Computer Modeling in Engineering & Sciences

Tech Science Press

Doi:10.32604/cmes.2025.074378

REVIEW

AI-Enabled Perspective for Scaling Consortium Blockchain

Yingying Zhou, Wenlong Shen, Tianzhe Jiao, Chaopeng Guo and Jie Song*

Software College, Northeastern University, Shenyang, 110819, China

*Corresponding Author: Jie Song. Email: songjie@mail.neu.edu.cn

Received: 10 October 2025; Accepted: 14 November 2025; Published: 23 December 2025

ABSTRACT: As consortium blockchains scale and complexity grow, scalability presents a critical bottleneck hindering broader adoption. This paper meticulously extracted 150 primary references from IEEE Xplore, Web of Science, Google Scholar, and other reputable databases and websites, providing a comprehensive and structured overview of consortium blockchain scalability research. We propose a scalability framework that combines a four-layer architectural model with a four-dimensional cost model to analyze scalability trade-offs. Applying this framework, we conduct a comprehensive review of scaling approaches and reveal the inherent costs they introduce. Furthermore, we map the artificial intelligence (AI)-enabled methods to the scaling approaches and analyze their effectiveness in enhancing scalability and mitigating these inherent costs. Based on this analysis, we identify the root causes of the remaining costs unresolved by AI and the new trade-offs introduced by AI integration, and propose promising research opportunities for AI-enabled consortium blockchain scalability, guiding future work toward more adaptive, intelligent, and cost-efficient consortium blockchain systems.

KEYWORDS: Consortium blockchain; scalability; artificial intelligence

1 Introduction

As a distributed ledger technology paradigm, blockchain's applications have evolved beyond its initial open and permissionless form. To meet the complex requirements of multi-party collaborative systems for compliance, performance, and governance, the consortium blockchain emerged as a permissioned architecture. Consortium blockchains are ideal for multi-party, regulated applications: improving supply chain traceability and resilience [1], ensuring auditable and controlled data sharing in healthcare [2,3], and governing multi-party transactions in finance and operations management [4]. However, trade finance platforms[1] and supply chains[2] require processing millions of transactions daily. By comparison, the throughput of mainstream consortium blockchains still primarily remains at a few thousand transactions per second (TPS) [5]. This significant gap highlights the urgent need for enhanced scalability.

To bridge this performance gap, a multitude of scaling approaches have been proposed that can be systematically categorized based on the architectural layer: the data, network, execution, and consensus layers. The approaches at the data and execution layers focus on increasing the number of transactions that can be processed in parallel. For example, the data layer employs sharding or directed acyclic graphs to enable parallel transaction confirmations [6,7], while the execution layer uses parallel and off-chain processing to increase the number of transactions handled per cycle [8,9]. In contrast, approaches at the network and

[1] https://www.swift.com/ (accessed on 12 October 2025).
[2] https://corporate.walmart.com/news/ (accessed on 12 October 2025).

consensus layers aim to reduce the consensus time and minimize propagation delays [10–12]. Designed for identity-based governance and selective information transparency, consortium blockchains sacrifice a degree of decentralization for higher performance. These scaling approaches take further measures to maximize throughput while maintaining security, which inevitably introduces costs in other dimensions, as we will discuss in more detail in Section 2.2. Meanwhile, these scaling approaches are often reliant on static pre-configurations. They cannot adapt to dynamic workloads and changing network conditions, leading to suboptimal performance.

With the widespread use of artificial intelligence (AI) for complex systems optimization, many studies have proposed AI-enabled enhancements to these scaling approaches [13]. The distinct advantage of AI-enabled scaling lies in its ability to learn and implement optimal policies in real-time and dynamic environments. For example, reinforcement learning is applied to dynamically tune network and consensus parameters [14,15], while unsupervised learning helps optimize data sharding by discovering underlying transaction patterns [16]. These intelligent methods aim to further enhance scalability through dynamic and adaptive control while also actively mitigating the inherent costs of the scaling approaches, thus seeking a better balance between performance and its associated costs. Despite the promising outlook, the path of AI enablement is not without its own challenges. On one hand, the integration of AI has not entirely eliminated all costs. On the other hand, AI technology itself brings new trade-offs, such as the resource overhead for model training and the transparency of the decision-making process.

Although previous surveys have explored scalability approaches for general blockchains [17–19], the interaction between AI and blockchain [20–23], and the performance analysis of consortium blockchains [24], a holistic analysis shows that their research relationship is absent. Those surveys on scalability often overlook the potential of AI, while those on the interaction between AI and blockchain do not focus on specific performance bottlenecks. This survey is stimulated by the impressive recent developments in scaling approaches, the rapid advances in AI-driven system optimization, and the increasing demand for high-performance consortium blockchain applications. Therefore, to bridge this gap, this study conducts a comprehensive survey that systematically maps AI-enabled enhancements to scaling approaches for consortium blockchains and systematically evaluates the associated costs. This synthesis of scaling approaches, their inherent costs, and their AI-enabled enhancements distinguishes our work from the existing surveys. Ultimately, this paper compares the performance and inherent costs of scaling approaches and the corresponding AI-enabled enhancements for the four layers, and reveals that while AI effectively reduces inherent costs and enhances performance, it often introduces new trade-offs. The foremost contributions in this study are enumerated as follows:

- Proposing a unified scalability framework that combines a four-layer architectural model with a four-dimensional cost model to systematically analyze scalability trade-offs.
- Conducting a comprehensive review of scaling approaches, using our scalability metrics to analyze their performance and inherent costs.
- Systematically mapping AI-enabled enhancements to the scaling approaches, analyzing how they enhance performance while mitigating associated costs.
- Identifying promising research opportunities to address remaining costs and analyze trade-offs for the integration of AI.

To realize the contributions mentioned above, we organize our survey based on four primary aspects. First, we propose the architecture of consortium blockchains and present the scalability metrics for evaluation. Second, guided by this architecture, we categorize the scaling approaches at each layer of the consortium blockchain and evaluate the approaches using our scalability metrics. Third, we analyze the state-of-the-art AI-enabled methods that further enhance these scaling approaches. Finally, based on the analysis above, we

propose future research opportunities for AI-enabled consortium blockchain scalability. Fig. 1 illustrates the organizational structure of this paper.

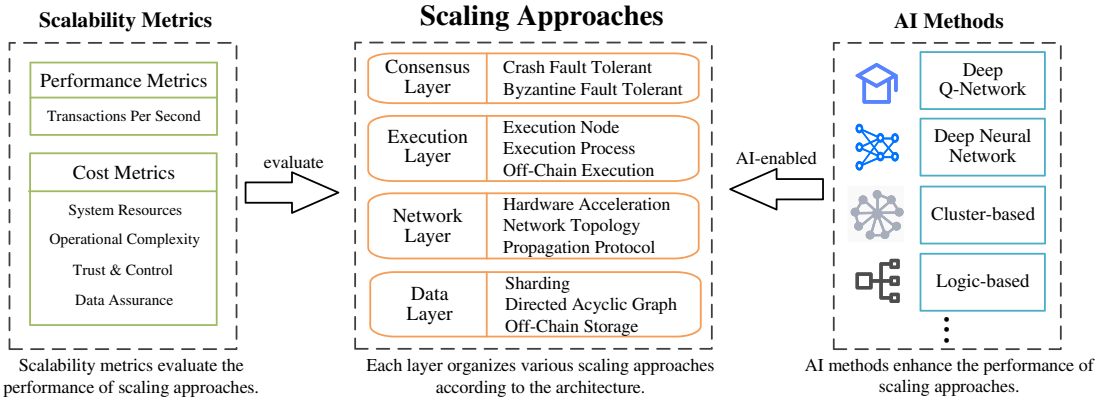

Figure 1: The organizational structure of this paper connecting AI methods, scaling approaches, and evaluation metrics. The figure highlights how AI methods enhance scaling approaches across four blockchain layers, whose effectiveness is evaluated through performance and cost metrics

2 Background

2.1 Consortium Blockchain Architecture

Blockchain, a revolutionary distributed ledger technology, has garnered widespread attention and applications globally in recent years [25]. Its core lies in the use of cryptographic principles and consensus mechanisms to ensure data security, immutability, and consistency, thus constructing a decentralized trust system [26]. Blockchain is categorized into three main types: public blockchains, private blockchains, and consortium blockchains [27]. Fig. 2 illustrates the differences among the three blockchains. Consortium blockchains, positioned between public and private chains, achieve more efficient transaction confirmation and more flexible permission management through pre-defined node admission mechanisms and consensus algorithms. Currently, various consortium blockchain systems have emerged, each optimized and designed for different application scenarios. Some of the well-known consortium blockchain systems include Hyperledger Fabric [28], R3 Corda [29], Quorum [30], Hyperledger Besu, and so on.

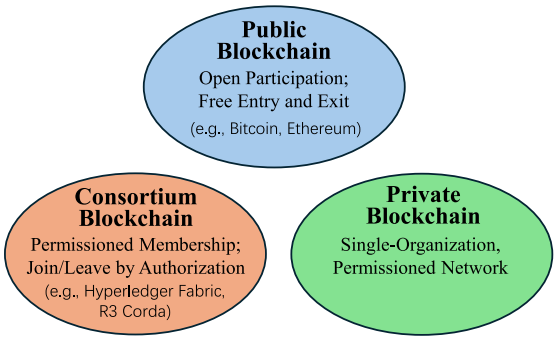

Figure 2: The three main types of blockchain are distinguished by their participation and access control models: public blockchains feature open participation, consortium blockchains operate on permissioned membership, and private blockchains are controlled by a single organization

Although these consortium blockchain systems have different focuses in their specific implementations, such as differences in consensus algorithms, data storage, and smart contract engines, they all follow similar design principles and have a common four-layer architectural model, including the data layer, network layer, consensus layer, execution layer, and application layer.

- **Data layer** is the foundation of the consortium blockchain architecture, responsible for storing data on the blockchain. The main function of this layer is to provide a secure and reliable data storage mechanism, ensuring data integrity and durability. The data layer needs to support efficient data access and querying, and be able to adapt to the ever-growing amount of data.
- **Network layer** is responsible for communication and data propagation between consortium blockchain nodes. The consortium blockchain network needs to provide efficient and stable communication protocols to ensure that nodes can synchronize data on time. The network layer also needs to have node discovery and management functions to facilitate the addition of new nodes and the maintenance of existing nodes.
- **Execution layer** is responsible for executing smart contracts and processing transactions. The consortium blockchain needs to provide a secure and trustworthy execution environment to support the deployment and execution of smart contracts. The execution layer is responsible for processing transactions, executing smart contract code, and updating the blockchain state.
- **Consensus layer** is the core of the consortium blockchain, responsible for reaching consensus on the state of the blockchain. Unlike public blockchains, consortium blockchains typically employ more efficient consensus algorithms to achieve faster transaction confirmation. The consensus layer is responsible for verifying the validity of new blocks and resolving any forks that may occur, ensuring the consistency of the blockchain.

2.2 Scalability Metrics

Consortium blockchains are typically applied in scenarios with high-performance requirements [31]. Examples include complex supply chain networks that may generate hundreds of thousands or even millions of transactions daily, and financial applications like cross-border payments and trade finance that have extremely high demands for transaction speed and throughput. Therefore, scalability is a crucial performance indicator for consortium blockchains. The essence of a blockchain is a distributed, immutable write-only database or ledger. Its primary operation is packaging new transaction data into blocks and appending them to the end of the chain. Therefore, the most intuitive performance metric for assessing its scalability is throughput, i.e., the number of transactions the system can process per unit of time (TPS) [32,33], which is defined as:

$$TPS = \frac{N_{tx}}{\Delta t_{round}} \tag{1}$$

where N_{tx} represents the total number of transactions packaged and appended to the chain per round, and Δt_{round} represents the time required for the round. The higher the TPS, the more transactions the blockchain system can handle, and the better its scalability.

Blockchain faces a well-known challenge known as the Trilemma [34,35]. This trilemma posits that it is difficult for a blockchain system to simultaneously optimize three core attributes: Decentralization, Security, and Scalability, as at most only two can be prioritized. Traditional public blockchains prioritize maximizing decentralization and security [36,37], which comes at the direct cost of limited scalability. To meet the demands of high-performance scenarios, consortium blockchains make a fundamentally different strategic trade-off within this trilemma: prioritizing security and scalability while sacrificing a

degree of decentralization. However, to satisfy the needs of increasingly demanding scenarios, researchers are pushing the boundaries of performance while striving to maintain or even strengthen the inherent security model of consortium blockchains. This intensified pursuit of scalability forces deeper compromises on decentralization and security. For instance, this pursuit often requires direct investments in system infrastructure and the management of more intricate operational processes. These actions, in turn, create consequences in other domains: control may become concentrated onto fewer components, while novel designs can introduce new challenges for guaranteeing data integrity and availability across a distributed system. We introduce a four-dimensional cost framework that measures the costs incurred to enhance scalability. Fig. 3 presents the relationship between the blockchain trilemma and our four-dimensional cost framework.

- **System Resources:** Measures the direct investment in foundational infrastructure required for scalability. This investment aims to either increase the number of transactions per round (N_{tx}) through architectural capacity expansion or decrease the time per round (Δt_{round}) through faster processing. For instance, to achieve higher throughput, a system might undergo capacity expansion by adding more nodes or memory, or enable faster processing by upgrading to high-performance servers and remote direct memory access (RDMA) capable networks. This direct capital and energy expenditure represents a tangible increase in system resources.
- **Operational Complexity:** Measures the management burden required for scalability. This burden arises from implementing and maintaining intricate architectures designed to either increase transaction throughput (N_{tx}) via structural parallelism or reduce confirmation latency (Δt_{round}) via end-to-end path compression. For instance, a sharded blockchain achieves structural parallelism but introduces significant management burden, such as handling dynamic reconfiguration and cross-shard protocols. Similarly, achieving path compression through optimized network topologies requires continuous monitoring and complex policy management. This ongoing effort to manage such intricate systems represents a significant increase in operational complexity.
- **Trust & Control:** Quantifies the consequential cost to decentralization, measuring the additional governance and oversight mechanisms required to diminish centralization pressure when scalable designs delegate critical responsibilities to a minority of roles or specialized components. For instance, to accelerate consensus, a system might rely on a small, fixed committee of high-performance nodes. This design concentrates power, thus incurring a trust & control cost as the system must now introduce rotation policies and heightened monitoring to manage the risks of this centralization.
- **Data Assurance:** Quantifies the consequential cost to security, measuring the additional cryptographic and protocol mechanisms required to enhance security assurance when scalable architectures physically or logically separate data and validation processes. For instance, an off-chain execution solution improves scalability by moving computation off the main chain. To ensure the results are correct and verifiable, the system must now bear the cost of generating and verifying cryptographic proofs. This additional computational and protocol overhead is the data assurance cost.

