SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V11I3P104

Volume 11 Issue 3, 31-40, March 2024


*Original Article*

# Split-Join: A Blockchain Framework for Improving Scalability and Performance

Vemula Harish[1], R. Sridevi[2], K.S. Sadasiva Rao[3]

[1]*Department of CSE, Jawaharlal Nehru Technological University, Telangana, India.*
[2]*Department of CSE, JNTUH College of Engineering, Telangana, India.*
[3]*Department of CSE, Sri Indu College of Engineering and Technology, Telangana, India.*

[1]*Corresponding Author : vemula.harish31@gmail.com*

Received: 05 January 2024 Revised: 05 February 2024 Accepted: 04 March 2024 Published: 31 March 2024

***Abstract*** *- Blockchain technology has proven its capability to secure data robustly and reliably, most enterprises are ready to move into blockchain due to its unique characteristics, and many researchers are contributing to improve the technology day by day, but still, there are performance and scalability issues need to be addressed in a better way, so that it can serve the globe by removing intermediaries and bringing the transparency and immutability. In this work, a split-join framework is proposed to improve the scalability and performance of the blockchain technology while maintaining the overall blockchain principles valid and allowing the parallel processing of the blocks efficiently through nonlinear principles, which offer efficient load balancing while processing the transactions. It aims to improve the performance and scalability aspects of blockchain. The proposed framework shows significant improvements in scalability and throughput; the number of parallel blocks processed increases with respect to split-chain length, resulting in higher throughput.*

***Keywords*** *- Blockchain, Scalability, Performance, Consensus, Split-join framework.*

## 1. Introduction

Blockchain technology, introduced by Satoshi Nakamoto [1], was initially limited to only cryptocurrencies. Still, nowadays, most industries are looking forward to bringing their business activities into the blockchain, and also it has become popular in various sectors like healthcare, digital asset management, supply chain management, IoT, and cloud etc.; blockchain solves the double spending problem without centralized architecture, it is very much helpful in maintaining the information in a tamper-proof manner.

The significant potential of the blockchain is that it provides confidentiality, transparency, integrity, and privacy over the peer-to-peer network that uses Decentralized Ledger Technology, known as DLT. Decentralized Applications (DApps) are becoming popular as web3-based applications. NFTs [2], and Metaverse [3] are other emerging areas of blockchain. Blockchain provides data integrity and confidentiality through asymmetric key cryptography algorithms such as SHA-256, Keccak-256, etc.

In general, traditional centralised systems process thousands of transactions per second; thus, they are highly scalable, exhibit minimum transaction latency, have confirmation time, and have high performance. At present, decentralized technology is grabbing the attention because of its unique nature, but it is not highly scalable compared to the traditional centralized systems, to achieve the same level of scalability, it requires refinement in terms of architecture, protocols, and algorithms.

Blockchain is classified into two broad categories: Public blockchains and private blockchains; initially, it was introduced to serve cryptocurrencies as a public blockchain network that added transparency by publishing the transaction in the network and offered immutability, i.e., once the transaction is confirmed, it cannot be modified. Earlier, it was limited to cryptocurrencies only, but later, people understood the potential behind the technology, and as a result, intelligent contracts evolved into blockchain 2.0 [4].

A smart contract is a program that contains the instructions agreed by both parties to conduct their business smoothly, and once these rules are installed onto the network, they are immutable. i.e., it allows business entities to operate in an untrusted environment.

Because the other party cannot modify the program alone, every transaction must be agreed upon by a certain pre-defined number of participants; then, only it validates the transaction and marks it as successful; otherwise, it rejects the transaction and invalidates it.

This agreement happens through consensus algorithms such as proof of work [5], proof of stake [6], etc.; consensus algorithms solve distributed consensus issues and propose a solution for Byzantine Generals problem.

Public blockchains such as Bitcoin [7], Ethereum [8], etc., are exhibiting low scalability due to their open nature and complex consensus algorithms, i.e., public blockchains are open to everybody, so that anybody can join the network, send transactions, and participate in mining the transactions into Blocks. Here, public blockchains allow anybody over the globe to participate; as it is allowing vast numbers of participants, it is challenging to make everyone agree for the same set of transactions to be accumulated into a single block, to achieve this, there should be only one winner to publish the block.

A node or network participant who mines the block is the miner, who calculates the block's hash to meet the target by choosing the appropriate nonce value. As all miners compete to create a block of transactions, sometimes more than one miner may emerge as the winner; then, both blocks are added parallelly, called forks in blockchain.

Forks can be resolved using the most prolonged chain principles. Due to these complexities, it is difficult to achieve consensus within less time and publish the block to the network; hence, bitcoin creates a single block every 10 minutes, showing seven transactions per second; Ethereum also suffers from scalability issues and supports 15 transactions per second [9], compared to public blockchains the Private blockchains are more scalable.

Researchers propose solutions to improve the scalability of blockchain platforms; they include optimizing the core architecture and enhancing the consensus protocols to avoid unnecessary overhead while mining, load balancing between the network participants, and parallelizing the transaction processing by overcoming the potential performance bottlenecks.

## 2. Related Study

Despite blockchain's unique and secure characteristics, the technology faces performance and scalability issues. There are specific improvements proposed by researchers using shard-based blockchains [10]; Qin Wang et al. [11] conducted a systematic study on Directed Acyclic Graph (DAG) based blockchains and discussed the consensus, security, privacy, and performance aspects in detail.

