Scalability of blockchain: review of cross-sharding with high communication overhead.

Firas Hammoodi Neamah Al-mutar¹*, Ahmed Ali Talib Al-khazaali ² and Baqar assam hataf²

¹Iraqi Drilling company, Basra, Iraq.
²University of Alkafeel of Alkafeel, Najaf, Iraq.

Abstract. Sharding method is separates the network into smaller groups to reduce latency and enhance blockchain speed. To reduce storage cost, divide the network into separate segments, and allow nodes to maintain track of a portion of the blockchain's data ledger, it was initially employed in databases. This technology is an excellent choice for enhancing blockchain performance because of its practical requirements and the speed at which blockchain applications are developing. It has garnered a lot of interest. There are a number of unresolved issues regarding the review and analysis of sharding. In this paper, we examine current state-of-the-art sharding schemes by categorizing them according to blockchain type and sharding technique—more specifically, cross-sharding with low communication overhead and systematically and thoroughly analyzing the benefits and drawbacks of each. Sharding lowers communication overhead since the performance of blockchain apps that use it has significantly improved over the method that should be studied for reducing the communication cost of block consensus. We present various open addresses after doing a comprehensive review and analysis of the communication overhead.

1 Introduction

Blockchain is a distributed, unchangeable digital ledger that has attracted a lot of attention from academics and businesses because of its cutting-edge features, including traceability, transparency, and decentralization[1]. It presents fresh security issues and solutions for decentralized applications in industries like supply chain management, digital banking, and energy. However, it is needed to increase the efficiency of current consensus processes to help keep up with market progress. The mainstream Bitcoin network is inefficient for everyday transactions since it can only process seven transactions per second (TPS) at its theoretical maximum. Blockchain technology don't perform as well as non-blockchain systems in terms of latency and throughput[2]. Specifically, the transaction throughput (TPS) of the Ethereum and Bitcoin blockchains are 3–4 and 15 TPS, respectively. PayPal and Visa, on the other hand, get 193 and 1667 TPS, respectively. Compared to non-blockchain servers like Google and YouTube, blockchain servers not only have a lot of storage data but also read data slowly. This highlights a significant issue with blockchain scalability. Thus, improving throughput and blockchain scalability are urgent issues for research and development[3].

The blockchain has a trilemma of achieving scalability, decentralization, and security. Approaches to solving scalability can be categorized into four main categories based on the layer in the blockchain stack. Layer 0 methods involve a multi-chain ecosystem, allowing interoperability between chains and enabling developers to create custom chains[5]. Layer 1 methods enhance the blockchain and network prototype to improve scalability, representing the blockchain and network. Layer 2 methods use mechanisms outside the main blockchain to improve scalability. Layer 3 methods, also known as multilayer scaling or hyper-scaling, compress data multiple times by adding another layer on top. wherever Layer 1 (on-chain) includes scaling solutions, which target core blockchain elements such as blocks, transactions, and other on-chain data structures to address the scalability problem[6].

*Corresponding author: Firas.almutar@idc.gov.iq
Researchers proposed solutions such as transaction parallelization, increase in block size, block interval reduction, invertible bloom lookup tables, sharding, private blockchains, block propagation, faster consensus mechanisms, and DAG-based chains[7].

Sharding is a method that divides the network into shards to improve blockchain performance and minimize latency. It was first used in databases to lower storage overhead, partition the network into distinct parts, and enable nodes to keep track of a section of the blockchain's data ledger[8]. Because of its practical needs and the quick development of blockchain applications, this technology has drawn significant amounts of attention and is a good option for improving blockchain performance[9]. There are several open issues based on review and analysis to improve sharding, in this paper, We examine current state-of-the-art sharding schemes by categorizing them according to the kind of blockchain and sharding method (i.e., cross-sharding with minimal overhead) and conducting a consistent and comprehensive analysis of their advantages and disadvantages. Additionally, as cross-shard transactions using the present consensus techniques are always beset by high communication overhead, the review focused on studies of low-communication-overhead sharding solutions. Since the efficiency of blockchain applications using sharding has greatly improved over the strategy that should be investigated to minimize the communication cost of block consensus, sharding reduces communication overhead. Meanwhile, state-of-the-art techniques for encrypting and compressing data might be investigated to minimize the volume of information that has to be sent between and among shards. Furthermore, drawing on the aforementioned thorough study, we identify a number of outstanding problems and suggest future research areas. In specific, the following is a summary of this paper's major contributions.

Based on the type of blockchain and the sharding approach, we categorize the current sharding schemes of blockchain.

We carry out a thorough analysis of current sharding systems that deal with cross-sharding with high communication overhead.

By carefully deliberating over and analyzing several unresolved problems, we also develop a list of future study directions.

This paper organizes as follows, a brief introduction to the basics of blockchain sharding is given in Section 2. The categorization of current research method sharding schemes according to blockchain type and sharding technique, which addresses cross-sharding with low communication overhead is illustrated in Section 3. We utilize evaluation criteria to examine the benefits and drawbacks of each. In Section 4, we discuss open addresses and potential research directions based on the literature evaluation. In the last section, we conclude this paper.

2 Background

2.1 The structure of blockchain

Blockchain is a distributed and secure ledger consisting of connected blocks of data and updated by consensus technologies. The identical replica is stored on every node in the blockchain network, which is chained to the hash of its most recent block[10]. Because every alteration in a block modifies the block's hash, which is different from the hash that was previously saved in the next block, tampering is prevented. Because it would involve changing blocks on many network nodes, it is therefore impossible to tamper with blockchain data illegally[11].

The blocks that make up the blockchain are composed of transaction data and a block header. The block header contains the hash, timespan, nonce, Merkle root of transactions, and other network elements from the previous block. The transaction data contains the block transactions indicated by the Merkle root[12]. The Merkle root is the base of the tree of hashes that is created by repeatedly hashing each pair of block transactions until only one hash value is left. This guards against tampering with the transaction data. The basic structure of the blockchain is illustrated in Fig. 2[13].

The consensus mechanism is used by blockchain to protect network reliability and generate new blocks. Proof of Work (PoW) is the basis for cryptocurrencies such Ethereum and Bitcoin are the most widely utilized consensus algorithms[14]. Miners compete in Proof of Work (PoW) by constantly calculating the hash of the block until a desired outcome is reached. The winner is the first node to achieve the desired outcome. The majority of authorized blockchains employ voting based PBFT and Raft consensus. Proof of Elapsed Time (PoET), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), Proof of Authority, Proof of Burn (PoB), Proof of Capacity (PoC), Tendermint, and Ripple are some more consensus techniques.