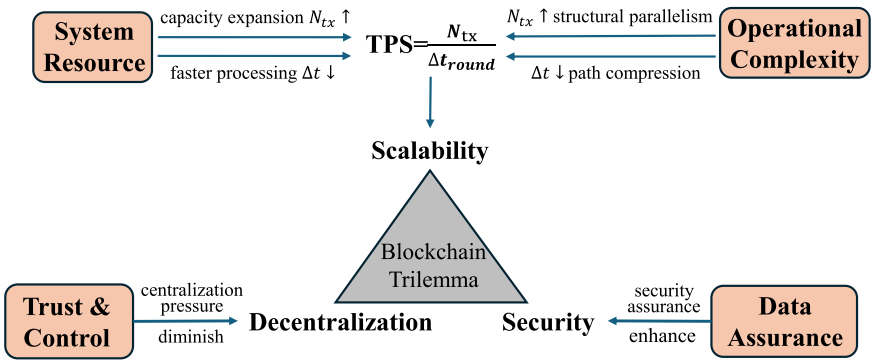

Figure 3: The relationship between the blockchain trilemma and our four-dimensional cost framework. The pursuit of higher scalability (TPS) involves increasing system resources to expand transaction capacity (N_{tx}) and accelerate processing (Δt_{round}), or accepting higher operational complexity to manage structural parallelism and compress critical paths. This effort creates direct trade-offs with the other two vertices, where trust & control measures the effort to diminish centralization pressure, and data assurance quantifies the cost to enhance security assurance

3 Scaling Approaches

As established in the previous section, enhancing scalability is a critical objective for consortium blockchains, but achieving it introduces significant costs across system resources, operational complexity, trust & control, and data assurance. This section provides a systematic and comprehensive review of the scaling approaches from four layers: data layer, network layer, execution layer, and consensus layer.

3.1 Data Layer Approaches

The data layer serves as the foundation of the consortium blockchain, responsible for storing transaction records and world states, and supporting the execution, consensus, and network layers. The organization of the data layer directly affects concurrency, conflict detection, and the complexity of consensus, thereby exerting a direct impact on the scalability of the consortium blockchain. However, most traditional consortium blockchains adopt a linear chain structure, where block generation and commitment must be globally serialized, resulting in low transaction-level parallelism and becoming a critical bottleneck to scalability. To overcome these limitations, researchers have proposed various improvements at the data layer, which can be categorized into three types: sharding, which partition the global ledger into multiple parallel shards; directed acyclic graph, which allow multiple transactions or blocks to be committed concurrently to break the restriction of linearity; and off-chain storage, which move part of the transaction data or states off-chain to reduce the workload on the main chain.

3.1.1 Sharding

Sharding is a scaling technique that partitions a blockchain system into multiple parallel shards so that different subsets of work can be handled concurrently [38]. In practice, sharding can target different objects: network sharding partitions nodes as illustrated in Fig. 4a, transaction sharding routes execution to different shards as shown in Fig. 4b, and state sharding assigns the world state to specific shards as depicted in Fig. 4c [39]. In fact, network sharding and transaction sharding primarily affect other layers, such as execution scheduling, parallelism, and peer-to-peer (P2P) propagation, and do not change the storage structure in the data layer. They will be discussed in later sections. By contrast, only state sharding directly impacts the data layer.

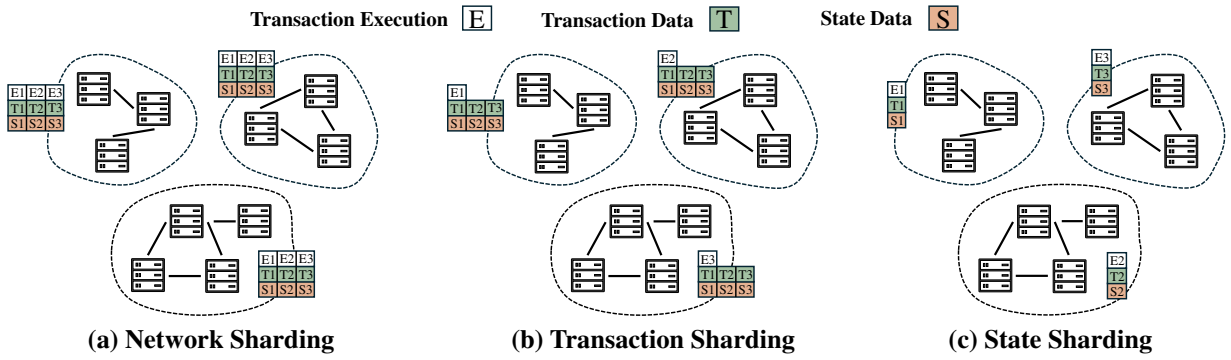

Figure 4: Schematic diagram of (**a**) Network Sharding, (**b**) Transaction Sharding, and (**c**) State Sharding, illustrating how transaction execution, transaction data, and state data are distributed across shards

State sharding partitions the global world state across shards and routes each transaction to the shards that own the touched keys, so that shards maintain and update only their local slice of the ledger [6,40–42]. A representative baseline in consortium blockchains is SharPer, proposed by Amiri et al. [6], which maps data shards to pre-formed clusters. Each cluster executes intra-shard transactions locally and stores only its local view. This design reduces the storage footprint and communication overhead for each node and allows multiple clusters to process disjoint workloads in parallel. Building on SharPer, Matani et al. [40] balance nodes into shards and then organize each shard into level groups in a hierarchical tree. This structure minimizes both the storage and communication overhead by localizing data and consensus, enabling each shard to process its workload with high autonomy. To address the high overhead of cross-shard transactions, Hong et al. [41] propose Pyramid, a layered sharding system. Instead of complete isolation, Pyramid introduces specialized bridge shards that store the full records of multiple internal shards. This design enables cross-shard transactions to be processed internally within a single bridge shard, committing them in one consensus round and significantly boosting throughput.

3.1.2 Directed Acyclic Graph

Unlike the traditional linear blockchain, Directed Acyclic Graph (DAG)-based approaches allow multiple nodes to be appended concurrently, avoiding the strict serialization bottleneck and thereby significantly enhancing system concurrency and throughput [43]. By maintaining partial order within the graph, these designs preserve consistency while supporting higher parallelism. Depending on what each DAG node represents, such approaches can be broadly classified into two categories: block-as-node DAG, where each node is a block containing multiple transactions and scalability is achieved through concurrent block generation to improve consensus efficiency; and transaction-as-node DAG, where each node corresponds to a single transaction and edges capture dependency relations, enabling fine-grained parallel execution.

For a transaction-as-node DAG, as shown in Fig. 5a, each transaction is represented as a node, with edges encoding dependencies or conflicts. Nexus proposed by Zhang et al. [7] builds on this by combining local and global DAGs: each shard maintains a local DAG to execute non-conflicting transactions in parallel, while the global DAG guarantees cross-shard consistency and conflict-equivalence to sequential execution. This directly increases concurrency in the execution layer and also impacts the sharding and consensus layers, ensuring global correctness across shards and ultimately enhancing system throughput.

For a block-as-node DAG, as shown in Fig. 5b, each node represents a block containing multiple transactions, with edges denoting inter-block references [44–47]. JointGraph, introduced by Xiang et al. [46], exemplifies this category by integrating the DAG structure with an efficient consensus mechanism, enabling

multiple blocks to be confirmed in parallel within the same round. This directly improves block production in the consensus layer and indirectly accelerates transaction validation and state updates in the execution layer, thereby reducing end-to-end latency.

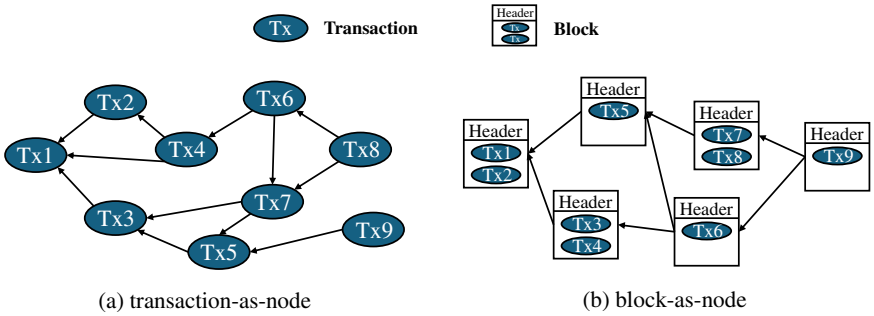

Figure 5: A comparison of the two primary DAG models: (**a**) a transaction-as-node DAG, where each vertex represents a single transaction, and (**b**) a block-as-node DAG, where each vertex is a block containing multiple transactions

3.1.3 Off-Chain Storage

Off-chain storage approaches refer to moving part of the blockchain's transaction data or states off the chain, delegating them to external storage nodes or multi-chain structures, while retaining only lightweight indexes or commitments on-chain [48–52]. It alleviates on-chain data load, making block generation and validation more efficient, and it allows compute- or storage-intensive tasks to be handled off-chain, shortening the on-chain critical path and increasing throughput.

Feng and Deng [48] propose a hot/cold data separation scheme, migrating cold blocks to an off-chain file system and keeping only hot blocks and essential index information on-chain. By reducing the on-chain storage footprint, this scheme alleviates the I/O and memory pressure on nodes, allowing them to validate and commit new blocks more efficiently. SlimChain designed by Xu et al. [49] adopts a stateless blockchain design that moves contract execution and state storage entirely off-chain, with only state roots and commitments retained on-chain. This stateless design drastically reduces the on-chain storage requirements and simplifies the data validation process, as nodes only need to verify lightweight commitments instead of re-executing transactions against a large state. This layered approach is validated by recent public blockchain engineering, such as the Ethereum Dencun upgrade [53]. This upgrade introduced Blobs, a dedicated and low-cost data availability layer specifically to support Layer 2 Rollups, a form of off-chain execution. This real-world deployment demonstrates the scalability benefits of separating data assurance from state execution, validating it as a promising direction for reducing on-chain data load and the associated system resources cost for high-throughput consortium chains.

Key insight: Table 1 summarizes the effectiveness of scaling approaches in the data layer. The scaling approaches achieve significant scalability improvements by restructuring the ledger to break serial bottlenecks. Regarding the impact on TPS, Sharding and DAG directly increase the number of transactions processed per unit of time (N_{tx}) through parallel processing, while DAG and Off-chain storage reduce transaction confirmation time (Δt_{round}) by shortening the on-chain critical path. However, this performance improvement introduces a clear cost profile across several dimensions. Specifically, complex cross-shard or off-chain protocols increase data assurance costs due to the need for intricate verification and rollback mechanisms. The introduction of new data structures, like those in DAGs, and complex management logic, such as shard reconfiguration, raises system resources and operational complexity. Furthermore, a

dependency on special nodes, such as bridge shards, sorters, or external off-chain components, introduces centralization risks for trust & control.

Table 1: The impact on TPS, cost, and performance of the scaling approaches in the data layer. (SR: System Resources, OC: Operational Complexity, TC: Trust & Control, DA: Data Assurance)

Scaling approaches	Ref.	Impact on TPS		Cost				Performance
		N_{tx}	Δt_{round}	SR	OC	TC	DA	
Sharding	[6]	↑	–	▲	✓	▲	✓	Outperforms sharded competitor AHL-B by 15% and non-sharded systems by over 2.7×.
	[40]	↑	–	▲	✓	▲	✓	Outperforms Rapidchain, Repchain, and Pyramid by approximately 7×, 6×, and 5×, respectively.
	[41]	↑	↓	✓	✓	✓	▲	Improves throughput up to 2.95 times over complete sharding systems (17-shard setup).
Directed acyclic graph	[44]	↑	↓	▲	✓	▲	▲	CDAG achieves >2000 TPS, a 28.57% throughput improvement over the Ripple protocol.
	[45]	↑	↓	▲	▲	–	✓	DePoA achieves 2.47× higher throughput and 5.76× lower latency than Clique (8-node setup).
	[46]	–	↓	▲	✓	✓	✓	At 150 nodes, JointGraph achieves ~60 events per second (EPS) throughput vs. ~3 EPS for Hashgraph.
Off-Chain storage	[48]	–	↓	✓	▲	▲	✓	Boosts transaction throughput 2.6-5.2× and query throughput 1.3-1.8× over normal blockchains.
	[51]	–	↓	✓	▲	–	✓	With 10 blockchains, GAM's throughput is over 4× higher than the next best competitor, Gravity.

Note: (↑: Increase, ↓: Decrease, ✓: Major cost, ▲: Minor cost, –: Not significant)

AI-enabled methods offer a promising direction to mitigate these costs. AI can automate the discovery of optimal data partitions, which significantly reduces operational complexity. This intelligent partitioning also leads to more efficient workload distribution and lower system resource requirements. Furthermore, AI can enhance data assurance by assessing node trustworthiness to ensure the integrity of data within shards, thereby reducing the reliance on costly verification protocols.

3.2 Network Layer Approaches

In consortium blockchains, the network layer, as the fundamental component for data propagation and node interaction, plays a critical role in system scalability. It not only determines the efficiency of transaction and block propagation among nodes but also affects the speed and stability of consensus. Based on existing studies, this paper categorizes scalability approaches at the network layer into three main types: hardware acceleration, network topology, and propagation protocol. Among these, network topology and the propagation protocol interact closely to influence overall propagation efficiency. For instance, an inefficient

combination, such as a random P2P overlay combined with the basic Gossip protocol as shown in Fig. 6, can lead to significant block transmission delays and redundant messaging. Conversely, optimizing these two aspects in tandem yields substantial gains. Fig. 7 shows that an effective synergy based on the Hyperclique overlay coupled with a clique-based relaying protocol [54] achieves efficient propagation.

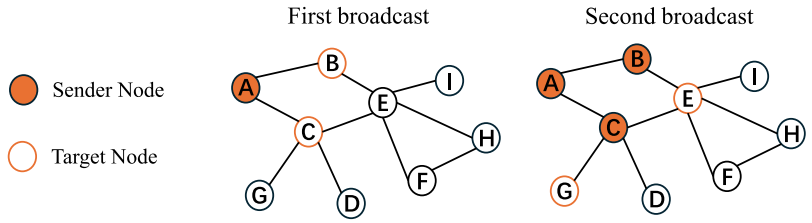

Figure 6: Message propagation on a random P2P overlay, where nodes connect arbitrarily, using the Gossip protocol, where nodes randomly forward messages to neighbors. After two broadcasts, 5 out of 9 nodes are informed, showing a gradual and potentially redundant spread

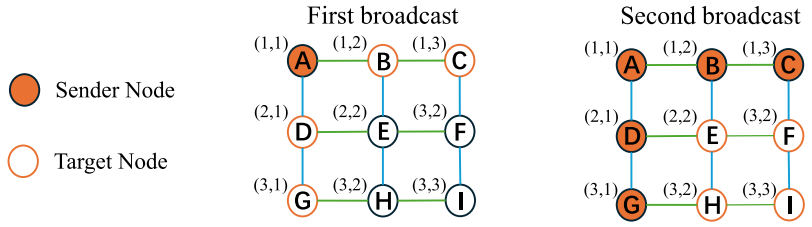

Figure 7: Message propagation on a Hyperclique overlay, where nodes form fully connected cliques based on coordinates, using the clique-based relaying protocol, where nodes relay messages to their neighbors in other cliques [54]. After two broadcasts, all 9 nodes are informed, achieving structured full network coverage

3.2.1 Hardware Acceleration

Beyond topology and protocol-level improvements, hardware acceleration has emerged as an important means of enhancing scalability. In datacenter and cloud environments, high-bandwidth, low-latency interconnects and efficient memory primitives can substantially reduce communication and copying overhead, enabling higher concurrency and throughput [55–57].

Bidl introduced by Qi et al. [55] leverages large batching, zero-copy transfers, and parallel processing in datacenter settings to cut redundant messaging costs. By reducing per-transaction overhead, it allows significantly more transactions to be processed within a given time window. In contrast, another line of work focuses more directly on high-performance interconnects.

CloudChain proposed by Xu et al. [57] combines a shared-memory model with RDMA in cloud environments. By treating inter-node communication as direct memory access (reads/writes) rather than traditional network packet exchanges, it minimizes the overhead of data serialization and network protocol processing. This results in lower per-transaction communication latency, which is crucial in the closely-coupled environment of a cloud.