Lang Li [12] et al. proposed a new DAG-based architecture for addressing issues such as the complexity of tracing the order of parallel blocks in DAG. Zicheng Wang, Bo Cui and Wenhan Hou [13] analyzed the Ethereum blockchain, introduced the dynamic load balancing based on sharding and analyzed throughput based on proof of work consensus. Canlin Li [14] et al. studied the load balancing issues in sharding and proposed a new protocol that efficiently handles the incoming transactions and distributes the transactions evenly between the shards.

Zhongteng Cai [15] et al. proposed a cooperation-based protocol for sharding, which allows shards to verify the correctness of transactions recorded in other shards based on voting. Hiroki Watanabe [16] et al. researched retrieving the historical state information from the blockchain ledgers and proposed a solution to read the info efficiently based on DAG.

Where each technique has its pros and cons. To realize the full potential of blockchain, it is essential to process thousands of transactions per second and scale them to allow many participants to submit their transactions parallelly. Hence, parallel and nonlinear approaches can significantly impact blockchain technology's performance and scalability.

### *2.1. Sharding*

Sharding improves blockchain technology's scalability [17] through partitioning or dividing the entire network into multiple small networks called shards. Shards are responsible for handling the transactions independently to process them parallelly.

This mechanism boosts the throughput to some extent. Still, there is a possibility of overloading a shard with plenty of transactions where other shards might be free and don't have any transactions awaiting. One must take care of efficient load balancing between the shards by continuously monitoring the traffic. As shards are small network chunks, they are also vulnerable [15] to 51% attack.

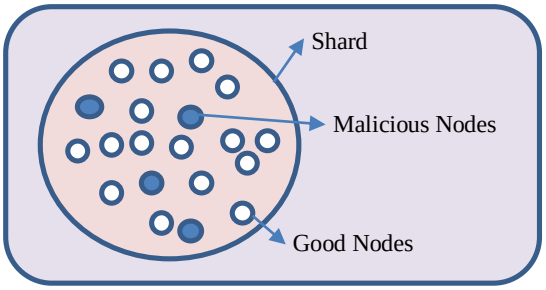


**Fig. 1 Shard**

Figure 1 shows an example of a shard that contains a total of 20 nodes, out of which four nodes are malicious, but in real-time, there will be millions of nodes that participate in the blockchain network; in this example, if we consider the percentage of malicious nodes over total number of nodes are about 20%, this percentage may increase or decrease because of its dynamic topological structure, nodes will join and leave as and when they want, hence it is easy to attack such small network chunks compared to an extensive single network of nodes.

So, a malicious user may gain access to the majority of the nodes to compromise the consensus decisions of the blockchain networks, which play a crucial role in deciding the transaction's approval; due to a lack of governance policies, if a malicious node gains access, then those invalid transactions approved by the majority of malicious nodes are permanently stored in the ledger. Because of its tamper-proof nature, once the blockchain is compromised, the changes made are permanent; nobody has the right to change them later, which may lead to a disastrous situation. The other problem with sharding is the load balancing between shards is difficult.

### *2.2. DAG Based Consensus*

Directed Acyclic Graph (DAG)-based consensus is the scalability [18] solution to the blockchain technology; DAG doesn't use blocks for storing transactions, unlike other blockchain networks. DAG-based blockchains such as IoTA store individual transactions instead of accumulating them into blocks; they are called tips. Tip is a newly arrived transaction that is not confirmed in the blockchain ledger. Every node possesses cumulative weight and weight.

The self-weight is always one (1), and the cumulative weight is obtained by adding all approving nodes. Here, every node that wants to add their transaction to the ledger needs to support the unconfirmed transaction that has already been added. They function as a miner approving the transaction; hence, the transactions in DAG are not final immediately. However, they are committed permanently to the ledger after a certain number of transactions are appended and accepted by newly arriving transactions or Tips.

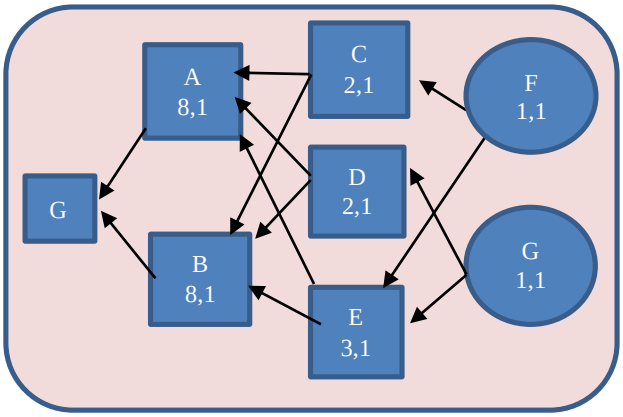


**Fig. 2 DAG-based consensus mechanism**

Figure 2 shows the DAG Consensus algorithm [20], where newly arriving transactions are denoted with the circles (F, G); the DAG consensus mechanism avoids the unnecessary waiting time of a transaction to fill the block, as all transactions are independent of each other one need not wait. However, using DAG, the complexity increases while the blockchain is scaling or when more participants join the blockchain. Because of its complex structure, retrieving the information from the blockchain is difficult.