There are three different types of blockchains: public, private, and consortium. Consensus-based public blockchains enable any individual to join and participate in it without requiring previous authorization[15]. The most common kinds of cryptocurrencies are Ethereum, Bitcoin, and most others. For users to access private blockchains, they must first have authorization. There is just one company that owns these centralized networks, and every node is recognized and permitted. Blockchains used by consortiums are recognized and need
membership; thus, are permitted. These partially centralized networks are the result of a consortium—a collection of organizations—that wish to exchange information but lack confidence[16]. Some instances are Corda and Hyperledger Fabric.

2.2 the scalability of Blockchain and sharding

The adoption of blockchain technology has been severely limited by scalability issues since networks find it difficult to remain effective when the number of transactions rises due to challenges with latency, throughput, and communication overhead[17]. One method that can increase throughput while lowering latency and communication overhead is sharding. Sharding separates the network into many shards, each with a subset of nodes. It is a technique that was first used in traditional centralized databases. By handling disparate transactions in parallel, shards increase the blockchain's processing efficiency[18].

The four stages of sharding execution are reconfiguration, consensus determination, setup and setting, and data initialization. During the data configuration stage, nodes generate identities like public keys and IP addresses. In the shard construction and setup phase, nodes and transactions are randomly separated into distinct shards. Nodes link and exchange identities inside a shard[19]. While cross-shard consensus is carried out amongst many shards to get global status, intra-shard consensus takes place within the same shard. A sharding system allocates nodes into various shards periodically through reconfiguration to preserve system security. Asynchronous, semi-synchronous, and synchronous networks can all have different network states. Partial synchronous networks attain stable states, asynchronous networks miss data about other nodes, and synchronous networks match node behavior and state[20]. While synchronous network-based sharding algorithms are simple, real-world situations sometimes require intricate partially synchronous, or asynchronous networks. Network sharding, transaction sharding, and state sharding are the three tiers of sharding approaches. Network sharding randomly divides each node in a blockchain network to various shards. Both functional and non-functional allocation are typically used in node allocation techniques[21]. Non-functional allocation uses randomization generators, for example RandHound and verified random function (VRF) to allocate nodes to shards based on their identities. Functional allocation, on the other hand, uses machine learning techniques to offer a multifunctional strategy that improves the sharding system's security and self-adaptability[21]. In surveys, the sharding technique is considered one to solve for scalability of blockchain, however, it confronted many challenges of which:

1. First: lack of sharding automatically: The efficiency and security of autonomous sharding are not sufficiently supported by the present study findings. Smart contracts were created to automate the execution of agreements without the need for a third party or to spend time, making it possible to accurately execute complicated digital promises[22]. Nevertheless, current systems do not automatically split nodes or transactions with efficiency, stability, and security throughout the sharding construction process; instead, they only execute transactions.

2. Second: limited universality: Because current optimization algorithms prioritize local
optimization over global optimization, it is challenging to create sharding that is clever enough to accommodate many circumstances and guarantee optimal performance[25]. Thus, it is still an intriguing and open problem to make sharding clever enough to match varied contexts. Fourth, Lack of guarantee of privacy: privacy protection in sharding is frequently disregarded because usernames and transaction information are publicly available on the blockchain. The sharding solutions that are now in place frequently depend on time-consuming cryptographic techniques or demand support from TEE, making them unsuitable for widespread use. As a result, in-depth research on privacy-preserving sharding in the literature is required[26].

Fifth, Deficient security: Because nodes are randomly divided without considering their heterogeneity, network sharding raises security issues and causes noticeable differences in the average hash of the blockchain system[27]. Because of cross-shard validation, transaction sharding adds extra communication cost that results in conflicts and inconsistencies. Preventing nodes from clashing is essential if one hopes to influence the blockchain. State sharding raises the expense of reserve storage to preserve the blockchain's global state. Because a centralized backup system has a single point of failure, it raises additional security risks because attackers may target shards for other attacks, such as double-spend attacks[28]. Lastly, High communication overhead: In contrast to conventional systems like Bitcoin and Ethereum, sharding in blockchains has increased throughput; yet, because it requires additional computing and communication expenses, it has a significant communication overhead. Cross-shard communication is required to retrieve data from other shards or maintain a uniform global state, which can drastically lower system throughput[29]. It's a work in progress to minimize communication overhead while maintaining atomicity. Significant resources, such as computer power, network bandwidth, and storage capacity, are in short supply in blockchain systems and are also used in intra-shard consensus processes. Sharding synchronization is frequently disregarded, particularly in asynchronous networks. Because nodes in an asynchronous network have varying perspectives on the global state, which affects efficiency consensus and makes achieving atomicity challenging, designing an efficient sharding system is a tough task. This paper reviews research that addresses approaches to reduce the high communication overhead connected with the sharding technique[30].

3 review of schemes

This section provides an in-depth review of various cutting-edge sharding methods, utilizing a particular focus on blockchain scalability and approaches for reducing the high communication overhead related to the sharding methodology. We analyze and evaluate each scheme. Furthermore, we provide a summary and comparison of all reviewed sharding approaches in Table 1. DSSBD provided a deep reinforcement learning-based dynamic state-sharding blockchain architecture (DSSBD) that maximizes a blockchain-based crowdsourcing system's block size, block interval, and shard count[31]. This method successfully thwarts several assaults, including replay, double spend, Sybil, and transaction atomicity, while greatly enhancing the blockchain's scalability. The proposed DSSBD offers improved throughput, latency, balancing, cross-shard transaction percentage, and node reconfiguration proportion all while preserving security. Future work will focus on integrating POS consensus procedures, dynamically selecting high-trust validators in S-Shard, and combining on-chain and off-chain scaling technologies to make the blockchain more scalable. By using state-sharding technology, DSSBD improved the blockchain's scalability and decentralization in a crowdsourcing system. The architecture ensures security and atomicity of cross-shard transactions using reconfiguration, sharding, and security modules. It also uses deep reinforcement learning to optimize the system's performance by finding the optimal trade-off between decentralization, security, and scalability. DSSBD is the first architecture to apply deep reinforcement learning in state sharding blockchain to enhance sharding and block generation, ensuring the security of the blockchain and the effectiveness of transactions. In the reconfiguration stage, transaction types are rapidly distinguished using a redesigned state tree, which lowers the query time complexity to O(1). In DSSBD, an epoch is a predetermined clock period that involves consensus-building and node reconfiguration. There are two types of nodes on a blockchain: non-miner nodes and data miner nodes. Some nodes must be incentivized to maintain the blockchain as non-miner nodes to guarantee transaction atomicity and security during the consensus phase and preserve stability throughout the reconfiguration phase.