3.2.2 Network Topology

Network topology strongly affects propagation efficiency and scalability in the network layer, and its optimization can be considered from two aspects: structural design and policy control. Structural design concerns the connection patterns and overall layout of nodes, which define transmission paths and

redundancy, while policy control focuses on dynamic neighbor selection and forwarding strategies within a given structure to improve efficiency and resource use. The former reflects static layout, whereas the latter emphasizes dynamic adjustment. Accordingly, this paper classifies network topology approaches into structural design optimization and policy control optimization.

In terms of structural design optimization, approaches fall into two types: overlay-centric broadcast [10,54,58,59] and network sharding [60]. For the former, the overlay is engineered as the dissemination plane. Blocks follow planned overlay routes rather than flood gossip, which bounds fan-out and shortens network diameter. vCubeChain [10] uses vCube's failure detection for leader election and an $O(\log n)$ reliable broadcast that can self-reconfigure under faults. Hyperclique [54] arranges nodes into overlapping cliques and relay blocks via intra-clique forwarding plus shared-node cross-clique relays. For the latter, network-sharding designs partition the peer set into shard-local subgraphs, using shard-internal gossip and gateway/relay links across shards to narrow broadcast domains, Zhou et al. [60] partitions peers into several shards each epoch using unpredictable randomness; a boss shard aggregates micro-blocks from normal shards and issues a signed routing table for the next epoch. This structural partitioning narrows broadcast domains and shortens transmission paths. Because membership is re-sampled every epoch and routes are refreshed, it also exhibits policy control optimization.

In terms of policy control optimization, different studies have proposed two methods from the node side [61,62] and the network control side [63,64]. On one hand, Hao et al. [61] design a trust-aware P2P topology in which nodes dynamically select reliable neighbors and remove inactive or low-trust peers. This strategy optimizes data propagation paths in real-time, reduces message redundancy, and mitigates network congestion, leading to lower and more stable block broadcast latency. From a centralized control perspective, Deshpande et al. [63] apply the Software-Defined Networking (SDN) paradigm. An SDN controller maintains a global view of the P2P network, allowing it to dynamically reconfigure the topology and optimize routing paths based on real-time traffic conditions. This centralized orchestration prevents network bottlenecks and ensures high propagation rates.

3.2.3 Propagation Protocol

The block propagation protocol directly determines the efficiency of block and transaction dissemination, making it a critical factor affecting throughput and latency. Existing studies optimize propagation from two complementary directions: relay node selection optimization, which focuses on selecting appropriate relay nodes to reduce redundancy and congestion; and block broadcast method optimization, which emphasizes improving how blocks are transmitted and combined to shorten critical paths and reduce communication complexity.

For relay node selection optimization, since consortium blockchains commonly rely on Gossip for block propagation, most existing works focus on enhancing Gossip-based protocols. Some methods aim to reduce redundancy and improve load balancing. For example, Matching-Gossip [65] and Fair Gossip [66] employ refined neighbor discovery and push strategies. These methods reduce redundant message transmissions and alleviate network congestion, ensuring a more uniform and rapid block delivery across the network. Another line of research introduces hierarchical and trust-based control. DC-SoC proposed by Dong et al. [67] partitions peers into a structured skeleton via density clustering and social credibility. This design creates more deterministic and efficient dissemination paths, reducing the number of redundant transmissions and shortening the average block propagation latency.

For block broadcast method optimization, studies aim to reduce the payload size and the complexity of asynchronous protocols [68–72]. The method proposed by Zhao et al. [68] broadcasts only block metadata

first. This approach allows peers to pre-validate headers and fetch full block bodies on demand, which cuts redundant data transmission and shortens the critical path for block validation across the network. Another method proposed by Bai et al. [69] leverages scalable multi-secret sharing with proxy re-encryption to reduce asynchronous reliable broadcast complexity from $O(n^3)$ to $O(n^2)$, directly lowering the protocol's latency. In addition, the asynchronous weak secret sharing preprocessing mechanism [70] continuously generates cryptographic materials in advance. This allows the online broadcast phase to proceed with minimal cryptographic overhead, significantly accelerating the execution of parallel reliable broadcast and Byzantine agreement protocols.

Key insight: Table 2 presents the effectiveness of the scaling approaches in the network layer. The scaling approaches enhance scalability by reducing communication latency and accelerating data propagation. Regarding the impact on TPS, all three approaches focus on reducing the time per round (Δt_{round}) by optimizing data transmission paths, minimizing redundant messages, and lowering per-transaction overhead. The costs associated with these approaches are substantial and follow a consistent pattern. The need for specialized equipment in hardware acceleration and the intricate management required for all three methods lead to higher system resources and operational complexity. Moreover, the introduction of centralized components, such as backbone nodes in Network Topology or coordinators in Propagation Protocols, increases trust & control risks. AI-enabled methods offer a promising direction to mitigate these costs. AI can automate the design of optimal network topologies and dynamically adjust propagation protocols, which significantly reduces operational complexity. This intelligent optimization also leads to more efficient data transmission and lower system resource requirements.

Table 2: The impact on TPS, cost, and performance of the scaling approaches in the network layer. (SR: System Resources, OC: Operational Complexity, TC: Trust & Control, DA: Data Assurance.)

Scaling approaches	Ref.	Impact on TPS		Cost				Performance
		N_{tx}	Δt_{round}	SR	OC	TC	DA	
Hardware acceleration	[55]	–	↓	✓	✓	▲	▲	Compared to Hyperledger Fabric and FastFabric, Bidl reduces latency by 60.2% and improves throughput 3.3×.
	[56]	–	↓	✓	✓	▲	–	Compared to the EoS blockchain, BoR reduces initial block sync latency by 20.2%.
Network topology	[54]	–	↓	▲	✓	✓	–	Compared to a hypercube, Hyperclique achieves full network coverage in more than half as many broadcast rounds.
	[61]	–	↓	▲	✓	✓	–	BlockP2P-EP reduces block broadcast latency by nearly 90% compared to Bitcoin's network.
	[64]	–	↓	✓	✓	▲	–	Achieves approximately 0.3 s transaction time, vastly outperforming Bitcoin (600 s) and Ethereum (10 s).
	[65]	–	↓	▲	✓	▲	–	In a 32-node network, Matching-Gossip reduces convergence time by about 43.5% compared to Gossip.

(Continued)

Table 2 (continued)

Scaling approaches	Ref.	Impact on TPS		Cost				Performance
		N_{tx}	Δt_{round}	SR	OC	TC	DA	
Propagation protocol	[68]	–	↓	✓	✓	▲	▲	The paper states its mechanism can improve TPS to over 20,000, a significant increase over traditional blockchains.
	[69]	–	↓	✓	✓	–	–	Reduces latency by 57.41% compared to the standard reliable broadcast protocol.

Note: (↑: Increase, ↓: Decrease, ✓: Major cost, ▲: Minor cost, –: Not significant)

3.3 Execution Layer Approaches

In consortium blockchains, the execution layer is primarily responsible for running smart contract logic on received transaction requests to produce execution results and read-write sets. Its performance directly determines the number of transactions that can be successfully processed and committed to the ledger per unit of time, and thus has a significant impact on throughput. On the one hand, the selection of execution nodes and the distribution of workloads affect overall resource utilization and efficiency, so unreasonable allocation or redundant execution can noticeably constrain TPS. On the other hand, the computational cost and concurrency capability of the execution layer determine the speed of transaction processing; excessive latency or conflicts can reduce effective throughput. To address these challenges, existing studies propose three main scalability approaches at the execution layer: execution node selection, which focuses on allocating nodes and balancing workloads; execution process optimization, which enhances concurrency and reduces redundant operations; and off-chain execution, which migrates part of the execution workload to external environments and only performs result verification on-chain.

3.3.1 Execution Node Selection

In some consortium blockchains, transaction requests are typically distributed through static configuration or random selection. This often results in certain execution nodes becoming overloaded under high concurrency, while others remain idle, leading to performance bottlenecks. To address this issue, researchers usually move in two directions. One improves scalability by directing requests to executors with lighter loads, which balances resource utilization across nodes and sustains higher throughput under heavy concurrency. The other enhances scalability by distributing requests to a group of executors that work in parallel, allowing the system to process more transactions simultaneously and expand capacity in proportion to the level of concurrency. Following this intuition, we discuss load-aware routing and transaction sharding as two approaches.

For load-aware routing, the key to scalability lies in preventing a few execution nodes from becoming persistent hotspots [73,74]. By steering requests toward nodes whose resources are less utilized, the system balances workloads across the network and reduces the chance of bottlenecks. Liu et al. [73] compute a load indicator from metrics such as CPU usage, memory consumption, and endorsement latency to guide request routing, while Cao et al. [74] rely on a ZooKeeper-based monitoring service that continuously collects node status and enables clients to select nodes dynamically. Both methods simplify the client's decision process and ensure a more balanced distribution of endorsement tasks, which prevents individual execution nodes from becoming bottlenecks under high concurrency.

For transaction sharding, scalability is achieved by decomposing the execution workload across multiple groups of nodes, so that each group only processes a portion of the transaction stream. This design reduces contention within any single group and enables parallel execution across shards, effectively expanding the system's processing capacity as the number of shards grows. Meepo proposed by Zheng et al. [75] introduces multiple execution environments per organization. Meepo enables the parallel dispatch and execution of transactions within a single organization, directly breaking the sequential processing bottleneck at the endorsement stage.

3.3.2 Execution Process Optimization

The execution stage directly determines the compute and I/O resources consumed by each transaction, the probability of conflicts and rollbacks, and the latency on the critical path. Therefore, it has the most direct impact on system throughput and end-to-end confirmation time: the faster the execution and the fewer the conflicts, the more valid transactions can be committed per unit time. We categorize execution optimizations into two classes: (1) transaction batch execution, which aggregates multiple transactions into batches to amortize per-transaction fixed costs and consolidate verification; and (2) transaction parallel execution, which increases concurrency at the execution layer through dependency analysis, conflict detection, and concurrency control.

The transaction batch execution process can be divided into batch formation [76,77] and intra-batch execution [78–80]. The former focuses on how transactions are aggregated and scheduled into batches. For example, Carbon proposed by Camaioni et al. [76] groups payment transactions into batches. This approach amortizes the fixed overhead costs associated with each transaction's execution and validation, allowing more transactions to be processed with the same amount of computational resources. The latter emphasizes how to efficiently process transactions once the batch has been formed. A representative work proposed by Thakkar et al. [78] optimizes Hyperledger Fabric's execution pipeline by enabling parallel verification and adopting batched read/write interfaces, which significantly reduces redundant computation within the execution phase.

The transaction parallel execution process can be divided into conflict detection before execution and execution control during parallel processing [8,81–84]. For example, SP-PoR introduced by Wang et al. [8] performs conflict detection by classifying transactions into normal and contract types. It then uses a clustered DAG structure for execution control, which allows non-conflicting transactions to be executed concurrently within the same epoch, thereby achieving partial parallelism at the execution layer while ensuring correctness. Jin et al. [81] conduct conflict detection by concurrently executing transactions and recording their read-write sets to build a dependency graph. During the validation phase, nodes deterministically replay these independent subgraphs in parallel, which maximizes the degree of concurrent execution while maintaining serializability and consistency.

3.3.3 Off-Chain Execution

Off-chain execution refers to performing part of transaction processing or state storage outside the main blockchain, while only committing necessary results, summaries, or proofs back on-chain. Fig. 8 illustrates this general architecture for off-chain execution. This approach reduces the computational and storage burden on on-chain execution nodes, which alleviates block congestion and consensus overhead, and thereby significantly improves system throughput. Depending on application scenarios and design objectives, existing off-chain execution methods can be broadly divided into two categories: payment-oriented off-chain execution, which leverages payment channels or dedicated settlement mechanisms to improve the efficiency

of high-frequency payments; and general-purpose off-chain execution, which typically adopts batching, off-chain storage, and proof verification to scale smart contracts and diverse transaction types.

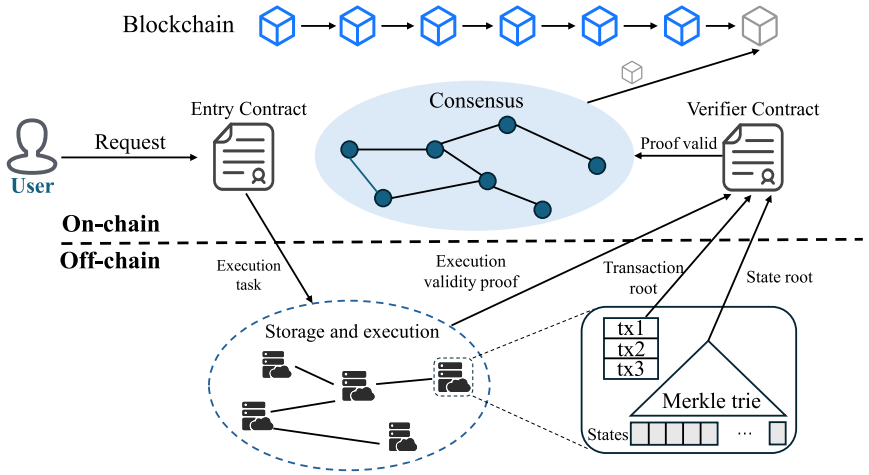

Figure 8: General architecture for off-chain execution. User requests, initiated via an on-chain Entry Contract, trigger execution and state updates within an off-chain cluster. The cluster processes these tasks and submits state roots and transaction roots along with a validity proof back to an on-chain Verifier Contract. Final commitment to the main blockchain ledger occurs after successful verification and subsequent on-chain consensus

In payment-oriented off-chain execution, the goal of off-chain execution is to reduce the overhead of frequent on-chain transfers. Xiao et al. [85] introduce multi-party payment channels, where most payments are settled off-chain. This removes a large volume of simple payment logic from the on-chain execution pipeline, freeing up resources for more complex transactions. CBOP introduced by Yang et al. [9] adopts a dynamic partitioning algorithm to decide whether transactions should be executed on-chain or off-chain. By moving a significant portion of the payment processing workload off-chain, these methods effectively reduce contention and processing load on the on-chain execution layer.

In general-purpose off-chain execution, the goal of off-chain execution is to reduce on-chain computation and storage overhead. ScorpioBase designed by Sui et al. [86] targets supply chain traceability by performing batch processing and storage off-chain. This design minimizes the on-chain execution to only verifying batch summaries, rather than processing each traceability record individually. SlimChain proposed by Xu et al. [49] introduces a stateless blockchain framework that moves contract execution and state storage entirely off-chain. This fundamentally transforms the role of the on-chain execution layer from complex computation to lightweight proof verification, which drastically reduces the computational resources required per transaction.

Key insight: Table 3 illustrates the effectiveness of the scaling approaches in execution layer. Execution layer approaches offer diverse strategies to enhance throughput, ranging from optimizing workload distribution to fundamentally altering the execution model itself. Regarding the impact on TPS, execution node selection improves throughput by efficiently routing workloads to avoid bottlenecks; execution process optimization directly boosts the number of valid transactions (N_{tx}) and reduces processing time (Δt_{round}) through parallel and batch processing; and off-chain execution provides the most significant gains by moving computation off-chain, which drastically cuts the on-chain critical path (Δt_{round}) and frees up resources. These performance gains introduce distinct costs. A common thread is the significant increase in operational complexity, as all methods require more sophisticated management or conflict resolution logic. Off-chain execution introduces major data assurance costs due to its reliance on verifiable proofs, and also adds

significant system resources from a holistic system perspective. Finally, trust & control costs can emerge when centralized schedulers or specialized off-chain nodes are introduced.