## 3. Proposed Framework

Blockchain technologies exhibit lower scalability in comparison to conventional centralized systems, to meet the current needs, blockchain must scale in terms of processing the number of transactions, but due to the peer-to-peer nature and decentralized architecture, it isn't easy to achieve high scalability without compromising the original nature of the blockchain technology. However, there are few scalable solutions given by researchers; still, there is a lot of scope for improving the scalability of blockchain technology. Researchers are now actively working on the scalability and performance aspects of the blockchain platforms.

The proposed framework addresses the scalability issues of blockchain platforms by adopting the nonlinear approach; in general, blockchain platforms process one block of transactions at a time; due to distributed consensus and synchronization issues, blockchain is not processing the transactions parallelly, thus, the Directed Acyclic Graph (DAG) based blockchains tried to address this issue by avoiding the grouping of transactions into blocks. Instead, they processed individual transactions parallelly using DAG. However, the complexity of retrieving or searching for transactions increases proportionally with respect to the DAG spread.

This study presents a new framework for blockchain technology that improves the scalability of blockchain systems by using nonlinear characteristics; using this approach, the performance of the underlying blockchain platform also increases, and the proposed framework modifies the way legacy blockchain ledger stores the block of transactions. Instead of linear, this framework introduces a nonlinear fashion to store blocks; all blocks are well connected. It is possible to trace all transactions in the blockchain from the genesis block to the very recent block of transactions; this approach allows parallelism and improves the blockchain technology's scalability.

### *3.1. System Architecture*

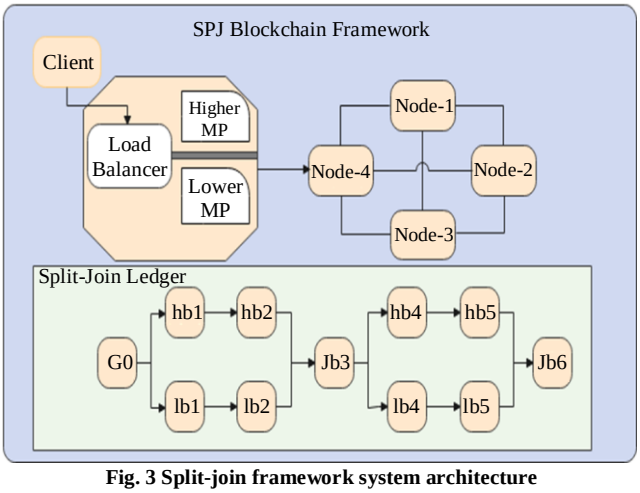


**Fig. 3 Split-join framework system architecture**

The split-join blockchain architecture distinguishes itself from standard blockchain platforms primarily in its simultaneous transaction processing and appending or creating blocks in the distributed ledger; it starts with a genesis block, "G0", also called a join-block. Genesis block is created while setting up the network for the first time. Once the genesis block is created, the blockchain state is transferred to "split-state" Subsequently, two blocks are created parallelly, and the chain splits into two; it is like forks in the bitcoin blockchain, but they again join after a certain number of blocks are added to split-chain.

The difference is the way transactions are accumulated into these blocks. Both the blocks receive transactions from different sources, i.e., separate mining pools, which allow independent processing of blocks; these mining pools are maintained to process the transactions parallelly; they are named Higher-Mining-Pool (HMP) and Lower-Mining-Pool (LMP).

Network participants (miners) competing to create higher blocks differ from those competing to develop lower ones. Initially, the transactions are received into the Primary Mining Pool (PMP), where all unconfirmed transactions are stored, and then there is a possibility of two states in the split-join blockchain they are "join-state" and "split-state". If the current state is join-state, then only one block will be created next, and it is labelled as join-block; for creating a join block, the split-join framework follows the traditional way of mining the transactions into blocks; as a result, the single block of transactions is completed.

Otherwise, if the current state is split-state, the load balancer evenly redirects the transactions into higher and lower mining pools. Load balancer is responsible for sending unique transactions; also, it manages the load between the mining pools to avoid overloading one of the mining pools where the other one is free; load-balancer avoids duplicate transactions in parallel blocks, for processing the unconfirmed transactions there will be two independent sets of miners competing to create higher split-chain block and lower split-chain block.

A split chain is a chain of blocks created parallelly after the previous join block. for example, consider that hb1, and hb2 blocks form a split-chain and their split-chain-length is 2.

*3.1.1. Split-Chain-Length*

It is the number of blocks created in split-chain; this value is configured before starting the genesis block and remains permanent throughout the blockchain lifecycle or until and unless the blockchain is alive.

Miners can pick transactions from either of the mining pools to create a block, computing hash. Here, two blocks are made independently and in parallel [19], and two miners will emerge as winners. In this approach, forks are not allowed and are resolved immediately by a lottery-based mechanism to reduce complexity. Because of the load-balancer component of the split-join framework, there is no chance of accumulating duplicate transactions into mining pools. Newly created blocks are higher blocks (hb1) and lower blocks (lb1), respectively. Here, both blocks consider Genesis as their previous block; there is no sequence for creating these blocks as they are completely independent and parallel.

Once higher and lower blocks are created, the split-join blockchain framework will check for the "split-chain-length" property before transferring the state to "join-state".

Based on the split-chain-length property split-join blockchain, it decides the number of split blocks to be created before merging and creating a join block.

Case 1: if the split-chain-length value is one (1), the blockchain will transfer its state to "join-state" this property indicates to miners whether to compete for split-block or join-block. i.e., Here, for "join-state", miners compete for the creation of a single block called join block, which is appended to a higher block (hb1) as well as a lower block (lb1).