Epochs in DSSBD establish clock intervals during which node reconfiguration and consensus processes take place. Nodes in a blockchain can be classified as miner or non-miner. It is advisable to support some nodes—which may be further separated into relay and shard nodes—maintaining the blockchain as non-miner nodes in order to ensure transaction stability and updating during consensus and reconfiguration. To enable cross-shard transactions, each relay node (R-Node) has a relay account that has to be divided into smaller accounts and donated to different shards. Smart contracts are used to pledge equity in order to establish R-Nodes. Similar to a committee, shard nodes are formed by voting and application, creating the S-Shard, a new shard. They manage the reorganization and reconfiguration of each shard's node based on security and transaction load at each epoch.
enable cross-shard transactions, each relay node has a relay account that has to be divided up into smaller accounts and donated to different shards. R-Nodes are created by pledging equity using smart contracts. Shard nodes are formed by application and election, much like a committee, creating a new shard called S-Shard. They are responsible for rearranging each shard's node based on security and transaction load at each epoch. In Fig. 2, the DSSBD model is shown. A node cannot perform centralized processing on individual relay accounts in order to become an R-Node for more shards; instead, it must spend extra resources for the storage of multi-shard state information and equity for mortgages. A portion of the transaction fee is sent to the relay account to support R-Nodes, keep the system running, and promote decentralization. This ensures effective cross-shard transaction processing.

![Fig. 2. displays the dynamic state-sharding blockchain architecture (DSSBD) model](image)

The QPM focuses on improving mining and transaction speed by quintessential parallel multiprocessing (QPM)[32]. In order to reduce transaction execution time, this entails dividing an instructed work into smaller jobs that operate in parallel. Programs and blockchains may also be optimized via the use of distributed and parallel computing techniques. The usage of distributed machine learning in intelligent applications is growing. A distributed strategy is used to allocate working nodes to the dataset, and a parallelized scheme is used for model training in a suggested framework for improving the L-BFGS algorithm. Because the distributed blockchain incentivizes nodes to contribute more to machine learning activities, algorithm performance is improved and becomes more accurate. As shown in Figs. 3.a and 3.b, the blockchain’s implementation of crucial parallel multiprocessing solves the scalability problem of transaction and mining performance. Up to six processes may operate concurrently to finish the mining and transaction duties. As seen in Fig. 1, the proof of work occurs with the addition of a completely mined block to the chain. QPM records the mining and transaction process speed of over 20 tries for each process number throughout our testing. A single CPU is used to calculate the mean and standard deviation in order to compare the results with a blockchain. The usage of this strategy in intelligent applications is growing. It has been demonstrated to have a throughput of 125 ktps, a latency of 8 ms, and a cost of 0.2–0.3 USD/hr.
The EC-VRF (Elliptic Curve-Based Verifiable Random Function) is a new Tyche consensus mechanism that chooses accounting and sharding verification nodes using an elliptic curve-based verifiable random function\[33\]. This guarantees security, efficiency, and unpredictable node selection. The sharding chain enhances transaction processing throughput by implementing multi-period, quick, and dynamic sharding accounting. By applying batch verification algorithms to the blocks of shards and producing verification blocks, the verification chain carries out secondary verification. Global consistency of transaction data is ensured by each shard storing the sharding state. In accordance with the FIPS186-4 digital signature standard, it uses Curve P-521 to assess the effectiveness of the EC-VRF and batch verification methods. The findings demonstrate that, even with a high number of nodes, the EC-VRF may be generated and verified in an effective and practical manner. The aggregation signature algorithm also shows a rapid generation of block header signatures, with an average running time of 1.97ms for each accounting node. However, the batch verification running time increases with the number of block heads, with a verification time of only 198.06ms when collecting 50 blocks by sharding accounting nodes. The batch verification algorithm is therefore efficient and feasible, demonstrating the effectiveness of the proposed cryptographic algorithm. Simulation results show EC-VRF's throughput reached 6540tx/s and an estimated latency of 8.59s.

LB-Chain (Load-Balanced and Low-Latency Blockchain) The two main issues facing the LB-Chain architecture are figuring out which and how many accounts should be migrated and developing an efficient and secure migration strategy in blockchain sharding to balance the load on various shards\[34\]. During migration, account migration is required to guard against hostile nodes assaulting the system and execution errors. To maintain security without suffering undue performance loss, an efficient and secure account migration procedure is required. Furthermore, as allocating every account in a large-scale system is impractical, the account allocation algorithm determines which accounts, and to what extent, should be transferred. Consequently, a workable account allocation mechanism is needed to distribute the loads across the shards while allocating a minimal number of accounts.

An analysis of the performance loss caused by uneven transaction load in blockchain sharding is presented in LB-Chain. It suggests a safe and effective migration plan for LB-Chain accounts and transactions that keeps high implementation efficiency while being safe in the case of blockchain sharding. Additionally, the authors provide a workable account allocation technique that moves just a small number of hot accounts to enhance transaction load balance. The system implementation entails creating an LB-Chain prototype and carrying out several tests. Results demonstrate that LB-Chain delivers near-optimal throughput in comparison to optimum load balance methods and successfully balances load among shards, lowering user-perceived transaction confirmation latency. An approach that lowers migration overhead and distributes transaction load over several shards. The account
allocation method and transaction prediction comprise its two components. The allocation service uses a 2-layer Long Short-Term Memory (LSTM) model to forecast how many transactions will be generated by various shards and accounts in each epoch. The loss function is set to MSE, and the model has 100 neurons with a dropout of 0.001. The prediction interval is expanded to estimate the number of transactions for each epoch for the following N epochs at a single prediction, resulting in a feasible method for allocating accounts. For example of learning outcomes of 200 epochs transaction prediction, which is sufficient for the subsequent account allocation. It is not possible to make predictions for every account since a large-scale system might have millions of accounts. Since most accounts only transmit a small number of transactions, there is insufficient data to enable learning for precise forecasts. The writers anticipate the transactions that each shard’s hot accounts will produce throughout each epoch by concentrating on them. To allocate and migrate, the allocation algorithm also concentrates on hot accounts, resulting in comparable performance with significantly less computing complexity. The account allocation algorithm is a technique that is used to show the NP-hardness and transaction prediction findings of the approach by regularly allocating places for accounts and the produced transactions in each short epoch.