Table 3: The impact on TPS, cost, and performance of the scaling approaches in the execution layer. (SR: System Resources, OC: Operational Complexity, TC: Trust & Control, DA: Data Assurance)

Scaling approaches	Ref.	Impact on TPS		Cost				Performance
		N_{tx}	Δt_{round}	SR	OC	TC	DA	
Execution node selection	[73]	–	↓	▲	✓	✓	–	Compared to Fabric's native service discovery, the proposed system improves overall throughput by 30%.
	[75]	↑	–	▲	✓	▲	▲	Meepo achieves over 140,000 cross-shard TPS, enhancing efficiency over traditional consortium sharding methods.
Execution process optimization	[78]	–	↓	▲	✓	–	–	The proposed lazy validation approach improves the throughput by 58% compared to the baseline Fabric.
	[8]	↑	↓	▲	✓	–	✓	Compared with PoR-Chain, the transaction throughput of SP-PoR is improved by about 1.5 times.
	[81]	↑	↓	✓	✓	▲	–	The protocol's throughput is 1.6× of S-BFT and 1.2× of Aria, respectively.
Off-chain execution	[9]	↑	↓	✓	▲	–	▲	The throughput of off-chain payment is about 40 times that of on-chain payment.
	[86]	↑	–	✓	✓	▲	✓	ScorpioBase can improve throughput by up to 12.3× compared to Hyperledger Fabric.
	[49]	↑	↓	✓	▲	▲	✓	Compared to Fabric#, SlimChain improves permissioned throughput by 1.4× to 2.6×.

Note: (↑: Increase, ↓: Decrease, ✓: Major cost, ▲: Minor cost, –: Not significant)

Alleviating these resulting costs is a prime opportunity for AI-driven optimization. By automating the sophisticated logic for workload distribution and intelligently determining which tasks to offload in real-time, AI directly addresses the challenge of high operational complexity. This dynamic management also ensures more efficient use of the underlying infrastructure, thereby reducing system resource demands.

3.4 Consensus Layer Approaches

The consensus layer is a crucial component primarily responsible for achieving agreement among participating nodes. As the scale of consortium blockchains continues to expand, designing a consensus protocol that ensures system consistency while improving performance has become crucial for addressing scalability. In some scenarios, failures are limited to node crashes, and thus consensus algorithms only need

to tolerate crash faults while maintaining efficiency. In other scenarios, however, consortium blockchains involve sensitive data and may face malicious or Byzantine behaviors, which place higher demands on security and consistency. Therefore, the choice of consensus algorithm for consortium blockchains must strike a balance between scalability and security. Based on these considerations, existing approaches can be broadly classified into two categories: Crash Fault Tolerant (CFT) algorithms and Byzantine Fault Tolerant (BFT) algorithms.

3.4.1 Crash Fault Tolerant Algorithms

When the scale of a consortium blockchain is larger and the trust level among nodes is higher, focusing primarily on issues related to node crashes due to failures, CFT algorithms are more appropriate. A representative CFT algorithm is Raft. Fig. 9a presents the leader-based log replication of Raft. Raft divides node roles into leader, follower, and candidate. During the consensus process, Raft selects the leader through an election mechanism. Following this, the leader node processes all write requests and replicates these requests to followers in the form of log entries to achieve global consistency. The efficiency of the leader node's log replication and election process is a core factor that restricts the scalability of the Raft algorithm. Therefore, existing CFT research is focused on optimizing Raft in two areas: leader election and log replication, to enhance scalability.

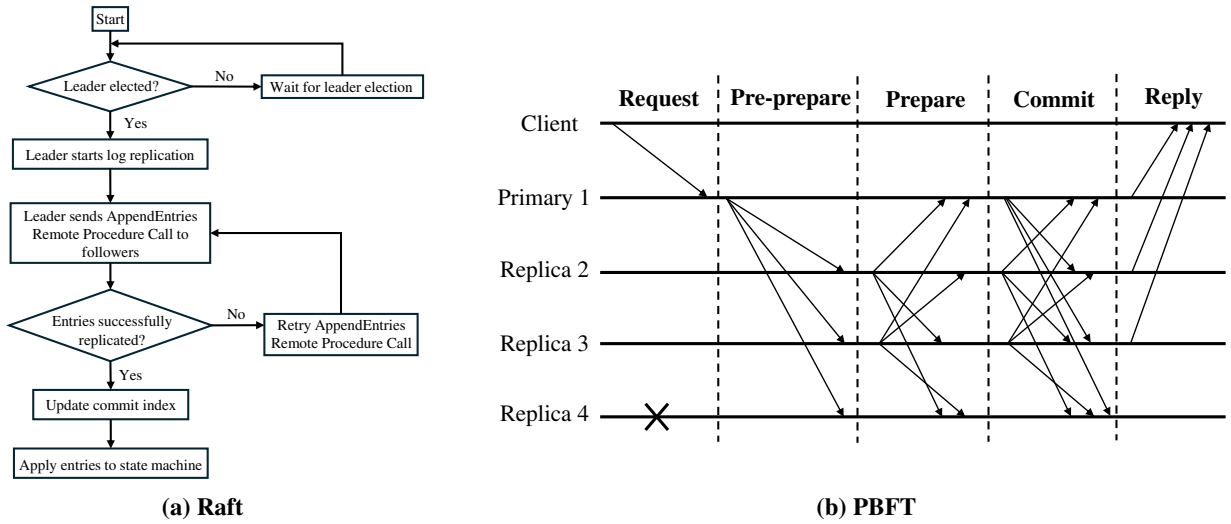

Figure 9: Schematic diagram of (**a**) Raft and (**b**) PBFT, illustrating the leader-based log replication and Byzantine consensus message flow, respectively

For the leader election optimization, the impact of elections on scalability manifests in two aspects: (1) the latency generated by the election process [87,88] and (2) the performance of the elected nodes [89–91]. The former focuses on optimizing the election process. PB-Raft [87] employs a weighted PageRank algorithm to shorten the election timeout for high-ranking nodes, thereby accelerating the convergence of the leader election process. Fu et al. [88] minimize the probability of split votes and subsequent election rounds by adjusting the affiliation of peers based on voting results, which reduces the overall latency of the election phase. The latter aspect aims to enhance the scalability of subsequent consensus processes by selecting high-performance leaders. P-Raft proposed by Lu et al. [89] ensures the elected leader has the necessary computational and network capacity to efficiently handle log replication through real-time assessment of each node's machine performance. RaftOptima introduced by Kondru and Rajiakodi [90] incorporates a

proxy leader, alleviating the leader's communication bottleneck by delegating command distribution and response collection tasks.

For the log replication optimization, existing research has proposed various methods to reduce the log replication cost for the leader node. SRaft proposed by Ye et al. [92] splits the consensus mechanism into replication and ordering phases. In SRaft, the replication phase utilizes an adaptive leaderless replication method, which distributes the network load across multiple nodes, thus mitigating the single-point bottleneck of the leader during log propagation. LRD-Raft [93] reduces the leader's bandwidth consumption and processing overhead by delegating part of the log replication tasks to follower nodes. Fu et al. [88] introduced a distribution mechanism during the log replication process, which parallelizes the log dissemination process and effectively reduces the communication complexity of the leader node.

3.4.2 Byzantine Fault Tolerant Algorithms

When the scale of a consortium blockchain is relatively small and there is a high demand for security, allowing for a certain proportion of malicious nodes, BFT algorithms [94–96] are the ideal choice. A representative BFT algorithm is Practical Byzantine Fault Tolerance (PBFT). Fig. 9b shows the Byzantine consensus message flow of PBFT. BFT algorithms ensure that the system can still reach consensus and prevent data tampering even in the presence of malicious nodes. The practical maturity of BFT algorithms for enterprise environments is confirmed by recent platform adoption. A significant validation is the integration of SmartBFT [94] as a production-ready ordering service in Hyperledger Fabric v3.0 [97]. This adoption underscores the viability of modern BFT protocols for consortiums demanding high security and resilience, moving beyond foundational algorithms like PBFT. BFT algorithms mainly include three stages: proposal, voting, and confirmation. Each stage involves significant communication and message exchanges between nodes, so these three processes affect the scalability of the consortium blockchain. To improve the scalability of BFT algorithms, optimizations can be made for these three stages. Therefore, the methods can be classified into proposal stage optimization, voting stage optimization, and confirmation stage optimization.

In terms of the proposal stage optimization, the main goals are to reduce sequential dependency and alleviate the leader bottleneck. Multi-pipeline HotStuff proposed by Cheng et al. [98] allows multiple leaders to propose blocks concurrently and utilizes pipelining to reduce idle time during the proposal phase, which improves the efficiency of the consensus process by increasing block production parallelism. Votes-as-a-Proof (VaaP) proposed by Fu et al. [12] allows nodes to propose blocks and vote concurrently, which reduces the serial dependency between the proposal and voting phases in traditional consensus protocols, thus shortening the time for a block to be confirmed.

In terms of the voting stage optimization, throughput can be improved by reducing redundant voting and mitigating the impact of malicious nodes [11,99–101]. For example, dynamically selecting high-reputation nodes for voting based on their performance and classifying nodes using the ID3 decision tree [11] or clustering mechanism [99] can reduce the communication overhead in the voting stage by shrinking the size of the consensus committee and avoid the performance degradation caused by slow or malicious nodes. TSBFT introduced by Tian et al. [100] uses the gossip protocol to disseminate voting messages and employs threshold signatures to aggregate vote results. The combination of these two mechanisms significantly reduces the communication overhead and message complexity during the voting phase.

In terms of the confirmation stage optimization, the main goal is to reduce confirmation latency [102–104]. Lyra proposed by Zarbafian and Gramoli [102] employs the commit-reveal mechanism to prevent reordering or front-running attacks, thereby reducing the overhead caused by invalid transactions; at the same time, it adopts a leaderless consensus mechanism that enables multiple nodes to advance

confirmation in parallel. Together, these two features reduce the communication burden and latency in the confirmation stage.

Key insight: Table 4 analyzes the effectiveness of the scaling approaches in consensus layer. Consensus layer approaches enhance scalability by optimizing the agreement process itself to make it faster and more efficient. Regarding the impact on TPS, both CFT and BFT algorithms primarily focus on reducing the time per round (Δt_{round}). They achieve this by accelerating leader election, streamlining data replication, and reducing the communication overhead in voting and confirmation stages. The costs associated with these optimizations are significant. The most prominent is the sharp increase in operational complexity, as optimizations for both algorithm types introduce more intricate protocol logic and management requirements. Additional system resources are consumed by cryptographic overhead and extra mechanisms like performance monitoring or signature aggregation. Furthermore, trust & control costs are elevated due to a dependency on high-performance leaders, small committees, or specialized sequencers to accelerate the process.

Table 4: The impact on TPS, cost, and performance of the scaling approaches in the consensus layer. (SR: System Resources, OC: Operational Complexity, TC: Trust & Control, DA: Data Assurance)

Scaling approaches	Ref.	Impact on TPS		Cost				Performance
		N_{tx}	Δt_{round}	SR	OC	TC	DA	
Crash fault tolerant algorithms	[87]	–	↓	▲	✓	▲	–	PB-Raft shows 8.9% higher throughput and 12.3% lower latency compared to the Raft algorithm.
	[89]	–	↓	▲	✓	▲	–	The throughput of P-Raft is improved by 14.2% compared with the Raft mechanism.
	[93]	–	↓	▲	✓	–	▲	LRD-Raft improves throughput by 34.6% and reduces latency by 25.7% compared to Raft.
Byzantine fault tolerant algorithms	[12]	↑	↓	▲	✓	▲	–	The peak throughput of VaaP is 274.5 blocks per second (BPS), while that of Sphinx is 33.6 BPS.
	[11]	–	↓	▲	✓	✓	▲	With 100 nodes, CE-PBFT improves throughput by over 5× compared to the original PBFT.
	[100]	–	↓	▲	✓	✓	–	TSBFT achieves 3.5× higher throughput compared to the PBFT protocol.

Note: (↑: Increase, ↓: Decrease, ✓: Major cost, ▲: Minor cost, –: Not significant)

However, The substantial costs in operational complexity and system resources, present an opportunity for AI-driven optimization. AI can streamline the intricate management of consensus protocols, which directly tackles the high operational complexity. This intelligent, real-time adaptation also curtails unnecessary cryptographic overhead, leading to a more efficient use of system resources.

3.5 Summary

Fig. 10 presents the distribution of scaling approaches across four layers. It illustrates the relative attention given to each layer and the specific techniques within them. Table 5 synthesizes the typical impact of approaches within these layers on performance metrics and cost dimensions. Data layer and execution layer offer the most significant potential for scalability enhancement. They provide multiplicative gains by directly expanding the system's concurrency to increase N_{tx}. The data layer, through techniques like sharding and DAG, breaks the limitations of a single, linear ledger, allowing for parallel transaction confirmations. Similarly, the execution layer, via batch processing and parallel execution, increases the number of valid transactions processed per cycle. In contrast, the consensus and network layers act more as divisive factors. Their primary role is to reduce the time per round (Δt_{round}) and efficiently realize the concurrency enabled by the other layers, ensuring that performance gains are not lost to protocol or propagation bottlenecks.

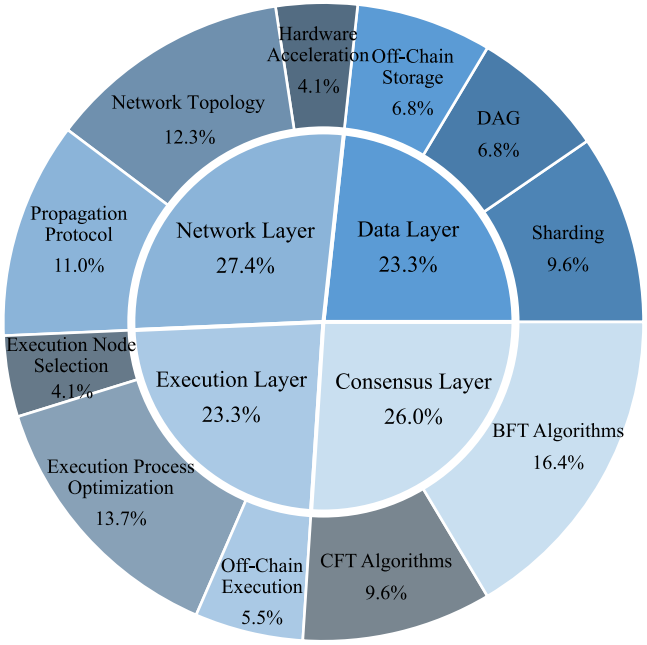

Figure 10: Distribution of research on scaling approaches for consortium blockchains. Research attention appears relatively balanced across the four layers, with slightly more emphasis on the network 27.4% and consensus 26.0% layers

Table 5: Comparative summary of scaling approaches across blockchain layers. This table compares the typical impact of scaling approaches within each layer on TPS, the associated cost profiles, and the range of performance improvements observed in the surveyed literature relative to baseline systems

Layer	Impact on TPS		Cost				Performance improvement compared to the baseline
	N_{tx}	Δt_{round}	SR	OC	TC	DA	
Data	↑	↓	✓	✓	▲	✓	TPS improved by 28.57%–1900%.
Network	–	↓	✓	✓	▲	–	TPS improved by 12.97%–230%. Latency reduced by 20.2%–99.95%.

