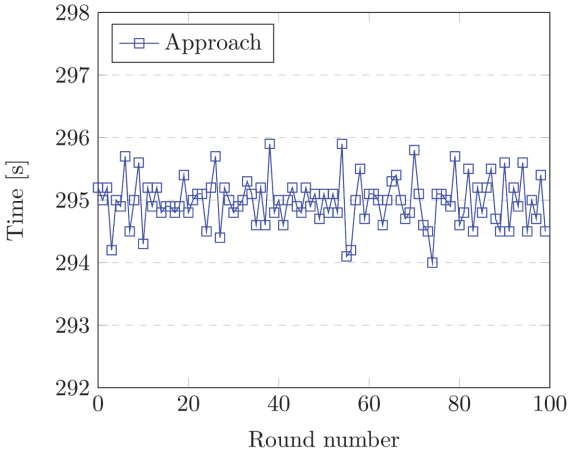

Fig. 5. Average execution time for 10,000 voting transactions using our approach with 15s block time and 12.5 million gas limit, conducted 100 rounds.

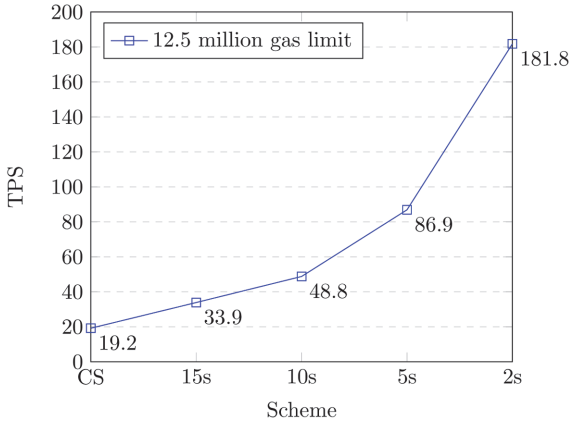

Fig. 6. Average TPS for the dierent block times of 15s, 10s, 5s, and 2s, compared with the current scheme.

Ethereum, would compare against the current scheme. A total of 10,000 voting transactions were executed, 5,000 on each sidechain, as well as the extraction of voting data and sending the summed voting data to the main blockchain.

Fig. 5 shows the execution time for the 100 rounds, giving an average of 295.0 seconds to execute all the transactions. This yielded a TPS of about 33.9 transactions per second and a speedup of 1.8 when compared to the current scheme. Thus, by utilizing our approach, we could gain a throughput advantage compared to the current scheme, even with the small overhead of extracting voting data and sending them to the main chain. Furthermore, these results demonstrated where the main chain was not executing any transactions. This means that while the sidechains were executing the voting transactions, the main chain could simultaneously execute smart contract transactions as well, whether it could be the same smart contract or other applications, potentially giving a further increase in speedup. Additionally, more sidechains could be added if needed, facilitating more transactions to be executed in parallel for even greater results.

E. RQ2: How much can the throughput be increased using two sidechains, where the sidechains have lower block times?

We tested our approach with 15s block time, similar to Ethereum, in RQ1, and further extended this by conducting experiments with 10s, 5s, and 2s block times. Fig. 6 presents our results in TPS, where the 10s, 5s, and 2s block times results in a TPS of 48.8, 86.9, and 181.8, respectively. This gave a speedup of 2.5, 4.5, and 9.5 as compared to the current scheme. With the lower block times, the limited number of transactions per block could be processed at a quicker interval, resulting in a higher throughput of TPS. These results do not take into consideration the main chain of execution of other transactions.

F. RQ3: How much can the throughput be increased using two sidechains, where the sidechains have a higher block gas limit, and how does this change depend on the different block times?

Lastly, we conducted experiments with a 25 million block gas limit, doubling that of the Ethereum blockchain, and also tested it with each of the different block times in RQ2. The results in Fig. 7 showed that the TPS was 69.0, 100.0, 142.9, and 249.8, for 15s, 10s, 5s, and 2s, respectively. This gave a speedup of 3.6, 5.2, 7.4, and 13.0 when compared to the current scheme (Table 1).

We could observe that by doubling the block gas limit, the throughput likewise doubled for the higher block times with a speedup of 2.0 and 2.1 for 15s and 10s, respectively, when comparing with the results from RQ2. However, for the lower block times, the speedup was 1.6

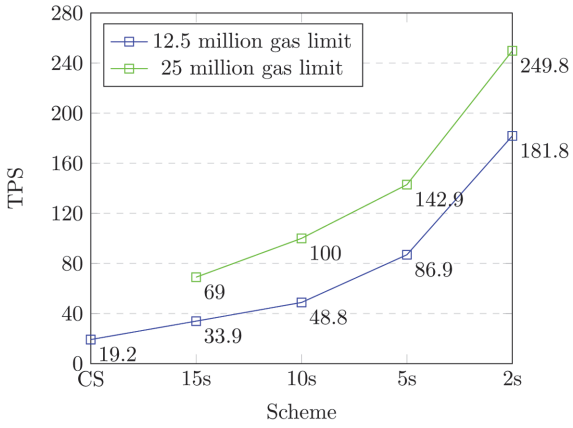

Fig. 7. Comparison of average TPS for the current scheme and dierent block times of 15s, 10s, 5s, and 2s, with 12.5 and 25 million gas limits, respectively.