**Algorithm 1:** Account Allocation Algorithm for Epoch $t$.

1: **INPUT:** $S$, $A_{hot}$, $n_j^t$, $q_j^{t-1}$, $l_j^t$, $x_{i,j}^{t-1}$, $m_i^t$, $\bar{l}^t$ for all $i, j$

2: $x_{i,j}^t \leftarrow x_{i,j}^{t-1}$,

3: $V_t = \tilde{V}_t = \frac{\sum_{i \in S} \left[ (\sum_{j \in A_{hot}} (l_j^t \cdot x_{i,j}^t) + m_i^t - \bar{l}^t)^2 \right]}{|S|}$

4: Sort each shard $i \in S$ by its load ($\sum_{j \in A_{hot}} (l_j^t \cdot x_{i,j}^t) + m_i^t$) in descending order, save to a sorted shard list $S_{heavy}$, find the most heavily-loaded shard $i_{heavy}$ and the most light-loaded shard $i_{light}$

5: **while** $\sum_{j \in A_{hot}} (l_j^t \cdot x_{i_{heavy},j}^t) + m_i^{t_{heavy}} > \bar{l}^t$ **do**

6: **for** $j$ in $A_{heavy}$ **do**

7: Move $j$ from $i_{heavy}$ to $i_{light}$, update $\tilde{V}_t$

8: if $\tilde{V}_t < V_t$ then

9: $x_{i_{light},j}^t \leftarrow 1$, $x_{i_{heavy},j}^t \leftarrow 0$, $V_t \leftarrow \tilde{V}_t$

10: Update the load on each shard, update $S_{heavy}$, $i_{heavy}$, $i_{light}$ and $A_{heavy}$

11: **go to** line 5

12: **end if**

13: **end for**

14: Remove $i_{heavy}$ from $S_{heavy}$, update $i_{heavy}$, $i_{light}$ and $A_{heavy}$

15: **end while**

16: **OUTPUT:** $x_{i,j}^t$ for all $i, j$

The suggested account allocation method aims to balance the load by moving as few accounts as is practical. Iteratively moving hot accounts from highly loaded shards to light-laden shards in each epoch improves load balancing since it addresses the inefficient and time-consuming issue in real systems. By design, the method is iterative and efficient. An example of the account allocation process is provided by Algorithm 1. Based on the prediction findings in epoch $t$ and the previous epoch $t-1$, loads and load variance are initialized. The hottest account is chosen from the most heavily loaded shard in each iteration, and its transactions are transferred to the shard with the least amount of load. Transactions involving queues are also moved. One computes the new variance $\tilde{V}$. The migration result is kept if the transaction load balance improves, and the subsequent iteration is entered once the parameters have been updated. If not, the account is not moved, and less popular accounts are
moved by the algorithm. If the algorithm is unable to boost $V_t$, it moves all hot accounts in heavy shards to less heavy shards. When all shards with loads over average do not see an improvement in load balancing. Assuming the quantity of malevolent nodes does not surpass the upper bound that the consensus mechanism can tolerate, LB-Chain offers protection during the account migration stage. This ensures that transactions involving account migration may be handled securely and that trustworthy nodes accurately route them to the relevant shard.

Every node in the network modifies its local routing information upon receiving the results of account allocation, ensuring that transactions are appropriately routed to new related shards. The consensus method safeguards the security of the account migration transaction, allowing only legitimate transactions to pass the consensus. To guarantee the security of these transactions, more investigation is necessary as account migration operations are cross-shard transactions. QuarkChain and LB-Chain both use a cross-shard transaction processing mechanism that splits a transaction into two parts: a fund deposit and a fund withdrawal. The beacon shard confirms the withdrawal portion, while the source shard carries out the sender's withdrawal. The destination shard receives the deposit and uses the consensus process to carry out the second half. This guarantees the atomicity (eventually atomicity) of cross-shard transactions, hence guaranteeing their security.

The Ethereum transaction distribution, which has been employed in earlier research, is one of the several transaction distributions that the LB-Chain protocol may operate under. Real Ethereum transaction data is used to assess the protocol's performance, and the results show notable performance increases. The protocol may be readily extended to other systems that have comparable transaction distributions, including consensus systems of the BFT type. It can also be adjusted to accommodate scenarios in which account behaviour and distribution diverge from those of Ethereum transactions. Because security and distribution concerns are less significant in consortium blockchains, the protocol, which is based on PoW consensus, may be extended to both private and consortium blockchains. The primary objective of enhancing the performance of the blockchain system is well-suited for the comparatively high-performance needs seen in consortium blockchains. The account approach, which links every transaction to a particular account, serves as the foundation for the mechanism design. Because of this, it is challenging to extend to the UTXO model; nonetheless, the account model offers a wider range of applications, including edge computing, digital healthcare, and the Internet of Things. Subsequent research endeavors will tackle the load-balancing issue in intricate smart contract operations.

Also, it divides the feasibility analysis of our protocol into two components: the allocation service and the blockchain sharding network. Because the protocol is built on the well-established, mature blockchain sharding technology (QuarkChain), it is quite practical in the blockchain sharding network segment. During the shard reconfiguration stage, the Cuckoo rule and distributed randomness generation method may be used to address the fundamental scheme, account migration. For security concerns, account migrations will not be carried out by the system during the shard reconfiguration period. Because the allocation service is a third-party organization that may be trusted or untrusted, centralized or decentralized, it is also quite viable. However, the allocation service's intricacy discourages it from taking on too many responsibilities, including having to safeguard the system. Excessive centralization can result in a less secure, centralized system that departs from the decentralized character of the blockchain. To address this problem, we minimized computational overhead in our design. The number of accounts that require prediction is small, and the gap between predictions is lengthy—one day, for example. By predicting the number of transactions over different epochs in a single forecast, the allocation service can lower computing costs and provide higher feasibility. In discussing the performance erosion of current blockchain sharding technologies, the essay makes its main point that an imbalance in transaction load is to blame. The framework called LB-Chain, which the authors suggest, ensures transaction load balance by scaling out sharding systems through account and transaction migrations. When deployed on QuarkChain, LB-Chain performs better than random transaction placement schemes in terms of load balancing, latency, throughput, and account fairness. According to experiments, LB-Chain enhances fairness by more than 60%, boosts throughput by more than 10%, and decreases confirmation latency by up to 90%.