(Continued)

Table 5 (continued)

Layer	Impact on TPS		Cost				Performance improvement compared to the baseline
	N_{tx}	Δt_{round}	SR	OC	TC	DA	
Execution	↑	↓	✓	✓	▲	▲	TPS improved by 40%–3900%. Latency reduced by 30%–87%.
Consensus	–	↓	▲	✓	✓	–	TPS improved by 8.9%–717%. Latency reduced by 12.3%–33%.

Note: (↑: Increase, ↓: Decrease, ✓: Major cost, ▲: Minor cost, –: Not significant)

However, as shown in Table 5, these performance gains come with significant trade-offs, with each layer presenting a distinct cost profile. 1) In the data layer, complex cross-domain protocols and verification increase data assurance; new data structures raise operational complexity and system resources; and dependency on special nodes introduces trust & control. 2) The network layer's costs arise as specialized hardware and maintenance increase system resources and operational complexity, while centralized controllers or backbone nodes increase trust & control. 3) In the execution layer, the sophisticated logic required for scheduling and parallel processing raises operational complexity; the introduction of off-chain components and verifiable proofs increases system resources and major data assurance costs; and the dependency on centralized schedulers or specialized nodes introduces trust & control risks. 4) Consensus layer sees increased operational complexity from complex protocol pipelines, higher system resources from cryptographic overhead, and elevated trust & control from a dependency on small committees or sequencers. Overall, the costs of system resources, operational complexity, and trust & control are present in all four layers. In addition, there are also data assurance costs in the data layer and the execution layer.

4 AI-Enabled Methods

As established in the previous section, scaling approaches of consortium blockchains introduce significant costs across four key dimensions. AI-enabled methods offer a new perspective. By leveraging data-driven learning and real-time adaptation, they seek to discover more efficient system configurations and operational policies, thereby pushing scalability boundaries while reducing the associated costs. This section provides a systematic review of these AI-enabled methods, exploring how various methodologies are applied across the Data, Network, Execution, and Consensus layers to achieve this dual objective.

4.1 Data Layer Methods

Section 3.1 introduces three primary scaling approaches for the data layer: DAG, sharding, and off-chain storage. Among them, sharding is the only category where AI-enabled enhancements have been actively explored. Sharding technology faces two core challenges: first, how to optimize the static partitioning of accounts or data to reduce cross-shard interactions; and second, how to dynamically adjust shard parameters to adapt to changing system loads. These two challenges naturally align with the characteristics of different AI methods. For the former, unsupervised learning excels at discovering underlying community structures from transaction data to guide shard partitioning. For the latter, reinforcement learning is adept at continuous decision-making and adaptive control in dynamic environments. Furthermore, to address specific issues like

node heterogeneity and unreliability, Other AI methods are used to assess node trustworthiness to ensure the security and integrity of data within shards. Consequently, AI methods in sharding are categorized as follows: reinforcement learning, unsupervised learning, and other AI methods (Table 6).

Table 6: The impact on cost and performance of the AI methods in the data layer. (SR: System Resources, OC: Operational Complexity, TC: Trust & Control, DA: Data Assurance)

AI methods	Ref.	Scaling approach opti-mized	Impact on cost				Performance
			SR	OC	TC	DA	
Unsupervised learning	[16]	Sharding	↓	↓	–	–	Boosts throughput by up to 2.68× and cuts latency by 88% over comparable protocols.
	[105]	Sharding	↓	↓	↑	↓	Cuts block delay by approximately 74% and improves throughput over Rapidchain.
	[106]	Sharding	↓	↓	–	–	More than doubles transaction throughput and cuts latency by about 76% vs. OmniLedger.
Reinforcement learning	[107]	Sharding	↓	↓	–	↓	Boosts throughput by 50.5% over Monoxide while also outperforming Brockerchain.
	[108]	Sharding	↓	↓	–	↓	Reduces training time by 70% while achieving an 8.3× higher system reward over baselines.
	[109]	Sharding	↓	↓	–	↓	Improves throughput by 33% over existing consortium blockchains.
Other AI Methods	[110]	Sharding	↓	↓	↑	↓	Delivers a about 3000 TPS throughput advantage and halves confirmation latency compared to Rapidchain.

Note: (↑: Increase, ↓: Decrease, –: Not significant)

4.1.1 Unsupervised Learning

Unsupervised learning methods, particularly clustering, graph methods, and probabilistic modeling, are highly suitable for optimizing the static partitioning of accounts in sharding. Their core strength lies in

uncovering hidden patterns and relationships within transaction data to identify communities of accounts. By partitioning closely related accounts into the same shard, these methods maximize intra-shard transaction locality. This locality reduces the reliance on high-latency, low-concurrency cross-shard communication protocols, thereby maximizing the system's parallel processing capability.

Various unsupervised models have been applied in this domain [16,105,106,109]. DBSRP-ML [16] proposes a label graph network to represent account states and utilizes a multi-label community detection algorithm for account partitioning. By ensuring that the vast majority of transactions can be processed entirely within a single shard, this method maximizes the system's parallel processing capability. MSLShard [105] considers network distance, node credibility, and access frequency, using a spectral clustering algorithm to partition nodes. By creating an optimized topology where nodes within a shard have low-latency connections, this method accelerates intra-shard consensus and transaction processing. HMMDShard [106] takes a different method by constructing a dynamic transaction graph and employing a Hidden Markov Model (HMM) for dynamic, fine-grained incremental sharding. By proactively adapting the data partitions to predicted transaction flows, this method allows the system to sustain a high degree of parallel execution.

4.1.2 Reinforcement Learning

Reinforcement learning methods are highly suitable for the dynamic adaptive control of a sharded architecture, especially in dynamic environments where transaction loads and network conditions are constantly changing [111,112]. Their core advantage is the ability to continuously adjust sharding parameters based on real-time feedback. This dynamic adjustment is crucial for increasing TPS as it prevents individual shards from becoming overloaded due to load variations, thereby ensuring the entire system can consistently engage in efficient parallel processing.

Several works highlight this direction [106–108,113]. DSSBD [107] models the optimal block generation problem as a markov decision process (MDP) and uses deep reinforcement learning to dynamically adjust the number of shards, block interval, and block size. By optimizing these parameters in real time, this method ensures that each shard can process transactions at maximum efficiency, avoiding backlogs caused by parameter mismatches. Similarly, DSPO-CB [106] also employs a Deep Q-Network (DQN) to select optimal sharding and consensus parameters. This dynamic enhancement avoids the performance bottlenecks that static configurations can create under changing loads, ensuring the sharded system operates efficiently. Fig. 11 illustrates the deep reinforcement learning (DRL) framework, where an agent learns to optimize the system by observing states and taking actions across all four layers.

4.1.3 Other AI Methods

Beyond the primary learning paradigms, other AI Methods, such as trust models from probabilistic logic, are used to address specific challenges in sharding [109,110,114]. The core of these methods is to model the heterogeneity and reliability of nodes quantitatively. A stable and reliable shard is a prerequisite for continuous and efficient parallel processing.

MSLTChain [110] is a representative study. It introduces a trust model based on multi-dimensional subjective logic within a tree sharding structure. This model does not directly optimize data partitioning or dynamic parameters but rather guides shard composition and maintenance by assessing and filtering nodes based on their credibility. By dynamically filtering out low-reputation nodes, the system ensures the stable operation of each shard, avoiding processing interruptions caused by node failures or malicious activities, thereby providing the foundation for sustained high TPS.

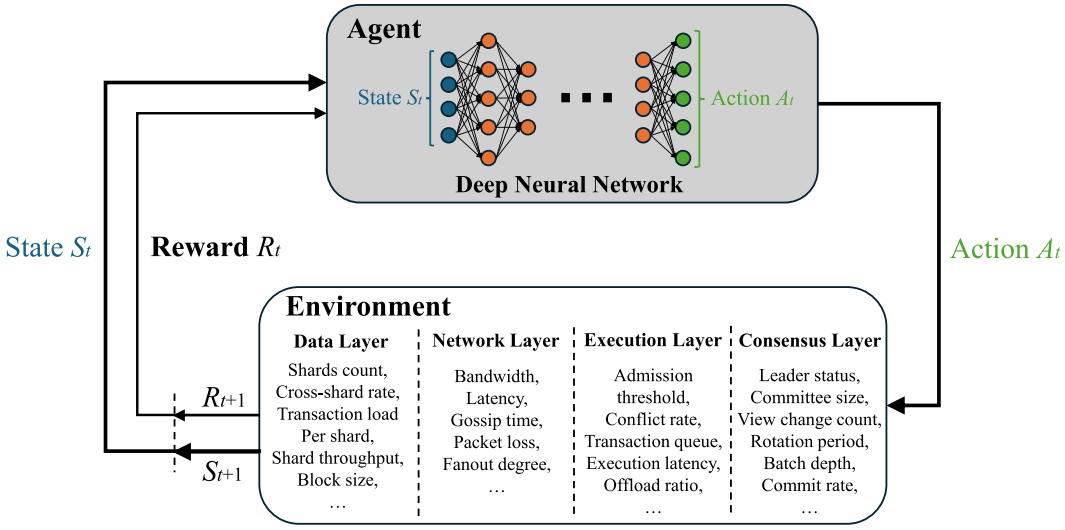

Figure 11: An overall framework illustrating how a deep reinforcement learning (DRL) optimizes the scalability of the four layers. The agent observes a comprehensive state from all four layers and takes actions to dynamically tune system parameters, creating a feedback loop (state → action → reward)

4.2 Network Layer Methods

Building upon the classification outlined in Section 3.2, the approaches of the network layer are primarily divided into three main types: hardware acceleration, network topology, and propagation protocol. Currently, AI methods have not yet been significantly applied in the hardware acceleration domain. However, AI has emerged as a powerful tool to elevate the enhancement of the other two categories. On one hand, for tasks involving the static structural design of the network, supervised learning can be used to predict optimal node placements, while unsupervised learning is excellent at discovering natural, efficient topologies and propagation paths from network data. On the other hand, for tasks that require dynamic policy control and real-time adjustment of network parameters, the adaptive, feedback-driven nature of reinforcement learning is an ideal fit. Consequently, this section categorizes AI methods in the network layer into three main types: reinforcement learning, supervised learning, and unsupervised learning (Table 7).

Table 7: The impact on cost and performance of the AI methods in the network layer. (SR: System Resources, OC: Operational Complexity, TC: Trust & Control, DA: Data Assurance)

AI methods	Ref.	Scaling approach optimized	Impact on cost				Performance
			SR	OC	TC	DA	
Supervised learning	[115]	Network topology	↓	↓	↑	–	Achieves an approximately 2.39× better TPS guarantee over random placement strategies.

(Continued)

Table 7 (continued)

AI methods	Ref.	Scaling approach optimized	Impact on cost				Performance
			SR	OC	TC	DA	
Unsupervised learning	[116]	Propagation protocol	↓	–	–	–	Reduces block propagation latency by 14%–19% over the original gossip implementation.
	[117]	Propagation protocol	↓	↓	↑	↓	Achieves the lowest block propagation time against genetic algorithm and graph attention network baselines.
Reinforcement learning	[118]	Network topology	↓	↓	–	–	Outperforms static schemes by consistently delivering higher throughput and lower latency.
	[14]	Network topology	↓	↓	–	–	Boosts throughput by 470.45% over default settings while also outperforming advanced tuners.
	[119]	Network topology	↓	↓	–	–	Adapts to new network bandwidths with about 72% fewer samples than non-adaptive reinforcement learning methods.

Note: (↑: Increase, ↓: Decrease, –: Not significant)

4.2.1 Supervised Learning

Supervised learning are employed to optimize the structural design of the network topology before deployment [115,120–122]. By training a model on data that maps network configurations to performance outcomes, these predictive methods can determine the optimal placement of nodes in a geo-distributed environment. This data-driven foresight helps in constructing a network topology that is inherently efficient, thereby improving scalability from the ground up.

A representative study by Lee et al. [115] demonstrates this method. They build a TPS predictor by comparing various supervised models, including random forest, gradient boosting decision tree (GBDT), deep neural network (DNN), and support vector machine (SVM). This predictor is used to select the optimal node placement strategy before the system is launched, which ensures an efficient initial network topology and guarantees high-speed data dissemination pathways from the start.

4.2.2 Unsupervised Learning

Unsupervised learning is highly effective for optimizing both the network topology and the propagation protocol by discovering efficient, underlying structures from network data without predefined labels. Clustering methods can partition nodes into geographically or logically proximal groups to optimize the topology, while graph-based and generative methods can learn the most efficient data broadcast paths to improve the propagation protocol.

Several studies exemplify these methods [116,117]. To optimize the propagation protocol, Xu et al. [116] introduce density clustering in Hyperledger Fabric's broadcast mechanism. This method constructs a highly dense and connected topology, which reduces redundant message transmissions and lowers overall block propagation latency. In a more advanced application, Kang et al. [117] use a Graph Refusion model, which combines a graph neural network with a diffusion model, to generate the optimal propagation trajectory. This generative method learns complex spatial relationships within the network to devise the most efficient path for block dissemination, minimizing latency.

4.2.3 Reinforcement Learning

Reinforcement learning methods are particularly well-suited for the dynamic policy control aspect of network topology and propagation protocol optimization. In environments with fluctuating conditions like network bandwidth, a reinforcement learning agent can learn an optimal policy to continuously adjust system parameters through interaction and feedback. This enables the network to self-optimize and maintain high message propagation rates, a key factor for scalability.

Several DRL methods have been proposed to achieve this [14,118,119]. The work by Liu et al. [118] uses DRL to dynamically select and adjust parameters such as block producers, block size, and interval. This allows the system to adapt the network's data traffic and load in real time, preventing propagation bottlenecks. Athena, proposed by Li et al. [14], introduces a multi-agent DRL algorithm for adaptive parameter tuning in Hyperledger Fabric, which improves the overall message dissemination speed across the network. Going a step further, Pei et al. [119] use meta-reinforcement learning (Meta-RL) to create a tuning agent that can quickly adapt to unknown network bandwidth changes. This method minimizes performance degradation due to network fluctuations and maintains high propagation rates.

4.3 Execution Layer Methods

Execution layer scaling approaches mainly include three categories in Section 3.3: execution node selection, execution process optimization, and off-chain execution. AI has been applied to all three areas to enhance scalability. The suitability of each AI method corresponds to the specific challenges of these tasks. For dynamic control problems like execution process optimization and off-chain execution, the capacity of reinforcement learning to learn optimal decision-making policies through continuous interaction with the environment makes it an ideal fit. For execution process optimization, unsupervised learning can identify and filter anomalous transactions without explicit labels. Finally, for complex execution node selection problems that require balancing multiple real-time metrics, Other AI methods like computational intelligence are highly suitable. Consequently, AI methods in the execution layer can be summarized into three categories: reinforcement learning, unsupervised learning, and other AI methods (Table 8).

Table 8: The impact on cost and performance of the AI methods in the execution layer. (SR: System Resources, OC: Operational Complexity, TC: Trust & Control, DA: Data Assurance)

AI methods	Ref.	Scaling approach optimized	Impact on cost				Performance
			SR	OC	TC	DA	
Unsupervised learning	[123]	Execution process optimization	↓	–	–	↑	Achieves a 47.81 compression ratio for transactions while detecting and rejecting anomalous data.
	[124]	Execution process optimization	↓	↓	–	–	Commits 17% more transactions than the best fixed architecture under dynamic workloads.
Reinforcement learning	[125]	Execution process optimization	↓	↓	↑	↓	Cuts time-critical transaction latency by 10×, reaching 10ms, while maintaining over 99% reliability.
	[126]	Off-chain execution	↓	↓	↓	↓	Reduces task latency up to 15% and energy consumption by 12% over existing strategies.
	[127]	Off-chain execution	↓	↓	–	–	Reduces total system cost by up to 22.2% against local or edge-only execution strategies.
Other AI methods	[128]	Execution node selection	↓	↓	↑	–	Boosts average throughput by 19% and lowers system latency by 17% vs. default endorsement.
	[129]	Execution node selection	↓	↓	↑	–	Improves average throughput by over 41% and reduces response latency by over 30%.