Table 1. Comparison of results to contract account

Block time (s)	TPS	Gas limit (million)	Speedup (CS)
15	33.9	12.5	1.8x
15	69.0	25	3.6x
10	48.8	12.5	2.5x
10	100.0	25	5.2x
5	86.9	12.5	4.5x
5	142.9	25	7.4x
2	181.8	12.5	9.5x
2	249.8	25	13.0x

and 1.4 for 5s and 2s, respectively, indicating that increasing the gas limit might not be as effective at lower block times. A reason for this might be that with the lower block times, there was not enough time to execute an adequate number of transactions to fill up the higher gas limit, but further experimentation might be needed before this conclusion could be properly established.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a solution by utilizing sidechains in parallel to gain speedup and scalability advantage regarding smart contracts. This solution tried to improve the scalability and to prevent further congestion of the Ethereum blockchain network. To demonstrate our solution, a slightly modified version of the ballot smart contract was used. The reason for modification was to ensure that commutative functions were used, and to add functions to extract and add data to the smart contract. The steps for our approach enabled users to interact with the smart contracts in sidechains, allowing parallel execution, before a middleware extracted the data from the sidechains, using read-only functions. Then, the middleware could send these extracted data to the main chain, thus allowing the main chain to get the data without having to execute the transactions. This prevented congestion in the main chain and allowed a faster throughput of transactions.

Based on this approach, we carried out three experiments to respond to the corresponding research questions. The results showed that a speedup from 1.8 up to 13.0 could be obtained depending on the block time and gas limit of the sidechains. In addition, more sidechains could be added, allowing the approach to scale for even greater results.

The next step to determine weaknesses of the research would be the implementation of the solution for several different applications and use cases. Moreover, figuring out how to design smart contracts to involve more commutative functions and testing the approach with several different smart contracts simultaneously would help to advance the research.

ACKNOWLEDGEMENTS

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (No. IITP-2021-2017-0-01628) supervised by IITP (Institute for Information communication Technology Promotion).

REFERENCES

1. S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008 [Online]. Available: https://bitcoin.org/en/bitcoin-paper.
2. A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton, NJ: Princeton University Press, 2016.
3. V. Buterin, "A next-generation smart contract and decentralized application platform," 2014 [Online]. Available: https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf.
4. Etherscan, "Ethereum Daily Transactions Chart," 2021 [Online]. Available: https://etherscan.io/chart/tx.
5. Blockchair, "Ethereum chart: transactions per second," 2021 [Online]. Available: https://blockchair.com/ethereum/charts/transactions-per-second.
6. Etherscan, "Ethereum Average Gas Price Chart," 2021 [Online]. Available: https://etherscan.io/chart/gasprice.
7. V. Saraph and M. Herlihy, "An empirical study of speculative concurrency in Ethereum smart contracts," 2019 [Online]. Available: https://arxiv.org/abs/1901.01376.
8. I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: a scalable blockchain protocol," in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Santa Clara, CA, 2016, pp. 45-59.
9. T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, "Adding concurrency to smart contracts," *Distributed Computing*, vol. 33, pp. 209-225, 2020.
10. P. S. Anjana, S. Kumari, S. Peri, S. Rathor, and A. Somani, "An efficient framework for concurrent execution of smart contracts," 2018 [Online]. Available: https://arxiv.org/abs/1809.01326.
11. M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Toronto, Canada, 2018, pp. 931-948.
12. Ethereum wiki, "On sharding blockchains FAQs," 2020 [Online]. Available: https://eth.wiki/sharding/Sharding-FAQs.
13. A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timon, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," 2014

[Online]. Available: https://blockstream.com/sidechains.pdf.
14. P. Robinson, D. Hyland-Wood, R. Saltini, S. Johnson, and J. Brainard, "Atomic crosschain transactions for Ethereum private sidechains," 2019 [Online]. Available: https://arxiv.org/abs/1904.12079.
15. Solidity, "What is Solidity?," 2021 [Online]. Available: https://docs.soliditylang.org/en/latest/index.html.
16. A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K. K. R. Choo, "Sidechain technologies in blockchain networks: an examination and state-of-the-art review," *Journal of Network and Computer Applications*, vol. 149, article no. 102471, 2020. https://doi.org/10.1016/j.jnca.2019.102471
17. Solidity, "Solidity by Example," 2021 [Online]. Available: https://docs.soliditylang.org/en/latest/solidity-by-example.html.
18. Hyperledger Besu [Online]. Available: https://www.hyperledger.org/use/besu.

Magne Saetran

Magne Saetran received M.S. degree in Computer Science and Engineering from Sogang University, Korea in August 2021, and is currently working as a software engineer at EMBlock Inc. His research interests are in blockchain and smart contracts.

Jungwon Seo

Jungwon Seo received M.S. degree in Computer Science and Engineering from Sogang University, Korea in March 2020, and is currently pursuing a Ph.D. degree. His research interests are in blockchain and consensus algorithms.

Sooyong Park

Dr. Sooyong Park is currently Director of Blockchain Research Center at Sogang University sponsored by Korean Government and was a President & CEO of National IT Industry Promotion Agency (NIPA) from September, 2012 until November 14, 2014. Before joining NIPA, he was a computer science professor (March, 1998-September, 2012) and dean of Graduate School of Information and Technology (March, 2011-September, 2012) at Sogang University. He holds Ph.D. in Information Technology from George Mason University, M.S. in Computer Science from the Florida State University, and B.S. at Sogang University.