Sharding Differential Evolution (SDE) presents blockchain sharding architecture that reduces communication overhead and increases shard trust[35]. To increase shard dependability and lower the chance of blockchain failure, it takes into account differences in shard trust, communication latency, and node count. The shard is assigned to adjacent nodes to minimize communication latency. The approach optimizes node allocation for high throughput and low latency by evaluating nodes beforehand using Differential Evolution (DE). A directory service (DS) committee and shard committees are two components of a blockchain sharding system. The former has a leader node for each committee and gathers micro-blocks from each shard. The DS Committee receives the micro-blocks that are produced when the leader node gathers consensus messages from all of the nodes in its shard. A new block is created after every micro-block has been gathered. All micro-blocks are consolidated into a single block for a blockchain sharding scheme by a special DS Committee. For every shard, a leader node is chosen, and it bundles transactions inside its shard to create a micro-block over a predetermined time period. In a shard, malevolent nodes are identified and labeled as such, but regular nodes are just that—normal. The SDE is designed...
to distinguish between normal nodes within a shard and malicious nodes, based on their detection and classification to as such:

a) Gene: A node assigned a number is referred to as a gene.

b) Individual: A sharding result that is assigned to n nodes is referred to as an individual.

d) Population: The population size is N

d) Individual from the mutation operation: This mutation process is differential.

e) Individual from crossover operation: Following a differential crossover operation, its shard, the shard to shards, is obtained.

f) Individuals in the following generation: They are obtained from a selection procedure.

A crucial part of a blockchain network is the Directory Service (DS) Committee, which is created via Proof of Work (PoW) consensus. It is built by the initial h nodes, who are in responsible of sharding the whole blockchain network. Sharding results are attained by many DS Committee nodes executing SDE. As a stochastic optimization method, DE produces non-unique sharding results. To ensure consistency and reliability, the PBFT consensus is used. A DS Committee member is chosen as a master node to carry out SDE, and via PBFT consensus, agreement is established with other nodes in the DS Committee over allocation outcomes, producing distinct sharding results for the whole blockchain network. By optimizing blockchain node allocation to appropriate shards, SDE seeks to improve security and decrease communication latency. An iterative approach to improve node allocation is shown for a sharding method based on differential evolution. Comparative experiments demonstrate that this methodology lowers the likelihood of a blockchain failure while increasing throughput and decreasing system latency. The suggested paradigm decreases system latency, boosts throughput, and minimizes communication latency between nodes.

SecuSca is a distributed ledger solution that minimizes replication of each blockchain block to lower storage demand. New blocks are added to the longest chain by linking them to the previous block via a chaining method. Blocks are kept on more nodes as more and more enter the system, and as each node's blockchain grows longer, the replication of blocks protected by a combination reduces slowly. As a result, less memory is needed for each node separately and more transaction storage capacity is made possible. Users don't need to save a lot of data when transactions rise. The parameter that nodes use to interact with each other and create a P2P network architecture is called the network model. Asynchronous, semi-synchronous, and synchronous networks are all possible. Semi-synchronous networks use a random variable to define message delivery time, whereas synchronous networks have a fixed upper constraint on the maximum network latency. In partially synchronous networks, members feel confident that all messages will reach their intended receivers, even though the maximum delay is uncertain. Network modelling is made easier and better adaptation to actual network dynamics is provided by partial synchrony. Since there is no maximum network latency on asynchronous networks, transmissions can take as long as they want to reach their intended recipient. This makes asynchronous networks the hardest to establish consensus on. Adversaries' ability to delay communications is limited by the network model, and network dynamics must be considered when designing and implementing a network. The SecuSca approach is a blockchain technology that combines user-triggered transactions into blocks, which are then added and replicated across the network. This approach aims to maintain maximum security and scalability by reducing full replication. It consists of two steps: efficient reduction, which lessens the repetition of certain blocks throughout the network, and efficient replication, which distributes the blockchain's global state to particular nodes while maintaining security. The blockchain is scalable as a result of this procedure, and the network can accommodate additional storage. In order to facilitate the sharding process and provide effective storage and security, the optimization function is added. This allows nodes with inadequate memory storage to engage in the protocol process.

Some nodes are unable to save transaction history using the SecuSca technique because of scalability issues. Rather, the blockchain is dispersed over a larger number of nodes, denoted as nj, which store blocks. These blocks are kept on nodes, n <= N, and are distributed across the network. Even if a node doesn't have any state, it may nevertheless take part in block and transaction validation procedures. Only half of the network's nodes may be tolerated by the system, and if honest nodes have more processing power than malicious nodes, they can withstand malevolent nodes. The replication of blocks at the beginning of the process must be large enough to discourage malicious nodes from attacking the network. For security purposes, there should be more nodes contributing blocks, however this number shouldn't be spread over the network. The highest replication occurs at small sizes and decreases as the size rises. The replication of blocks is correlated with the size of the blockchain in each node. To maintain security, the blockchain makes use of cryptographic blocks. The replication of earlier blocks declines as the blockchain becomes bigger, while the number of most recent blocks stays constant. The method chooses a replication of every previously verified block on every node for every new block, bi, with high replication. Transactions are removed from memory on each node until the minimal replica is reached. Distributed among nodes, SecuSca maintains a minimal copy of all blocks, with each node holding a portion of the blockchain's header and state blocks, including Merkle roots. By lowering the number of explicit transactions in each block and maintaining the blockchain's header for every block, SecuSca lessens block replication. When a new block is introduced, transactions are clearly reflected in it. After that, transactions are only verified when a certain size is
reached in subsequent blocks. In order to avoid scalability problems, this step makes sure that transactions are proportionate to the network demand. Transactions are erased from the blockchain when they are verified and buried. The sharding process in the blockchain involves inserting a new block and replicating it over the network, following an optimization function (R) to balance security and scalability, ensuring a balance between replication and network stability.

\[
R(d) = \begin{cases} 
\frac{\alpha_N}{\gamma_0} \cdot (d - \gamma_B) + \alpha_0 & \text{if } d < \gamma_0 \\
\frac{\alpha_0}{\gamma_B - \gamma_0} & \text{if } \gamma_0 < d < \gamma_B \\
\alpha_0 & \text{if } d > \gamma_B 
\end{cases}
\]  

(1)

Algorithm 2, which describes how to add a new block to the blockchain, was created by the SecuSca. In order to compute replication over the network and guarantee scalability and security, the sharding function \( R(d) \) is activated. The quorum \( Q \) across the network \( N \) verifies the block's insertion and replication; the block is not placed until the quorum is attained.