Note: (↑: Increase, ↓: Decrease, –: Not significant)

4.3.1 Unsupervised Learning

Unsupervised learning methods contribute to execution process optimization by identifying patterns, such as anomalies, from unlabeled data. This capability is particularly valuable for pre-filtering transactions before they enter the resource-intensive execution and endorsement stages, thereby reducing the workload on the execution nodes.

A DNN-based contract policy proposed by Sapkota et al. [123] exemplifies this method. It uses a Long Short-Term Memory (LSTM) autoencoder, a type of unsupervised model, to identify anomalous Internet of Things (IoT) transactions based on reconstruction error. By identifying and filtering these anomalous transactions before they enter the execution pipeline, the framework reduces the computational load on endorsing peers, resulting in a faster and more efficient execution process.

4.3.2 Reinforcement Learning

Reinforcement learning methods are highly versatile for enhancing execution layer scalability, addressing challenges in both on-chain process optimization and off-chain execution. By continuously interacting with the system to learn dynamic strategies, reinforcement learning can adjust key parameters in real time. For on-chain optimization, it excels at adapting transaction scheduling and execution parameters. In off-chain execution, it is particularly effective for managing complex task offloading decisions in dynamic environments.

For on-chain execution process optimization [124,125,130,131], AdaChain introduced by Wu et al. [124] models the selection of blockchain architecture and parameters as a contextual multi-armed bandit problem. By learning to choose the best execution configuration for the current workload, it ensures the system operates with optimal efficiency, avoiding processing bottlenecks. PBRL-TChain designed by Zhang et al. [125] introduces deep reinforcement learning to continuously optimize key aspects such as transaction admission and priority scheduling. This optimization of the transaction pipeline reduces execution conflicts and re-processing overhead.

For off-chain execution [126,127,132–134], Li et al. [126] employ Independent Deep Q-Networks to intelligently decide where to execute tasks based on security and resource constraints. This efficient management of off-chain tasks reduces the computational burden on the blockchain network, allowing it to scale more effectively for security-critical operations. Nguyen et al. [132] employ Multi-Agent Deep Deterministic Policy Gradient (MADDPG), a cooperative reinforcement learning approach illustrated in Fig. 12, to jointly optimize task offloading and block mining decisions. By finding optimal offloading strategies, this method frees up on-chain execution resources, allowing the blockchain to process other transactions with lower latency and higher concurrency.

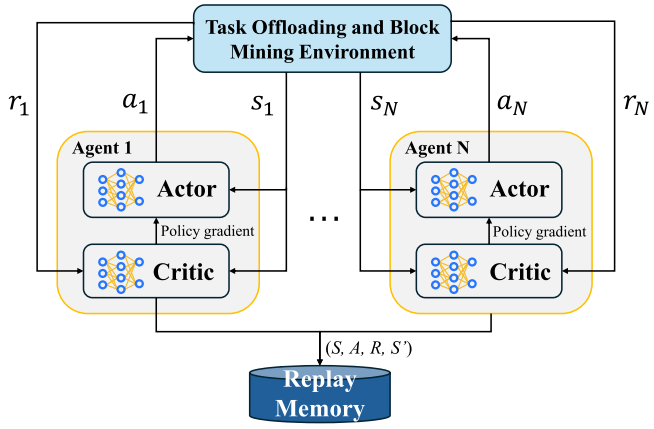

Figure 12: Simplified architecture of the MADDPG algorithm. Agents interact with the environment, storing experiences in a shared Replay Memory. Each agent uses an Actor and a Critic network, with the Critic guiding the Actor's updates via policy gradients learned from sampled experiences

4.3.3 Other AI Methods

Beyond traditional machine learning paradigms, other AI methods, such as computational intelligence methods, are used for execution node selection. These methods excel at multi-criteria decision-making without requiring a model to be trained on historical data. They can dynamically evaluate and rank endorsement nodes based on multiple real-time resource metrics, ensuring a balanced workload distribution across execution nodes.

Two studies highlight this direction. Dra-Fabric, introduced by Wu et al. [128], uses the multi-criteria decision making (MCDM)-based methods to rank candidate endorsement nodes based on their real-time computational, storage, and network resource utilization. This intelligent allocation avoids overloading individual nodes and reduces endorsement latency. Similarly, Pdo-Fabric, proposed by Yu et al. [129], uses fuzzy logic to evaluate the resource status of endorsement nodes. This evaluation then guides a scheduling algorithm to dynamically distribute transaction proposals across different organizations, which improves resource utilization during the endorsement phase and leads to faster parallel execution of transaction proposals.

In addition, a prominent production-level example demonstrating the principle of off-chain computation with on-chain governance is the Ritual Infernet network [135]. This system operationalizes the off-chain execution model for the specific task of AI inference. In this architecture, on-chain smart contracts submit inference requests to the decentralized off-chain network, which performs the computation and returns the result. This model directly aligns with the off-chain execution principle of outsourcing intensive tasks while retaining on-chain governance, where the smart contract defines the request and verifies the outcome provided by the decentralized network.

4.4 Consensus Layer Methods

Section 3.4 categorizes the scaling approaches in the consensus layer of consortium blockchains into two categories: CFT and BFT. AI-driven enhancements have been applied to both categories. CFT protocols such as Raft focus on tolerating benign failures, and their performance bottlenecks in leader election and log replication can often be modeled as classification or regression problems, making them an excellent fit for supervised learning. In contrast, BFT protocols such as PBFT must maintain performance in complex, dynamic environments with potentially malicious actors. This process requires strategies that can be continuously and adaptively optimized. Reinforcement learning methods are an excellent fit for such a task. Consequently, AI methods in the consensus layer can be grouped into reinforcement learning and supervised learning (Table 9).

Table 9: The impact on cost and performance of the AI methods in the consensus layer. (SR: System Resources, OC: Operational Complexity, TC: Trust & Control, DA: Data Assurance)

AI methods	Ref.	Scaling approach optimized	Impact on cost				Performance
			SR	OC	TC	DA	
Supervised learning	[136]	Crash fault tolerant algorithms	↓	↓	↑	–	Machine learning classifiers distinguish between node and link failures with up to 98% accuracy.

(Continued)

Table 9 (continued)

AI methods	Ref.	Scaling approach optimized	Impact on cost				Performance
			SR	OC	TC	DA	
	[137]	Crash fault tolerant algorithms	↓	↓	↑	–	Reduces consensus messages by 45% and improves TPS stability over the traditional Raft algorithm.
	[138]	Byzantine fault tolerant algorithms	↓	↓	–	–	Outperforms Raft and HotStuff at large scale, achieving about 2× higher throughput with 200 nodes.
	[139]	Byzantine fault tolerant algorithms	↓	↓	–	–	Boosts average throughput by over 41% and cuts response latency by over 30% vs. the default endorsement strategy.
Reinforcement learning	[15]	Byzantine fault tolerant algorithms	↓	↓	↑	–	Achieves up to 3× higher throughput and significantly lower latency over the classical PBFT protocol.
	[140]	Byzantine fault tolerant algorithms	↓	↓	–	–	Achieves higher throughput and success rates while significantly reducing message complexity over traditional PBFT.
	[141]	Byzantine fault tolerant algorithms	↓	↓	↑	–	Improves throughput by 1.6× and reduces latency by 25% over a non-learning-based scheme.

Note: (↑: Increase, ↓: Decrease, –: Not significant)

4.4.1 Supervised Learning

Supervised learning is well-suited for consensus protocols where performance can be improved by making data-driven predictions that reduce operational overhead. For BFT protocols, these methods can be used to build reputation models that evaluate node performance based on historical data, enabling the selection of more efficient committee members. For CFT protocols, they can classify system events to avoid unnecessary protocol actions or predict optimal parameters to improve processes like log replication.

For CFT optimization, Choumas and Korakis [136] use supervised classifiers to determine whether a leader election in Raft is triggered by a genuine node failure or a temporary link failure. By preventing unnecessary leadership transitions, this method enhances the operational stability and availability of the Raft protocol. In another CFT work, the Cell-Based Raft algorithm [137] uses a federated learning paradigm to

determine the optimal transaction batch size. As shown in Fig. 13, by predicting the best configuration, this method maximizes the efficiency of the log replication phase in their Raft-based algorithm.

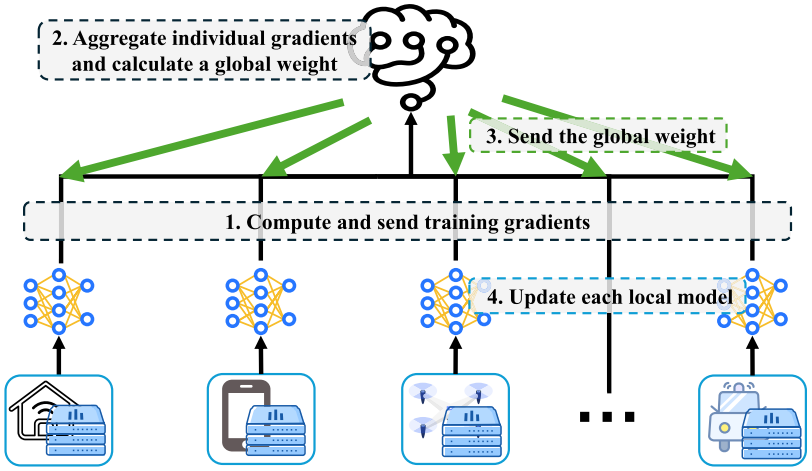

Figure 13: Basic architecture and process of federated learning, employed by Cell-Based Raft to determine optimal cell size. Consensus nodes (clients, bottom) compute local gradients based on performance with different cell sizes and send them to a central server (cloud, top). The server aggregates gradients and sends back the global model update. Nodes then update their local models to predict the optimal cell size for maximizing throughput

For BFT optimization [138,139,142–144], Deng et al. [138] employ a backpropagation neural network to build a dynamic reputation model for nodes. By ensuring the consensus committee is composed of high-performance nodes, this method accelerates the multi-stage validation process inherent in BFT algorithms. The work by Riahi et al. [139] uses multi-task learning to simultaneously classify multiple node attributes, such as honesty and availability. This allows for a more holistic selection of committee members, which enhances the efficiency of the PBFT voting stage.

4.4.2 Reinforcement Learning

Reinforcement learning methods are particularly effective for optimizing complex BFT protocols, which often operate in dynamic environments. BFT protocols involve multiple configurable parameters, and finding the optimal settings becomes challenging when transaction loads and network conditions constantly change. Through direct interaction with the system and performance-based rewards, a reinforcement learning agent can learn an optimal policy for dynamically adjusting these key parameters, allowing the BFT protocol to maintain high performance in fluctuating workloads.

Recent works demonstrate this with both modern and classic reinforcement learning methods [15,140,141,145]. Li et al. [15] employ Double Dueling DQN to build a PBFT enhancement model. This continuous adaptation ensures the PBFT protocol's parameters are always optimized for the current network conditions, resulting in more efficient consensus rounds. Similarly, Qiu et al. [140] model view switching, access selection, and resource allocation as an MDP. Their Dueling DQN agent learns an optimal strategy that improves the efficiency of leader selection and resource allocation within the BFT process. Representing a classic distributed reinforcement learning paradigm, Ameri and Meybodi [141] use a Learning Automata and Goore Game method. This game-theoretic model streamlines the block verification process, aiming to achieve faster convergence to a consensus decision than traditional BFT protocols.

4.5 Summary

Fig. 14 maps the application of AI methods to scaling approaches across the four architectural layers. AI-enabled methods have significantly improved the performance of scaling approaches. Table 10 provides a comprehensive overview of these AI-enabled methods, categorized by AI type and the specific blockchain layer they target. For each combination, the table highlights representative techniques and outlines the core enhancement strategy. Reinforcement learning has proven to be the most impactful, providing dynamic, adaptive control for sharding locality and parameter adaptation in the data layer, online tuning of network topologies and propagation paths in the network layer, adaptive admission, scheduling, and task offloading in the execution layer, and self-adjusting consensus mechanisms such as leader rotation, view change timing, and batching depth in the consensus layer. Complementing this, supervised and unsupervised learning play crucial roles in more predictive or static tasks, such as consensus layer event discrimination and reputation modeling, and pre-filtering anomalous transactions in the execution layer.

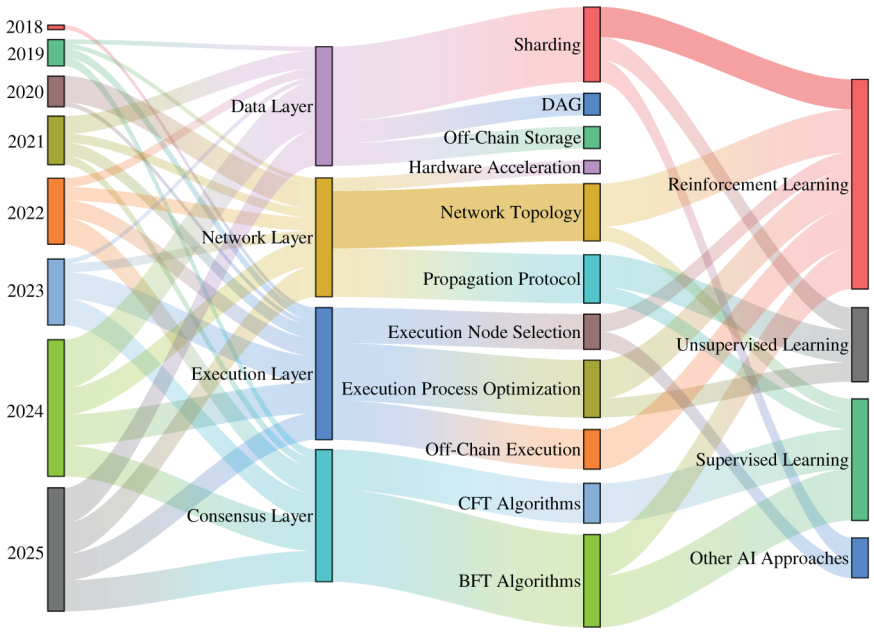

Figure 14: A Sankey diagram illustrating the evolution and interconnection of research from 2018 to 2025. It maps AI methodologies (right column) to specific scaling approaches within the four architectural layers (middle column) over time (left column). The width of the flows indicates the relative research focus connecting these areas

Table 10: comparative analysis of AI methods for scalability enhancement

AI methods	Layer	Technique	Enhancement strategy
Unsupervised learning	Data	Multi-label community detection; HMM	•Increase intra-shard locality to reduce cross-shard traffic; •Dynamic re-sharding to maintain load balance.