Case 2: if the split-chain-length value is two (2), then the blockchain will wait in "split-state" until the creation of hb2 and lb2 is completed, later the blockchain will transfer its state to "join-state" Figure 4 shows the example of blockchain ledger with split-chain-length = 2.

The blockchain state holds based on the split-chain-length value. This continuous process results in a blockchain ledger, as shown in Figure 4.

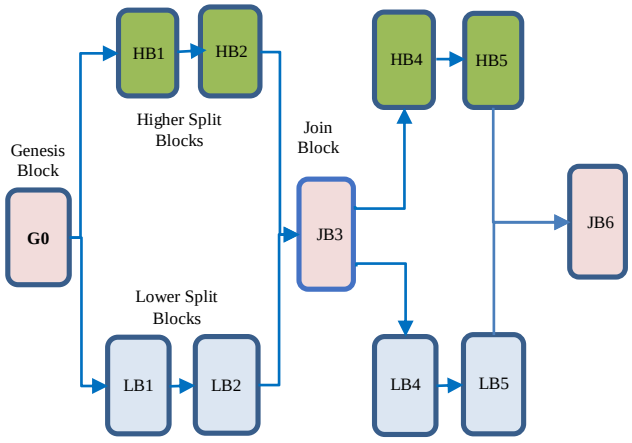


**Fig. 4 Split-join ledger architecture**

***3.2. Lottery-Based Mechanism***

Miners will create a join block once the "split-state" is completed. Suppose two miners solve the block simultaneously and emerge as a leaders. In that case, one of the blocks is accepted based on a lottery mechanism, i.e., if

multiple miners emerge as winners. The algorithm chooses a random number between 1 and 10*N, where N is the number of miners competing to publish the block of transactions. Accept the block created by the miner who got the minimum value. This approach guarantees only one miner will always publish the block.

For example, if three miners are competing to publish the block of transactions, then N=3, let's assume that Miners got a set (S) of random numbers between 1-30, i.e., Miner1=15, Miner-2 = 23 and Miner-3 = 12 then the algorithm finds "m" which is the minimum of S, S = {15, 23, 12}, here m = 12, Miner3 will publish the block of transactions and other blocks are ignored. This approach avoids the forks while creating a split-join block.

### *3.3. Split-Join Block Properties*

All properties of a block in traditional blockchains hold in the split-join block, but some additional properties are included to facilitate the parallel processing of transactions; they have the hash of the previous join block, previous higher block, and last lower block.

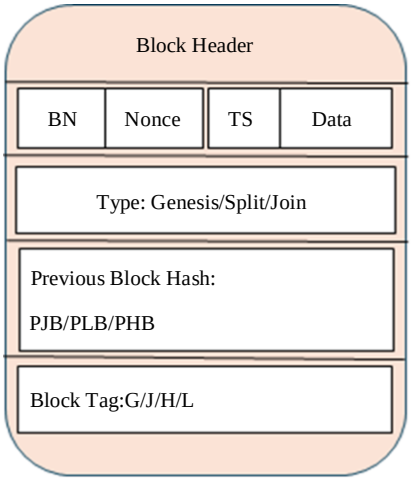


**Fig. 5 Split-join block**

Where,

BN - Block Number,
TS - Time Stamp,
PJB - Previous Join Block,
PLB - Previous Lower Block,
PHB - Previous Higher Block.
G - Genesis,
J - Join,
H - Higher,
L - Lower.

### *3.4. Join Block Properties*

The join block is the particular block in the split-join framework; this block combines the chain splits back into a single chain, responsible for approving the transactions accumulated into previous higher and lower blocks. Join-block verifies any duplicate transactions available in both higher and lower chain blocks created after the previous join block is confirmed.

Additionally, the join block stores the hash of the last higher and lower blocks for maintaining the overall transactions included in split chains; this allows tracking all transactions committed to the blockchain ledger from the latest join block to the genesis block. Merkle tree is used to link all the transactions inside the single block like traditional blockchain ledgers.

The transactions in the split-join framework are not final until the join block is committed to the ledger permanently.

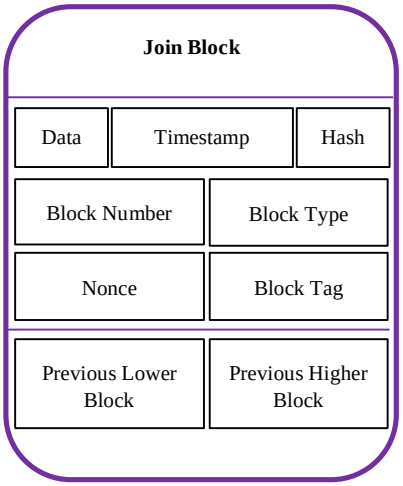


**Fig. 6 Join block**

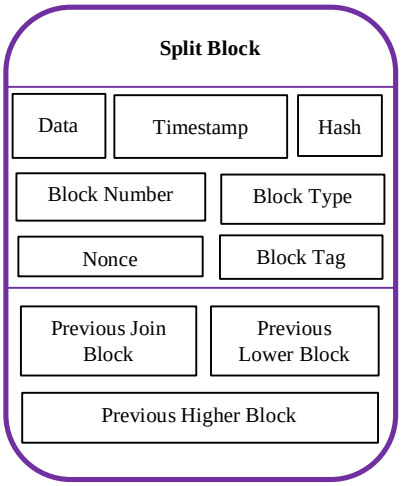


**Fig. 7 Split block**

Figure 6 shows the join-block properties; the genesis block is also considered the join block, and its block type is 'Genesis'. Block type property value will be 'join' for all other join blocks except the genesis block. For further distinguishing the block, the 'Block Tag' property is used; the

block tag value will be 'J' for all join blocks, and for the Genesis block, it is 'G'; the genesis block doesn't store any previous hash values as it is the first block created in the blockchain.