Algorithm 2. The process of inserting a new block in the blockchain of the SecuSca

**Input:** \( B = \{b_1, b_2, ..., b_n\} \): a set of blocks in the blockchain \( B \).

\( N = \{n_1, n_2, ..., n_N\} \): set of nodes in the network.

\( b_{n+1} \): the next block, \( d \): the depth of the blockchain in a node, \( Q \): the Quorum

**Output:** \( NR_j \): The number block replication in the blockchain

1. **For each** \( bj \) where \( j \geq n + 1 \)
2. **IlStep1:** Calculate the number of replications to ensure the security
3. \( NR_j \leftarrow R(d) \)
4. **IlStep2:** Verification of the Quorum agreement
5. **While** \( Verification(Q, NR_j) = False \)
6. Go to step 1
7. **return** \( NR_j \)

An ideal blockchain function that strikes a balance between scalability and security is the SecuSca. By freeing up local disk space, it encourages scalability and enables users to store more transactions. But there still has to be work done, such enabling the blockchain to continue operating even in the event that nodes remove transactions from their local chains. Even if nodes share state, the consensus protocol should include inter-shard connections and enable data retrieval from another node if necessary.

GradingShard is a blockchain sharding mechanism that divides a blockchain network into multiple subnetworks, each processing transactions in parallel[36]. It uses network sharding to split the network into subnetworks, while transaction sharding divides each transaction into different network shards. This combination improves security and blockchain throughput, with experimental tests showing that the network can pack 500,000 transactions in 5 seconds. The consensus is Proof of Stack (PoS), and GradingShard uses an algorithm that combines VRF and PoS to determine the network shard of a node. This algorithm reduces transaction latency and increases transaction throughput while improving security. The VRF_POS algorithm enhances the conventional VRF function to allow for network sharding. The modified algorithm has an \( O(n) \) complexity. In addition to producing a collection of verifiable pseudo-random strings (result), the classic approach may also generate evidence using the message (info) and private key (SK). Instead of using a text that the verifier sends at random, the VRF_Verify function confirms that the random string is the owner of the public key (pk) that corresponds to the private key defined by the VRF generator. The algorithm's flow is as follows:

1. The prover generates a pair of public and private keys, represented as pk and sk, respectively.
2. The prover computes the VRF_HASH and VRF_PROOF functions locally, yielding res and proof, in that order.
3. The prover inputs data into the VRF_POS algorithm together with pk, sk, and proof messages (such as the blockchain network's first block hash or block height).
4. The following are the main improvements to the conventional VRF method, broken down into the prover and the verifier. The prover completes the following duties:
   - Assuming that the result value is new_result, the result res is mapped using the spatial mapping function to the range \([0, value)\) (value is the number of nodes).
The validator completes the subsequent tasks:

The first step is to confirm that the proof and res match. Obtaining the PK and information to revalidate them is the second stage; if it doesn't, the verification was unsuccessful.

Finally, validators confirm that res and new_result match. Both new_result and true results are generated upon a successful match. If not, the output is false and null.

Algorithm 3. VRF_POS

Require: height of previous block, \( h \); private key of prover, \( sk \); public key of prover, \( pk \); tokens invested by prover, \( tok \);

Ensure: result of VRF_POS function, \( result \); proof of VRF_POS function, \( proof \);

\[ res \leftarrow VRF\_HASH(sk \| h) \]
\[ proof \leftarrow VRF\_Proof(sk \| h) \]
\[ result \leftarrow Mapping(res) \]
\[ validator \leftarrow (res, proof, tok) \]

if \( res = VRF\_P2H(proof) \) \& \( res = VRF\_Verify(pk, hight, proof) \) then

\[ validator \leftarrow (pk, h) \]
\[ new\_result \in [0, value] \leftrightarrow \frac{res}{value} \]
return \((new\_result, true)\);

else

return \((res, false)\);

end if

Based on their unique identities, verifiers are categorized as leader nodes, first-level nodes, and second-level nodes. Verifiers are all nodes that operate together in network sharding and consensus. The leader belongs to none of the shards. Nodes in the first and second levels are the same shard's validators. The responsibilities of different validators are as follows:

Leader node: package and release the last block upon verification of the sub-MerkleRoot hash value and signature given by the first-level node of each shard.

First-level nodes are responsible for proposing transactions, grouping related transactions for this shard, calculating and signing the sub-MerkleRoot hash result, and finalizing consensus.

Nodes in the second tier: transmit the transaction to the network and assist in signing the sub-MerkleRoot hash result. Fig. 4: Grading-Shard depiction, all four shards in this design consist of a leader node, a first-level node, and several second-level nodes. Each component could possibly be verified only once.

To obtain the node definitions, each node performs a local run of the VRF_POS algorithm (refer to Algorithm 3), obtaining a result value. Let \( y \) be the outcome of VRF_POS;

This node is a leader node if \( y < 1 \). Along with packing the sub-Merkle root that the first-level node submitted in each network shard, the leader node is also responsible for publishing new blocks and confirming the signature of those blocks;

If \( y \geq 1 \). Either the first or second level might include the node. Determine its identity by using the node's account capital, \( w \).

This node is a first-level node if \( y/w < 1 \). To finish the consensus, these nodes must build a sub-MerkleRoot, package transactions that belong to this shard, propose transactions, and sign;

This node is a second-level node if \( y/w \geq 1 \). Assisting in the completion of the sub-MerkleRoot, spreading transactions around the network, and using these nodes to create transaction hashes.
Network sharding is configured using VRF_POS, and node identification is determined using PoS. The fundamental steps in network sharding are as follows:

- Initializing the system. A pair of public keys $pk$ and private key $sk$ is generated by every registered blockchain node in the network.
- The leader is chosen using the VRF algorithm. Let us assume that the value has 1000 nodes. Based on the random text of the complete network node—the input of type VRF algorithms—create a random integer and map it to $[0, \text{value}]$. The height of the block above, the block hash expressed as a random string $S$, and the overall network synchronization time are a few examples of these random strings. Additionally, feed type VRF algorithms with the node's private key, $sk$.