(Continued)

Table 10 (continued)

AI methods	Layer	Technique	Enhancement strategy
	Network	Density-based clustering; Graph refusion/diffusion	•Backbone construction via clustering to cut redundant broadcasts; •Path optimization via graph refusion/diffusion to reduce propagation delay.
	Execution	DNN/LSTM autoencoder	•Filter and compress anomalous transactions to reduce execution load.
Supervised learning	Network	Regression/ classification predictors; Graph Attention Network	•Placement prediction for nodes/replicas to shorten propagation paths; •Attention routing to lower information latency.
	Consensus	Event classifier; Backpropagation neural network; Multi-task learning	•Failure vs. network-jitter classification to avoid unnecessary leader changes; •Batching and log-parameter prediction to reduce messages; •Segmented-DAG and parameter tuning to shorten confirmation path.
Reinforcement learning	Data	DRL; Proximal Policy Optimization; DQN	•Self-tune sharding to balance load; •Hotspot-aware shard resizing to avoid overload.
	Network	Multi-agent DRL; DRL; Meta-RL	•Self-tune gossip/endorsement concurrency to avoid congestion; •Adaptive neighbor selection to speed up dissemination; •Bandwidth shifts to sustain propagation rate.
	Execution	Contextual bandit; MADDPG; Reinforcement learning; DQN	•Online configuration selection to remove execution bottlenecks; •Joint offloading-caching decisions to lower latency and energy; •Adaptive load balancing to relieve hotspots.

(Continued)

Table 10 (continued)

AI methods	Layer	Technique	Enhancement strategy
	Consensus	Dueling DQN; Actor-Critic; Learning automata/goore game	•Adaptive view-switching and batch-depth control to accelerate rounds; •Consensus-based resource and leader allocation to improve efficiency; •Game-theoretic voting to speed up convergence.
Other AI methods	Data	Subjective logic; Stackelberg game	•Credibility evolution and game-theoretic decisions to optimize leader selection
	Execution	MCDM-based; Fuzzy logic	•Endorser ranking to shorten queues; •Proposal distribution optimization to increase parallelism.

While our analysis is organized by layer, many AI models operate across these boundaries, introducing two cross-layer challenges. On one hand, data from different layers and organizations is often heterogeneous, and its patterns can change over time, making it difficult for AI models to perform consistently. On the other hand, the black-box of many AI models poses a transparency problem, which can hinder the auditability and governance essential for blockchain systems. These challenges motivate privacy-preserving coordination (e.g., federated learning) and transparency mechanisms (e.g., explainable AI) as complementary enablers for cross-layer enhancement.

Despite the optimizations provided by AI, significant costs persist across all layers. AI methods are highly effective at reducing system resources and operational complexity, but the cost of trust & control often increases, and the data assurance costs in the data and execution layers are not fully mitigated. The root cause is that AI's core advantage lies in transforming optimization problems that previously relied on manual and static configuration into dynamic and adaptive policies based on real-time observation. This shift significantly reduces resource waste from wrong configurations and automates the operational burden, thereby generally lowering system resources and operational complexity. However, the AI model concentrates the power of making decisions and system responsibilities onto a few intelligent nodes. This intensifies the dependency on a minority of components, leading to an increase in trust & control costs. In addition, the data assurance costs in the data and execution layers stem from the structural data separation introduced by architectures like sharding and off-chain execution, which must be underwritten by cryptographic proofs and protocol-level rollback mechanisms. While AI can reduce the frequency at which these mechanisms are triggered, it cannot eliminate the fundamental need for them, and thus, the associated data assurance costs remain significant.

Finally, the integration of AI itself is not without costs, creating new trade-offs that mirror the four dimensions of concern. AI models can increase system resources due to computational demands for training and inference, and add new components that raise operational complexity. Furthermore, a complex AI model can become a new centralized point of failure, introducing trust & control risks, while its black-box nature may pose challenges to the auditability required for data assurance.

5 Opportunities

Although AI-enabled methods have made significant strides in mitigating the inherent costs of scaling consortium blockchains, there remains considerable room for enhancement. The remaining costs and the new trade-offs introduced by AI integration represent key opportunities for future research. This chapter delves into these opportunities. First, we analyze how AI, acting as an optimizer and coordinator, can further reduce the persistent Trust & Control costs across the four layers. We then provide a systematic analysis of the new trade-offs that arise from integrating AI itself, proposing corresponding mitigation strategies for the four cost dimensions.

5.1 Addressing Remaining Costs: Trust & Control

In Section 4.5, we discuss the remaining trust & control and data assurance costs after the optimizations of AI-enabled methods. Although AI offers powerful optimization tools, it is essential to recognize that several costs in blockchain scalability are rooted in domains where AI is not the primary method. For instance, to reduce the high data assurance costs in the data layer and lower system resource costs in the network layer, it requires designing lightweight forensic proofs and managing offline cryptographic material pre-generation. These depend on cryptographic innovation, not AI models. Similarly, to reduce data assurance in the execution layer, it is necessary to guarantee formally verifiable conflict rules, which is a matter of strict protocol and semantic design. For the trust & control cost, AI's role in the blockchain can act as an optimizer and coordinator. The key research opportunities in each layer where AI can fulfill this role are as follows.

Data Layer. The Trust & Control cost in the data layer primarily originates from the introduction of critical central nodes like sorters and aggregators, which are necessary for cross-shard or off-chain interactions. These nodes control the entire data channel, leading to a concentration of power. A key optimization opportunity lies in decoupling data availability from transaction ordering, which disperses the control over data processing across multiple stages, including encoding/distribution, consensus on digests, and subsequent forensics. As a result, the power of any single role is significantly diminished. The core idea is to first reach consensus on data digests and then pull the full data in parallel. In this model, the sorter/aggregator no longer controls the complete data flow, and retrieval/forensics are parallelized into later stages, thus reducing the bottleneck risk of central nodes. Research in this area can draw inspiration from DispersedLedger [146], which uses verifiable information dispersal to achieve near-optimal communication efficiency and significantly alleviates the centralization pressure caused by slow nodes.

Network Layer. The Trust & Control cost in the network layer mainly comes from the introduction of centralized control planes like backbone relay networks or SDN to improve propagation efficiency, leading to a structural dependency on a few key components. An important optimization opportunity is to build multi-active relay networks with fast failover capabilities. This method avoids binding all traffic to a single trusted relay and downgrades the decision-making power of the control plane to an advisory role, supplemented by automated fallback mechanisms. This reduces the structural dependency on a minority of nodes. Specifically, this can be achieved by deploying multiple parallel relay links that can replace each other at any time. The AI control plane only provides routing suggestions rather than enforcing them. The actual switching is triggered automatically by verifiable evidence and is recorded in auditable logs. Inspiration for this can be drawn from the design of Dumbo-NG [101], whose pipelining of throughput and latency optimization and fault tolerance can inform the threshold settings for an evidence-based switching mechanism.

Execution Layer. The Trust & Control cost of the execution layer is particularly evident in off-chain execution scenarios, where both the execution of business logic and the adjudication of results are outsourced to a few off-chain nodes. A core optimization opportunity lies in implementing verifiable Service

Level Objectives (SLOs) for off-chain execution, which allows for outsourcing execution while retaining governance. This method keeps the governance layer on-chain, while the business acceleration happens off-chain. This avoids handing over both execution and adjudication powers entirely to a few off-chain outsourcers. The specific method is to have the off-chain execution responsible only for providing verifiable computational results under the constraint of being punishable for violations. The on-chain layer retains a complete fallback path, including SLO monitoring, timeout-based rollbacks, and re-execution if necessary. In this model, AI is only used for breach prediction or workload forecasting, not for final judgment. This model, where AI acts as a predictor across different organizational boundaries, naturally calls for privacy-preserving learning techniques. A promising future direction is the use of federated reinforcement learning [147], which would allow multiple consortium members to collaboratively train a sophisticated policy for task offloading or resource scheduling without sharing their sensitive, local transaction data. This federated approach maintains the decentralized ethos of the consortium while enabling the adaptive optimization required to manage these complex execution strategies.

Consensus Layer. The Trust & Control cost in the consensus layer is primarily due to shortcuts taken to accelerate, such as small committees, fixed leaders, or sorters, which inevitably lead to dependency concentration. A key optimization opportunity is to dynamically and decentrally manage these key roles through random sampling and high-frequency rotation. This method breaks down long-term critical responsibilities into short-term and replaceable roles. It ensures accountability without relying on personal reputation through auditable logs and threshold signature mechanisms, thus weakening the risk of long-term power consolidation. This direction can be implemented by using Verifiable Random Functions for the random sampling of committee members and setting short tenures to achieve high-frequency rotation. The system must also be equipped with auditable voting logs and threshold signature mechanisms to ensure transparency and security. Within this framework, AI can provide dynamic recommendations for parameters like rotation rhythm and committee size, but the final decisions must be constrained by the protocol's safety-critical upper and lower bounds.

5.2 Analyzing AI Integration Trade-Offs

Although AI brings new perspectives and methods for scaling consortium blockchains, its introduction is not without cost; rather, it introduces new trade-offs that correspond to the four cost dimensions we used in the previous analysis of scaling approaches. We elaborate on these trade-offs, detailing the costs brought by the integration of AI and proposing corresponding mitigation strategies for each, to ensure that while leveraging AI to enhance scalability, its potential negative impacts can be effectively managed.

System Resources. AI models, particularly for training and online inference, introduce significant computational overhead, increasing system resources demands [148]. Mitigation strategies include developing lightweight, specialized models through techniques like pruning and distillation, offloading model training to non-critical environments, and setting strict resource budgets and timeouts for online inference, with clear fallbacks to simpler heuristics.

Operational Complexity. The integration of AI adds a full machine learning operations lifecycle, elevating operational complexity. This includes managing feature extraction, model versioning, performance monitoring, and detecting data or concept drift. To manage this, robust, unified observability platforms are required to monitor system, business, and model metrics simultaneously. Automated fail-safes, such as circuit breakers that revert to static policies upon detecting anomalous AI behavior, and canary deployments, are essential for safe management.

Trust & Control. An AI model can become a new centralized policy control plane, creating trust & control cost. The entity that trains or controls the model holds significant power. Mitigation requires

making AI-driven decisions auditable and transparent. A critical research avenue here is the application of Explainable AI to enhance decision transparency. Explainable AI methods can provide clear, human-understandable justifications for the AI's recommendations, which is essential for auditability and for building trust among consortium members. This can be complemented by logging decision rationales and further decentralizing control through multi-party governance, such as threshold signatures for model updates and implementing redundant, independent models for critical decisions.

Data Assurance. The integrity of AI-driven systems depends on the integrity of their data, introducing data assurance challenges. The risk of training data poisoning or adversarial inputs can compromise the entire system's logic. Mitigation strategies include establishing strong data provenance and traceability for training data, continuous monitoring for anomalous inputs or prediction patterns, and ensuring that critical AI-driven actions are backed by verifiable cryptographic proofs or have well-defined, secure rollback paths.

6 Conclusion

Scalability remains a critical bottleneck for the widespread adoption of consortium blockchains in high-demand applications. This paper provides a comprehensive survey of the approaches designed to improve scalability, viewed through the dual lens of scaling approaches and their AI-enabled enhancements. Scaling approaches inevitably introduce significant costs, particularly in operational complexity and system resources, creating a clear trade-off between performance and cost. AI primarily mitigates these costs by providing dynamic adaptation, which automates complex management tasks and optimizes resource utilization, thereby reducing operational complexity and system resource demands. AI fails to resolve the fundamental trust & control costs inherent in scaling approaches and simultaneously introduces its own set of trade-offs. Our review of AI-enabled methods demonstrates their powerful capability to not only further enhance scalability but also to mitigate these inherent costs. However, this survey also highlights that AI is not a panacea. It fails to eliminate all costs and introduces new trade-offs.

In conclusion, addressing consortium blockchain scalability is crucial to realizing the technology's full potential. Future opportunities focus on the co-design of scaling approaches and lightweight, verifiable AI, creating a synergy between structural innovation and intelligent optimization. One promising research opportunity is to mitigate the persistent trust and control costs that arise when scaling approaches centralize responsibilities. For instance, exploring federated reinforcement learning can enhance collaborative scaling by decentralizing the training of optimization models. Another critical opportunity is managing the new trade-offs introduced by AI integration. To address this, applying Explainable AI is crucial for providing decision transparency and ensuring the auditability and trustworthiness of AI-driven systems.

It is important to acknowledge a limitation of our work. Due to variations in experimental platforms, baseline implementations, and metric standards across existing studies, our comparisons offer directional conclusions rather than absolute rankings. Establishing standardized benchmarks and reproducible experiments will be a crucial direction for future validation. By systematically mapping the intricate landscape of performance and costs for scaling approaches and AI-enabled methods, this survey provides a structured foundation for future opportunities, guiding the development of the next generation of efficient, robust, and cost-aware blockchain systems.

Acknowledgement: Not applicable.

Funding Statement: This paper is supported by the National Natural Science Foundation of China under Grant No. 62472077.

Author Contributions: Conceptualization, Jie Song; methodology, Yingying Zhou and Chaopeng Guo; formal analysis, Yingying Zhou and Chaopeng Guo; data curation, Wenlong Shen and Tianzhe Jiao; writing—original draft preparation, Yingying Zhou; writing—review and editing, Jie Song; visualization, Wenlong Shen, Yingying Zhou and Tianzhe Jiao; supervision, Jie Song. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Reddy KRK, Gunasekaran A, Kalpana P, Sreedharan VR, Kumar SA. Developing a blockchain framework for the automotive supply chain: a systematic review. Comput Indust Eng. 2021;157:107334. doi:10.1016/j.cie.2021.107334.
2. Ng WY, Tan TE, Movva PVH, Fang AHS, Yeo KK, Ho D, et al. Blockchain applications in health care for COVID-19 and beyond: a systematic review. Lan Dig Health. 2021;3(12):E819–29. doi:10.1016/s2589-7500(21)00210-7.
3. Sonani R, Alhejaili R, Chatterjee P, Alnafisah KH, Ali J. Secure malicious node detection in decentralized healthcare networks using cloud and edge computing with blockchain-enabled federated learning. Comput Model Eng Sci. 2025;144(3):3169–89. doi:10.32604/cmes.2025.070225.
4. Kloeckner M, Schmidt CG, Wagner SM. When blockchain creates shareholder value: empirical evidence from international firm announcements. Product Operat Manag. 2022;31(1):46–64. doi:10.1111/poms.13609.
5. Gorenflo C, Lee S, Golab L, Keshav S. FastFabric: scaling hyperledger fabric to 20 000 transactions per second. Int J Netw Manag. 2020;30(5):e2099. doi:10.1002/nem.2099.
6. Amiri MJ, Agrawal D, El Abbadi A. SharPer: sharding permissioned blockchains over network clusters. In: SIGMOD '21: Proceedings of the 2021 International Conference on Management of Data. New York, NY, USA: ACM; 2021. p. 76–88.
7. Zhang Y, Wang X, He X. Nexus: efficient and conflict-equivalent DAG-based permissioned blockchain sharding. In: 2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS); 2024 Jun 19–21; Guangzhou, China. p. 1–6.
8. Wang K, Feng G, Ji Z, Tu Z, He S. SP-PoR: improve blockchain performance by semi-parallel processing transactions. Comput Netw. 2024;245:110394. doi:10.1016/j.comnet.2024.110394.
9. Yang L, Dong X, Tong W, Ma S, Qiao H, Xing S. Secure off-chain payment in consortium blockchain system. In: 2020 International Conference on Networking and Network Applications, NANA; 2020 Dec 10–13; Haikou, China. p. 259–64.
10. Freitas AES, Rodrigues LA, Duarte EP Jr. vCubeChain: a scalable permissioned blockchain. Ad Hoc Netw. 2024;158:103461. doi:10.1016/j.adhoc.2024.103461.
11. Xiao J, Luo T, Li C, Zhou J, Li Z. CE-PBFT: a high availability consensus algorithm for large-scale consortium blockchain. J King Saud Univ-Comput Inf Sci. 2024;36(2):101957. doi:10.1016/j.jksuci.2024.101957.
12. Fu X, Wang H, Shi P. Votes-as-a-Proof (VaaP): permissioned blockchain consensus protocol made simple. IEEE Trans Parallel Distrib Syst. 2022 Dec 1;33(12):4964–73. doi:10.1109/tpds.2022.3211829.
13. Krishnapriya S, Singh S. A hybrid machine learning and blockchain framework for IoT DDoS mitigation. Comput Model Eng Sci. 2025;144(2):1849–81.
14. Li M, Wang Y, Ma S, Liu C, Huo D, Wang Y, et al. Auto-tuning with reinforcement learning for permissioned blockchain systems. Proc VLDB Endowment. 2023;16(5):1000–12. doi:10.14778/3579075.3579076.
15. Li C, Qiu W, Li X, Liu C, Zheng Z. A dynamic adaptive framework for practical byzantine fault tolerance consensus protocol in the internet of things. IEEE Transact Comput. 2024;73(7):1669–82. doi:10.1109/tc.2024.3377921.
16. Dai H, Yuan L, Wu J, Chen H, Bao H. DBSRP-ML: dynamic blockchain sharding reconfiguration protocol based on multi-label. Comput Netw. 2025;269:111401. doi:10.1016/j.comnet.2025.111401.
17. Boughdiri M, Abdellatif T, Guegan CG. A systematic literature review on blockchain storage scalability. IEEE Access. 2025;13(1):102194–219. doi:10.1109/access.2025.3578451.