Figure 7 shows the split-block properties; the block type property value will be 'split', and the block tag will be 'H' for higher split-block and 'L' for lower split block. It contains the previous join blocks hash if split-chain-length is 1; for all other values >1 of split-chain-length, it holds the previous split-blocks hash.

To understand this in detail, let us consider the example, see Figure 4 (Split-Join Ledger architecture) for the following example; let us consider a split-block with the block number 'hb1' (Higher Block - 1) then it is in higher split-chain, it is the first block created after the genesis block and hb1 contains the previous hash of only one block, i.e., genesis block (G0) also tagged as 'J' (Join block).

This split block (hb1) contains a hash of previous join block, which holds for all split blocks preceded by the immediate join block. Moving forward to hb2, it is also a divided block with the tag 'H'. hb2 is created because the split-chain-length = 2; the current blockchain state is 'split'. In this case, hb2 has no immediate previous join block, but another split block, hb1, precedes it.

In this case, it cannot store the previous join blocks hash; instead, it stores the previous higher blocks hash as its current blocks hash is 'H'; in the case of lb2, it stores the hash of previous lower blocks hash as its current tag is 'L'. This is how the block tag property determines whether it is in higher or lower split chains.

### *3.5. Adding SPJ Block Algorithm*

Input :

- Latest Block: The current block of split-join blockchain to be confirmed.
- Blockchain Parameters: Parameters and settings of the blockchain.

Output :

- New Block: Update the ledger of Split-Join blockchain by appending the latest block.

Begin

Step 1 :

- Fetch the latest block details from the split-join blockchain.
- Read the details of current blockchain state, block-type, and block-tag.

Step 2 :

- Append the block based on the latest confirmed block type.

Case 1 :

If latest-block-type = "Genesis" then

- Update the block parameters
- Change the block-chain-state to "split-state"
- Add current block tag based on mining pool ('H'/'L')
- Check if from higher-mining-pool then set current block tag ='H'
- Check if "from lower-mining-pool" then current block tag ='L'
- Add previous block hash.
- Append the block to the existing split-join ledger.

Case 2 :

If latest-block-type = "Split" then

- Check the latest block-tag committed to ledger.
- If tag value is ('H' / 'L'), then Update current block-tag based on Mining Pool Type.
- Check If the split-chain-length > 0 then
- Check If previous block-tag is 'G' or 'J'.
- Decrease the split-chain-length of current split-chain ('H'/'L') by 1, Until it becomes zero '0'.
- Check If split-chain-length is equal to '0'.
- Change the current blockchains state to "join-state."
- Append the Block to the spit-join ledger.

Case 3 :

If latest-block-type ="Join" then

- Change the blockchain-state = "split-state"
- Update current blocks tag based on mining pool ('H'/'L')
- If from higher-mining-pool then set current block tag ='H'
- Else If "from lower-mining-pool" then set Current block tag ='L'
- Add the previous block hash from latest blocks hash.
- Append the Block to the split-join ledger.

End

### *3.6. Algorithm Complexity*

When the most recent block is the genesis block or the join block, the difficulty of adding a block to the split-join ledger is O (1). However, when the most recent committed block is a split block, the complexity of the process depends on several parameters, including the length of the split chain, the state of the blockchain, and the information about the blocks that came before it.

### *3.7. Searching a Transaction in SPJ Ledger*

All transactions in the latest join block committed to the genesis block can be searched, and any transaction can be verified throughout the ledger. The transactions are also labelled with either 'l' or 'h' to indicate whether the transaction is included in the higher or lower split chain.

**Table 1. Performance, scalability and finality of blockchain platforms**

| Aspect | Hyperledger Fabric | Bitcoin | Ethereum | IOTA | Split-Join |
|---|---|---|---|---|---|
| **Purpose** | Permissioned, enterprise use | Digital currency | Smart contracts | Internet of Things (IoT) | For enterprise as well as public blockchains |
| **Performance** | High | Low | Moderate | High | High |
| **Scalability** | Scalable with modular design | Limited (due to PoW) | Working on scalability solutions (EIP 1559, ETH 2.0) | Scalable | Scalable with parallel processing |
| **Finality** | Can be configured (e.g., immediate or eventual) | Eventual (PoW) | Eventual (PoS) | Immediate | Eventual, (all transactions are final till the latest join block, which is confirmed) |

It helps improve search performance by eliminating unnecessary comparisons. i.e., if a transaction is included in a higher split-chain, it avoids searching in lower split-chain, reducing the significant number of comparisons.

## 4. Results and Discussion

Traditional blockchain platforms process the block of transactions serially, i.e., the second block is created only after completing the first block. Parallel processing of blocks is not supported due to its decentralized nature; serial processing of blocks of transactions is a performance bottleneck; it impacts the scalability of the underlying blockchain platform. Split-join blockchain allows blockchains to process transactions parallelly and significantly improve the scalability.