Sharding technology has been proposed by VRF_POS to improve blockchain transaction throughput. An increasing number of researchers are studying blockchain sharding technology, which is a blockchain parallel processing transaction mechanism. This scheme's examination is important to address the issue of conventional blockchains' poor transaction throughput, which is built on network sharding and transaction sharding.

The improved blockchain structure, which includes the network sharding and transaction sharding functionalities of the blockchain structure, was evaluated by VRF_POS to evaluate the effectiveness of the sharding technology scheme presented in VRF_POS. It indicated that a blockchain network's transaction volume increases with the number of shards present. VRF_POS is a linear growth relationship. VRF_POS to provide a secure and effective throughput increase methodology. A blockchain network with over 100,000 nodes may be split into 1024 shards at a ratio of around 100:1, according to our simulation. The blockchain network can process 500,000 transactions in 5 seconds with this technique.
<table>
<thead>
<tr>
<th>TL;DR</th>
<th>Results</th>
<th>Methods Used</th>
<th>Limitations</th>
<th>Contributions</th>
<th>Practical Implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB-Chain [34]</td>
<td>- Feasibility analysis of LB-Chain divided into blockchain sharding network and allocation service parts. - Transaction load imbalance affects confirmation delay, but migration reduces delay. - Account migration scheme needed to move account current state across shards.</td>
<td>- Transaction load imbalance has the most negative effect on sharding performance. - Load imbalance causes extremely long user-perceived TCD (hundreds of seconds). - Cross-shard transactions have a smaller impact on TCD (many seconds). - Load imbalance results in a substantial decrease in TPS (up to 35). - Cross-shard transactions don't affect TPS in any way.</td>
<td>- Feasibility analysis of LB-Chain divided into blockchain sharding network and allocation service parts - Ideal Allocation algorithm for theoretical upper bound of account allocation - Account allocation problem formulation - Generalization of protocol to consortium blockchains - Focus on improving performance of blockchain system in consortium blockchains</td>
<td>- Ideal Allocation algorithm is time-consuming and infeasible in real implementation. - The account allocation problem is formulated as an equation. - The protocol can be generalized to consortium blockchains. - Security and decentralization are less important in consortium blockchains. - Performance requirements are relatively high in consortium blockchains.</td>
<td>- Systematic study on imbalanced transaction load in blockchain sharding system. - Proposal of LB-Chain, a blockchain framework for load balance. - The current blockchain's sharding significantly degrades performance. - LB-Chain framework improves load balance, delay, throughput, and fairness. - LB-Chain reduces confirmation latency by up to 90%. - LB-Chain increases throughput by more than 10%. LB-Chain improves fairness by more than 60%.</td>
</tr>
</tbody>
</table>
Paper presents a parallel multiprocessor approach to solve blockchain scalability. Approach increases transactions per second and reduces mining latency and cost. Distributed approach partitions blocks among processes and achieves scalability. High throughput, low latency, and less expensive than current methods.

- Quintessential parallel multiprocessor approach decreases latency for mining.
- Mining function shows improvement with lower latency using multiple processes.
- With four processes, the lowest recorded mean execution time for mining is 1.028 seconds.
- With six processes, the highest mean execution time for mining ever recorded is 1.211 seconds.
- Making a transaction has mixed results with multiple processes.
- Transaction function execution time increases with 6 processes.
- Proposed approach achieves scalability with low latency, high throughput, and lower cost.
- Throughput of the proposed approach is 125ktps, comparable to existing approaches.

- Scalability issues in blockchain technology.
- Low throughput and high transaction latency in proof-of-work (PoW) cryptocurrencies.
- Performance degradation with larger chain sizes.
- Poor performance due to limited data capacity of the blockchain.

- The suggested method boosts the quantity of transactions per second (tps). The suggested method lowers mining latency and offers low latency, fast throughput, and reduced cost.
<p>| [35] | - The paper suggests that in blockchain systems, security and scalability must be traded off. - Introduces a dynamic sharding model to minimize replication and maintain security. - Implements the approach as a proof of concept and evaluates its efficiency. | - The experiments studied block replication efficiency and blockchain size. | - Proposed SecuSCa approach with trade-off between security and scalability. - Dynamic sharding for efficient replication and storage of blockchain data. - Optimization function introduced to aid the sharding process. - Replication-validation algorithm for validating replicated blocks over the network. | N/A | - Designs a new blockchain system called SecuSCa. - Maintains a balance and trade-off between security and scalability. - Formulates trade-off as a multi-objective optimization problem. - Allows dynamic sharding to save replications while maintaining security. - Implements the proposed approach as a proof of concept and evaluates its efficiency. | - Proposed approach, SecuSCa, makes a trade-off between security and scalability. - Allows for more storage capacity in the blockchain. - Dynamic sharding reduces replication in the network while maintaining security. - Good outcomes and a notable advancement above conventional blockchain. - Further improvements needed, such as handling saturated nodes. |</p>
<table>
<thead>
<tr>
<th>[33]</th>
<th>- FortunChain is a scalable blockchain system for state sharding. - It uses a novel Tyche consensus protocol for node selection. - The system achieves high throughput and low latency. - It ensures global consistency of transaction data through state sharding.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- FortunChain achieves a throughput of approximately 6540tx/s. - The estimated latency of FortunChain is 8.59s. - The efficiency and feasibility of FortunChain are demonstrated through simulation results.</td>
</tr>
<tr>
<td></td>
<td>- The EC's design - Sharding formation method - Sharding candidate verification node selection algorithm - VRF for node selection - Selection algorithm for accounting nodes</td>
</tr>
<tr>
<td>N/A</td>
<td>- Design of a scalable blockchain system named FortunChain. - Enable fast consensus and state sharding.</td>
</tr>
<tr>
<td></td>
<td>- FortunChain proposes a scalable sharding blockchain system to improve blockchain throughput. - It uses a novel Tyche consensus protocol and EC-VRF for efficient node selection. - Multi-period, fast, and dynamic sharding accounting improves transaction processing performance. - The verification chain ensures global consistency of transaction data through secondary verification. - Simulation results show high throughput and low latency, demonstrating the system's efficiency.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>- suggested a unique blockchain sharding architecture for the distribution of nodes.</td>
</tr>
<tr>
<td></td>
<td>takes into account variations in communication latency and shard trust, and node count difference.</td>
</tr>
<tr>
<td></td>
<td>Aims to improve shard security and performance of blockchain sharding.</td>
</tr>
<tr>
<td></td>
<td>- Existing blockchain sharding methods fail to consider shard trust difference.</td>
</tr>
<tr>
<td></td>
<td>- Using sharding technology to increase the scalability of blockchain systems; taking into account the difference in shard trust while allocating node.</td>
</tr>
<tr>
<td></td>
<td>- Improvement of shard security and performance compared to existing methods.</td>
</tr>
</tbody>
</table>
GradingShard is a sharding protocol to improve blockchain throughput. It uses network sharding and transaction sharding for efficiency and safety. It can divide a blockchain network into 1024 shards with a 100:1 ratio. It can process 500,000 transactions in 5 seconds with a throughput of 33,000 TPS. It can effectively identify and prevent double spending threats.