Blockchain technologies are designed with different objectives and operational paradigms, catering to various application domains. Hyperledger fabric [21] is a permissioned framework for enterprise applications, while Bitcoin focuses on digital transactions with limited scalability. Ethereum extends to smart contracts, IOTA [22] for IoT, and split-join blockchain for enterprise and public blockchains.

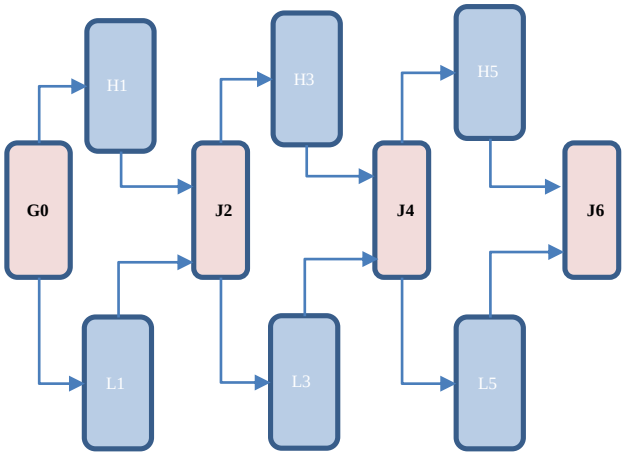


**Fig. 8 Split-join ledger where split-chain-length = 1**

Each platform addresses specific needs, such as transaction efficiency, scalability, and finality; compared to split join blockchain, the performance, scalability, and finality features of blockchain platforms are outlined in Table 1. Figure 8 shows the split-join ledger architecture with a split-chain-length is 1, i.e., the blockchain state will change from split-state to join-state after creating a single block in split-chain, H1, H3 and H5 are part of higher split-chain and L1, L3 and L5 are part of lower split-chain, after creating H1, L1 it is joined at J2, H1 and L1 are parallel blocks that contain same block number with different block labels (l/h). The total number of blocks confirmed till the Join Block (*JBi*) is given as Tb where 'i' is the block number.

$$Tbn = \sum_{i=0}^{n} JBi + \left(\frac{JBi}{2}\right) + 1$$

To better understand the advantage of using a split-join framework, let us consider the example that there are 12 blocks confirmed in traditional blockchain, the block size is 100, where each block can accumulate 100 transactions maximum within a given period 't'; let us assume t=10 minutes. The total number of blocks created is 12, and the time taken to develop these blocks is 120 Minutes, confirming the completion of 1200 transactions. Here, traditional blockchain cannot scale beyond the limitations due to serial processing of transactions and complex consensus mechanisms.

Due to public consensus, we cannot reduce the block creation time to a minimal value in Bitcoin-like blockchains; in such cases, split join blockchain helps us process a greater number of transactions within the same period. The total number of blocks created in the split-join ledger is given by. Block JBi = 12, JBi should be a join-block because transactions are only confirmed till the latest join-block is committed to the split-join ledger; upon substituting the JBi value, we get.

$$Tbn = \sum_{i=0}^{n} 12 + \left(\frac{12}{2}\right) + 1$$

$$Tbn = 19$$

The split-join framework can process 19 blocks within 120 minutes, and the maximum number of transactions it can process is 1900. As the blockchain grows, the number of transactions that can be processed improves significantly.

**Table 2. Number of blocks in traditional vs Split-join with SPL = 1**

| Block Creation Time in Minutes | Number of Blocks Committed in Traditional Blockchain | Number of Blocks Committed in Split-Join Blockchain |
|---|---|---|
| 10 | 2 | 4 |
| 10 | 4 | 7 |
| 10 | 8 | 13 |
| 10 | 12 | 19 |
| 10 | 20 | 31 |
| 10 | 50 | 76 |
| 10 | 100 | 151 |
| 10 | 200 | 301 |
| 10 | 400 | 601 |
| 10 | 500 | 751 |
| 10 | 1000 | 1501 |

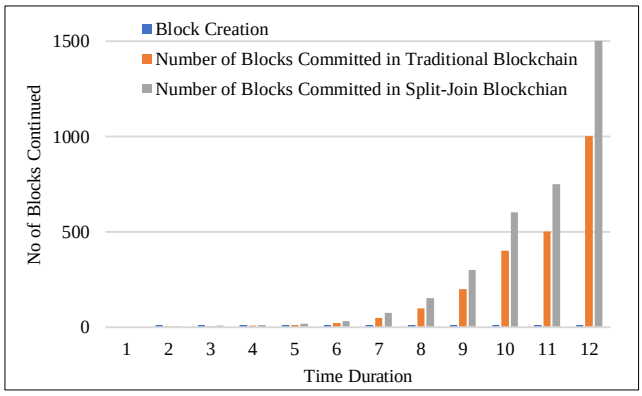


**Fig. 9 Traditional vs Split-join blockchain**

Figure 9 shows that as the blockchain is growing, there is a significant impact on the number of blocks created within the same period compared to traditional blockchain platforms; irrespective of the blockchain performance or efficiency in transaction processing, this framework will allow blockchains to scale big enough to process more transactions.

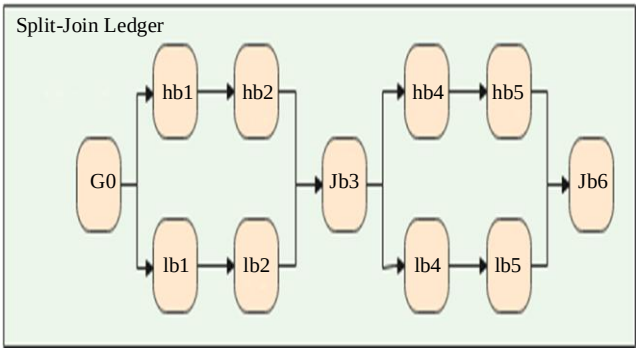


**Fig. 10 Ledger with split-chain-length is 2**

Consider the split-chain-length is 2, and then the total number of blocks is confirmed until the join-block 'JBi' is given as Tbn.

$$Tbn = \sum_{i=0}^{n} IBn + \left(\left(\frac{JBi-3}{3}\right) * 5\right)$$

IBn is given as the Initial number of blocks created until the first join block (J3) is created, i.e., '6' in Figure 10 above, for example, to calculate the number of blocks committed till join block-6 where SPL=2 then Tbn =11.

$$Tbn = \sum_{i=0}^{n} 6 + \left(\left(\frac{6-3}{3}\right) * 5\right)$$

**Table 3. Number of blocks in traditional vs Split-join with SPL = 2**

| Block Creation Time in Minutes | Number of Blocks Committed in Traditional Blockchain | Number of Blocks Committed in Split-Join Blockchain |
|---|---|---|
| 10 | 3 | 6 |
| 10 | 6 | 11 |
| 10 | 9 | 16 |
| 10 | 12 | 21 |
| 10 | 48 | 81 |
| 10 | 99 | 166 |
| 10 | 201 | 336 |
| 10 | 399 | 666 |
| 10 | 501 | 836 |
| 10 | 999 | 1666 |

From the Figure11, it is evident that the total number of blocks committed to the split-join ledger has increased significantly compared to Figure 10; expanding the split chain length allows more parallel blocks to be processed.

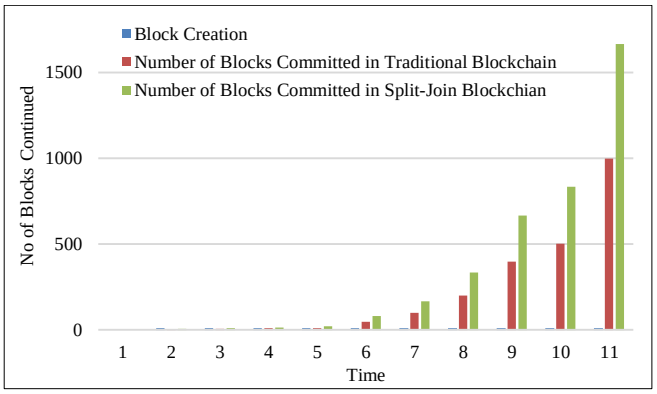


**Fig. 11 Traditional vs Split-join blockchain**

### *4.1. Overhead Analysis of Split-Join Blockchain*

Compared to the traditional blockchains, this framework adds a load balancer to process the transactions in parallel to avoid overloading one of the split chains, whereas the other is free. This also eliminates duplicate transactions in mining pools, it forwards the transactions by verifying that it does not exist in any of the mining pools. However, the transactions of higher and lower split chains are verified later by miners while confirming the subsequent blocks. This adds little overhead to the existing functionality but solves the issues of double spending and avoids intentional, repeated transaction submissions to mislead the underlying system. The proposed system resolves the forks immediately by selecting one of the miners as a winner using a lottery-based mechanism instead of accepting the most extended chain principle, which ignores the transactions included in the small chain. This model will significantly improve the blockchain network's overall scalability and performance at negligible overhead.

### *4.2. Split-Join Blocks*

Using the split-join architecture, the following blocks are created. The Genesis block is designated as G and numbered as '0'. Afterwards, two split blocks are created with the same block number '1', but their 'Tag' property distinguishes them as belonging to a higher or lower split chain. Finally, block 2 is the joining block, labelled J. Join blocks are responsible for approving the transactions of previous split blocks.

BlockNumber: 0
Data: {"data":"}
BlockType: "Genesis"
Previous Hash: ""
Hash: "0"
Tag: "G"

BlockNumber: 1
Data: {"amount":503,"from": "John","to":"Smith"}
BlockType: "Split"
PreviousHash: "0"
Hash:
"00ab6fb6dd6c19a4c19dc978d8b100656cee99c80e7d
bfdd0d4fe868493dfee0"
Tag: "H"

BlockNumber: 1
Data: {"amount":100,"from":"Alice","to":"Bob"}
BlockType: "Split"
PreviousHash: "0"
Hash:
"0007b1eb93cc48095c7b90ce3767de7b233d7172ab1
576174451d254e37ef484"
Tag: "L"

BlockNumber: 2
Data: {"amount":5000,"from":"Alice","to":"John"}
BlockType: "Join"
Hash:
"001ee6e4702f0704c809188f7577b7f964f281e28e9fd
b71e029cd63e2d74f0b"
Tag: "J"
PreviousLowerBlockHash:"0007b1eb93cc48095c7b9
0ce3767de7b233d7172ab1576174451d254e37ef484"
PreviousHigherBlockHash:"00ab6fb6dd6c19a4c19dc
978d8b100656cee99c80e7dbfdd0d4fe868493dfee0"

## 5. Conclusion

Split-join framework provides a new solution to the scalability issues of blockchain. Public and private blockchain platforms can adopt it because it does not change any of the existing features of blockchain technology. It enhances the performance of underlying blockchain platforms significantly.