- Limitations of existing sharding-based consensus algorithms for distributed miners and transactions. Limitations imposed by the BFT algorithm and complexity of cross-shard transactions.

- Increasing the speed of block out. Transaction sharding scheme to identify and prevent double spending threats.

- Use of PoS consensus algorithm and VRF for network shard allocation. Improve transaction throughput and reduced latency.

- Experimental analysis shows linear growth relationship between number of shards and transaction volume.

GradingShard improves blockchain throughput for public blockchains. It can divide a blockchain network into smaller subnetworks for parallel processing. The transaction throughput can reach 33,000 TPS, applicable to substitute VISA transactions. GradingShard effectively identifies and prevents double spending threats. It combines network sharding and transaction sharding to increase blockchain security.
| DSS BD [31] | - In this paper, we combine deep reinforcement learning with state sharding. - Blockchain architecture with dynamic state sharding for safe and scalable crowdsourcing platforms. - A training technique to choose and modify the block size, block interval duration, and shard count. - The value function Q is used to lessen the impact of past choices. - The agent uses a double DQN learning method to watch and choose actions. | - Transactions modeled as data flow snapshots in the proposed DSSBD. - Implementation of blockchain state sharding with relay transactions, METIS, and VR. - Double DQN training combines number of shards, block size, and block interval. - Simulation results demonstrate the performance of the architecture. | - Combination of state sharding and deep reinforcement learning - Use of sharding module, reconfiguration module, and security module - Optimization of state sharding using deep reinforcement learning | - Blockchain is having trouble becoming more scalable. - Blockchain nodes' storage space needs are very large. - Blockchain's security and stability will decline as it becomes more scalable. - Ensuring fairness and transparency in crowdsourcing systems - Improving system efficiency and reducing information redundancy - Addressing performance limitations of current blockchain-based crowdsourcing systems - Using state sharding technology to improve scalability and decentralization - Proposing a blockchain architecture based on dynamic state sharding using deep reinforcement learning - Considering load balancing during the reconfiguration phase - Effective | - enhances the decentralization and scalability of blockchain in crowdsourcing systems. - Assures blockchain security and atomicity of cross-shard transactions. - Strives the ideal balance between scalability, security, and decentralization. - Improves the crowdsourcing system's performance. |
by resisting attacks such as transaction atomicity attack, double spending attack, sybil attack, and replay attack.
4 Open issues and future research directions

We highlight a number of outstanding difficulties and suggest further future research paths based on the examination and comparison of the evaluated systems in Section 3. In this section, we provide some open addresses following of the thorough evaluation and analysis of communication overhead that has been performed previously.

Chain sizes: Blockchain, in which users lack the ability to connect to new nodes and complete nodes fail to keep in synchronization with the current ledger, is an issue for blockchains with increasing storage requirements. The length of the ledger's history record, the frequency of adding new blocks, block size, and the amount of data stored on-chain for transaction verification and state change execution are all factors that might cause state bloat. The main difficulty consists of handling and verifying an increasing amount of data without raising the storage needs for complete nodes. Scalability issues with blockchain technology include low throughput and insufficient transaction latency in proof-of-work cryptocurrencies, decreasing efficiency as chain sizes increase, and poor efficiency because of data storage limitations. The enormous storage capacity requirements of blockchains limit their scalability, and state sharding research currently being conducted depends on account sharding, which could impact security and stability.

implementation time: Periodically, the shard allocation mechanism must relocate specific nodes to different shards. A node moving to a different shard, however, has non-negligible overhead as it must synchronize the ledger of the new shard and locate peers. The problem is known as the "reshuffling problem" in the blockchain community. The allocation procedure takes a long time to execute and is not practical in practice. It is possible to extend the protocol to consortium blockchains. Additionally, blockchains used in consortium have very high-performance requirements.

failure risk: Traditional blockchain sharding techniques fail to take differences in shard trust. Current approaches also fail to consider differences in communication latency across shards, it fails to consider the difference in node counts between shards. A higher chance of blockchain failure because of these restrictions.

consensus approaches: To avoid the overhead associated with sharding technology, participating nodes are assigned to shards according to their requirements for business. This reduces the number of times a participating node must swap out and the percentage of non-cross-shard transactions, which makes management easier and costs less. However, when the size and throughput grow over time, this results in uneven shard size and an extraordinarily long epoch reconfiguration for the nodes involved. This ultimately puts congested transactions on a single shard and hits the intra-consensus bottleneck. Sharding-based consensus techniques now in use are not meant to provide distributed miners and transactions with a fair and safe way of support. Avoided are the BFT algorithm's constraints and the intricacy of cross-shard transactions.

5 Conclusion

This study discusses the state-of-the-art sharding approaches and applies suggested criteria to assess their benefits and drawbacks. Since high communication overhead is typically an issue for cross-shard transactions using the present consensus methods, the review focused on research into low-communication-overhead sharding solutions. In addition, the study explores open research questions and makes recommendations for possible future research approaches. It offers a comprehensive and perceptive review of state-of-the-art sharding methods, pointing out approaches for more study in the future.

References