Furthermore, it proposed a solution to avoid duplicate transactions in forks using load-balancer component and avoid the overloading issue of sharding-based blockchain platforms, reducing the overall complexity by avoiding the unnecessary forks while creating the blocks using a lottery-based mechanism. Thus, the proposed framework allows blockchains to scale without compromising features and boosts performance.

It is recommended to use split-chain-length limited to one or two to avoid unnecessary overhead on join blocks while approving the transactions included in split-chains. In future, this work could be further extended by enhancing the framework features, fine-tuning the architectural components and adopting best practices to boost the scalability and performance of blockchain.

## References


[1] Satoshi Nakamoto, A Peer-to-Peer Electronic Cash System, Bitcoin.org, pp. 1-9, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] N'guessan Patrice Akoguhi, and M. Bhavsingh, "Blockchain Technology in Real Estate: Applications, Challenges, and Future Prospects," *International Journal of Computer Engineering in Research Trends*, vol. 10, no. 9, pp. 16-21, 2023. [CrossRef] [Publisher Link]

[3] Thien Huynh-The et al., "Blockchain for the Metaverse: A Review," *Future Generation Computer Systems*, vol. 143, pp. 401-419, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Zibin Zheng et al., "An Overview on Smart Contracts: Challenges, Advances and Platforms," *Future Generation Computer Systems*, vol. 105, pp. 475-491, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Mohammed Adam Kunna Azrag et al., "A Novel Blockchain-Based Framework for Enhancing Supply Chain Management," *International Journal of Computer Engineering in Research Trends*, vol. 10, no. 6, pp. 22-28, 2023. [CrossRef] [Publisher Link]

[6] Cong T. Nguyen et al., "Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities," *IEEE Access*, vol. 7, pp. 85727-85745, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Ayushi Singh et al., "A Survey of Blockchain Technology Security," *International Journal of Computer Engineering in Research Trends*, vol. 6, no. 4, pp. 299-303, 2019. [Publisher Link]

[8] Gavin Wood, "*Ethereum: A Secure Decentralised Generalised Transaction Ledger*," Ethereum Project Yellow Paper, pp. 1-41, 2014. [Google Scholar] [Publisher Link]

[9] Anshika Bhalla, Top Cryptocurrencies with their High Transaction Speeds, Blockchain Council, 2024. [Online]. Available: https://www.blockchain-council.org/cryptocurrency/top-cryptocurrencies-with-their-high-transaction-speeds/

[10] Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih, "A Novel Methodology-Based Joint Hypergeometric Distribution to Analyze the Security of Sharded Blockchains," *IEEE Access*, vol. 8, pp. 179389-179399, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Qin Wang et al., "SoK: Diving into DAG-Based Blockchain Systems," *arXiv*, pp. 1-38, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[12] Lang Li, Dongyan Huang, and Chengyao Zhang, "An Efficient Dag Blockchain Architecture for IoT," *IEEE Internet of Things Journal*, vol. 10, no. 2, pp. 1286-1296, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Zicheng Wang, Bo Cui, and Wenhan Hou, "A Dynamic Load Balancing Scheme Based on Network Sharding in Private Ethereum Blockchain," *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, Los Alamitos, USA, pp. 362-367, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Canlin Li et al., "Achieving Scalability and Load Balance across Blockchain Shards for State Sharding," *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*, Vienna, Austria, pp. 284-294, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Zhongteng Cai et al., "Benzene: Scaling Blockchain with Cooperation-Based Sharding," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 639-654, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] Hiroki Watanabe et al., "Enhancing Blockchain Traceability with DAG-Based Tokens," *2019 IEEE International Conference on Blockchain (Blockchain)*, Atlanta, USA, pp. 220-227, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[17] D. Bhanu Sravanthi, and P. Venkata Krishna, "Digital Railway Ticketing Using Ethereum and Smart Contracts," *International Journal of Computer Engineering in Research Trends*, vol. 10, no. 4, pp. 167-171, 2023. [CrossRef] [Publisher Link]

[18] Xun Xiao, "Accelerating Tip Selection in Burst Message Arrivals for DAG-Based Blockchain Systems," *IEEE Transactions on Services Computing*, pp. 1-13, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] Jia Kan, Shangzhe Chen, and Xin Huang, "Improve Blockchain Performance Using Graph Data Structure and Parallel Mining," *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, Shenzhen, China, pp. 173-178, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[20] Huma Pervez et al., "A Comparative Analysis of DAG-Based Blockchain Architectures," *2018 12th International Conference on Open Source Systems and Technologies (ICOSST)*, Lahore, Pakistan, pp. 27-34, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[21] Omar Levano-Stella, Jonardo L. Lerios, and Mohamed Remaida, "A Blockchain-Based Approach for Securing IoT Devices in Smart Homes," *International Journal of Computer Engineering in Research Trends*, vol. 10, no. 10, pp. 8-15, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[22] Wellington Fernandes Silvano, and Roderval Marcelino, "Iota Tangle: A Cryptocurrency to Communicate Internet-of-Things Data," *Future Generation Computer Systems*, vol. 112, pp. 307-319, 2020. [CrossRef] [Google Scholar] [Publisher Link]