18. Rao IS, Kiah MLM, Hameed MM, Memon ZA. Scalability of blockchain: a comprehensive review and future research direction. Cluster Comput J Netw Softw Tools Applicat. 2024;27(5):5547–70. doi:10.1007/s10586-023-04257-7.
19. Sanka AI, Cheung RCC. A systematic review of blockchain scalability: issues, solutions, analysis and future research. J Netw Comput Appl. 2021;195(3):103232. doi:10.1016/j.jnca.2021.103232.
20. Ressi D, Romanello R, Piazza C, Rossi S. AI-enhanced blockchain technology: a review of advancements and opportunities. J Netw Comput Appl. 2024;225:103858. doi:10.1016/j.jnca.2024.103858.
21. Ural O, Yoshigoe K. Survey on blockchain-enhanced machine learning. IEEE Access. 2023;11:145331–62. doi:10.1109/access.2023.3344669.
22. Qammar A, Karim A, Ning H, Ding J. Securing federated learning with blockchain: a systematic literature review. Artif Intell Rev. 2023;56(5):3951–85. doi:10.1007/s10462-022-10271-9.
23. Song J, Zhang P, Alkubati M, Bao Y, Yu G. Research advances on blockchain-as-a-service: architectures, applications and challenges. Digital Communicat Netw. 2022;8(4):466–75. doi:10.1016/j.dcan.2021.02.001.
24. Dabbagh M, Choo KKR, Beheshti A, Tahir M, Safa NS. A survey of empirical performance evaluation of permissioned blockchain platforms: challenges and opportunities. Comput Secur. 2021;100:102078. doi:10.1016/j.cose.2020.102078.
25. Zhuang P, Zamir T, Liang H. Blockchain for cybersecurity in smart grid: a comprehensive survey. IEEE Transact Indust Inform. 2021;17(1):3–19. doi:10.1109/tii.2020.2998479.
26. Polcumpally AT, Pandey KK, Kumar A, Samadhiya A. Blockchain governance and trust: a multi-sector thematic systematic review and exploration of future research directions. Heliyon. 2024;10(12):e32975. doi:10.1016/j.heliyon.2024.e32975.
27. Ren K, Ho NM, Loghin D, Nguyen TT, Ooi BC, Ta QT, et al. Interoperability in blockchain: a survey. IEEE Transact Know Data Eng. 2023;35(12):12750–69. doi:10.1109/tkde.2023.3275220.
28. Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, De Caro A, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In: EUROSYS '18: Proceedings of the Thirteenth Eurosys Conference; 2018 Apr 23–26; Porto, Portugal. p. 1–15.
29. Brown RG, Carlyle J, Grigg I, Hearn M. Corda: an introduction. R3 CEV. 2016 Aug;1(15):14.
30. Baliga A, Subhod I, Kamat P, Chatterjee S. Performance evaluation of the quorum blockchain platform. arXiv:1809.03421. 2018.
31. Liu Y, He J, Li X, Chen J, Liu X, Peng S, et al. An overview of blockchain smart contract execution mechanism. J Ind Inf Integr. 2024;41:100674. doi:10.1016/j.jii.2024.100674.
32. da Silva Costa TB, Shinoda L, Moreno RA, Krieger JE, Gutierrez M. Blockchain-based architecture design for personal health record: development and usability study. J Med Internet Res. 2022;24(4):e35013. doi:10.2196/35013.
33. Rehman M, Javed IT, Qureshi KN, Margaria T, Jeon G. A cyber secure medical management system by using blockchain. IEEE Transact Computat Social Syst. 2023;10(4):2123–36. doi:10.1109/tcss.2022.3215455.
34. Altarawneh A, Herschberg T, Medury S, Kandah F, Skjellum A. Buterin's scalability trilemma viewed through a state-change-based classification for common consensus algorithms. In: 2020 10th Annual Computing and Communication Workshop and Conference (CCWC); 2020 Jan 6–8; Las Vegas, NV, USA. p. 727–36.
35. Reno S, Roy K. Navigating the blockchain trilemma: a review of recent advances and emerging solutions in decentralization, security, and scalability optimization. Comput Mater Contin. 2025;84(2):2061–119. doi:10.32604/cmc.2025.066366.
36. Song J, Zhang S, Zhang P, Park J, Gu Y, Yu G. Illicit social accounts? Anti-money laundering for transactional blockchains. IEEE Transact Inform Foren Secur. 2025;20:391–404. doi:10.1109/tifs.2024.3518068.
37. Song J, Zhang S, Park J, Gu Y, Yu G. Dimension expansion for learning money laundering activities hidden in transaction network. IEEE Transact Computat Soc Syst. 2025. doi:10.1109/tcss.2025.3599534.
38. Solat S. Parallel Committees: a scalable, secure, and fault-tolerant distributed NoSQL database architecture. Cluster Comput J Netw Soft Tools Applicat. 2025;28(4):254. doi:10.1007/s10586-024-05081-3.
39. Qi X. S-Store: a scalable data store towards permissioned blockchain sharding. In: IEEE Conference on Computer Communications (IEEE INFOCOM 2022); 2022 May 2–5; London, UK. p. 1978–87.

40. Matani A, Sahafi A, Broumandnia A. Improving scalability in blockchain systems using multi-level sharding based on heterogeneity of network nodes. Computing. 2025;107(2):63. doi:10.1007/s00607-025-01414-1.
41. Hong Z, Guo S, Li P, Chen W. Pyramid: a layered sharding blockchain system. In: IEEE INFOCOM, 2021-IEEE Conference on Computer Communications; 2021 May 10–13; Vancouver, BC, Canada. p. 1–10.
42. Li C, Pan H, Qian H, Li Y, Si X, Li K, et al. Hierarchical sharding blockchain storage solution for edge computing. Future Generat Comput Syst Int J Esci. 2024;161:162–73. doi:10.1016/j.future.2024.06.048.
43. Song J, Zhang P, Qu Q, Bai Y, Gu Y, Yu G. Why blockchain needs graph: a survey on studies, scenarios, and solutions. J Parallel Distr Comput. 2023;180(1):104730. doi:10.1016/j.jpdc.2023.104730.
44. Gupta H, Janakiram D. CDAG: a serialized blockDAG for permissioned blockchain. arXiv:1910.08547. 2019.
45. Jo Y, Park C. Enhancing ethereum PoA clique network with DAG-based BFT consensus. In: 2024 IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2024; 2024 Aug 19–22; Copenhagen, Denmark. p. 655–9.
46. Xiang F, Huaimin W, Peichang S, Xue O, Xunhui Z. Jointgraph: a DAG-based efficient consensus algorithm for consortium blockchains. Softw Pract Exper. 2021;51(10):1987–99. doi:10.1002/spe.2748.
47. Spiegelman A, Giridharan N, Sonnino A, Kokoris-Kogias L. Bullshark: dag BFT protocols made practical. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2022. p. 2705–18.
48. Feng L, Deng X. An efficient hot/cold data separation scheme for storage optimization in consortium blockchain full nodes. Cluster Comput. 2025;28(9):1–20. doi:10.1007/s10586-025-05301-4.
49. Xu C, Zhang C, Xu J, Pei J. SlimChain: scaling blockchain transactions through off-chain storage and parallel processing. Proc VLDB Endowment. 2021;14(11):2314–26.
50. Yu J, Zhang X, Wang J, Zhang Y, Shi Y, Su L, et al. Robust and trustworthy data sharing framework leveraging on-chain and off-chain collaboration. Comput Mater Contin. 2024;78(2):2159–79. doi:10.32604/cmc.2024.047340.
51. Wu Z, Wang Y, Wang L. GAM: a scalable and efficient multi-chain data sharing scheme. Inform Process Manag. 2025;62(3):104004. doi:10.1016/j.ipm.2024.104004.
52. Liu C. HPCLS-BC: a novel blockchain framework using heterogeneous peer-node and cloud-based ledger storage for Internet of Things applications. Future Generat Comput Syst Int J Esci. 2024;150:364–79. doi:10.1016/j.future.2023.09.015.
53. Buterin V, Feist D, Loerakker D, Kadianakis G, Garnett M, Taiwo M, et al. EIP-4844: shard blob transactions [Internet]; 2022 [cited 2025 Oct 29]. Available from: https://eips.ethereum.org/EIPS/eip-4844.
54. You X, Zhang S, Wang T, Liew SC. Hyperclique: a novel P2P network structure for blockchain systems. In: 2024 International Conference on Computing, Networking and Communications; 2024 Feb 19–22; Big Island, HI, USA. p. 804–9.
55. Qi J, Chen X, Jiang Y, Jiang J, Shen T, Zhao S, et al. Bidl: a high-throughput, low-latency permissioned blockchain framework for datacenter networks. In: Proceedings of the 28th ACM Symposium on Operating Systems Principles, SOSP 2021; 2021 Oct 26–29; Online. p. 18–34.
56. Huang B, Jin L, Lu Z, Zhou X, Wu J, Tang Q, et al. BoR: toward high-performance permissioned blockchain in RDMA-enabled network. IEEE TRansact Serv Comput. 2020;13(2):301–13. doi:10.1109/tsc.2019.2948009.
57. Xu M, Liu S, Yu D, Cheng X, Guo S, Yu J. CloudChain: a cloud blockchain using shared memory consensus and RDMA. IEEE Transact Comput. 2022;71(12):3242–53. doi:10.1109/tc.2022.3147960.
58. Cheng L, Tan H, Li X, Pan W, Zhao H, Yuan M, et al. A hierarchical overlay network optimisation model for enhancing data transmission performance in blockchain systems. Sci Rep. 2024;14(1):31900. doi:10.1038/s41598-024-83399-z.
59. Neiheiser R, Matos M, Rodrigues L. Kauri: BFT consensus with pipelined tree-based dissemination and aggregation. ACM Trans Comput Syst. 2025;19:61. doi:10.1145/3769423.
60. Zhou Z, Qiu Z, Yu Q, Chen H. A dynamic sharding protocol design for consortium blockchains. In: 2020 IEEE International Conference on Big Data (BIG DATA); 2020 Dec 10–13; Atlanta, GA, USA. p. 2590–5.

61. Hao W, Zeng J, Dai X, Xiao J, Hua QS, Chen H, et al. Towards a trust-enhanced blockchain P2P topology for enabling fast and reliable broadcast. IEEE Transacti Netw Serv Manag. 2020;17(2):904–17. doi:10.1109/tnsm.2020.2980303.
62. Gupta S, Nandi S. Optimized node placement and dynamic session node selection for permissioned blockchain in industrial IoT. Future Generat Comput Syst Int J Esci. 2026;175(1):108057. doi:10.1016/j.future.2025.108057.
63. Deshpande V, Badis H, George L. Efficient topology control of blockchain peer to peer network based on SDN paradigm. Peer-to-Peer Netw Applicat. 2022;15:267–89. doi:10.1007/s12083-021-01248-6.
64. Xiao Y, Liu Y, Li T. Edge computing and blockchain for quick fake news detection in IoV. Sensors. 2020;20(16):4360. doi:10.3390/s20164360.
65. Pei X, Li H, Tan S, Huang W, Zhang Y, Wang H. Matching-gossip: optimizing blockchain broadcast performance to address the CAP trilemma. In: 2024 6th International Conference on Blockchain Computing and Applications; 2024 Nov 26–29; Dubai, United Arab Emirates. p. 263–70.
66. Berendea N, Mercier H, Onica E, Riviere E. Fair and efficient gossip in hyperledger fabric. In: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS); 2020 Nov 29–Dec 1; Singapore. p. 190–200.
67. Dong X, Dang X, Xu Z, Ye K, Han H, Zheng E. DC-SoC: optimizing a blockchain data dissemination model based on density clustering and social mechanisms. Appl Sci. 2024;14(21):10058. doi:10.3390/app142110058.
68. Zhao C, Zhang S, Wang T, Liew SC. Bodyless block propagation: TPS fully scalable blockchain with pre-validation. Future Generat Comput Syst Int J Esci. 2025;163:107516. doi:10.1016/j.future.2024.107516.
69. Bai F, Xu H, Shen T, Zeng K, Zhang X, Zhang C. RBC-MSS: asynchronous broadcasting protocol based on multi-secret sharing. J Supercomput. 2025;81(6):768. doi:10.1007/s11227-025-07211-z.
70. Dolev S, Wang Z. SodsBC: stream of distributed secrets for quantum-safe blockchain. In: 2020 IEEE International Conference on Blockchain (BLOCKCHAIN 2020); 2020 May 2–6; Toronto, ON, Canada. p. 247–56.
71. Shrestha N, Yu Q, Kate A, Losa G, Nayak K, Wang X. Optimistic, signature-free reliable broadcast and its applications. arXiv:2505.02761. 2025.
72. Rohrer E, Tschorsch F. Kadcast-NG: a structured broadcast protocol for blockchain networks. IEEE-ACM Transact Netw. 2023;31(6):3269–83. doi:10.1109/tnet.2023.3279897.
73. Liu C, Li M, Wang Y, Wang Y, Huo D, Chen Y. Achieve better endorsement balance on blockchain systems. In: Proceedings of the 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD); 2021 May 5–7; Dalian, China. p. 581–6.
74. Cao N, Han H, Lu Y, Liu J, Tian W, Li X, et al. A dynamic decision-making method for endorser node selection. In: International Conference on Artificial Intelligence and Security. Cham, Switzerland: Springer; 2021. p. 690–9.
75. Zheng P, Xu Q, Zheng Z, Zhou Z, Yan Y, Zhang H. Meepo: multiple execution environments per organization in sharded consortium blockchain. IEEE J Select Areas Communicat. 2022;40(12):3562–74. doi:10.1109/jsac.2022.3213326.
76. Camaioni M, Guerraoui R, Komatovic J, Monti M, Roman PL, Vidigueira M, et al. Carbon: scaling trusted payments with untrusted machines. IEEE Transact Depend Secure Comput. 2025;22(2):1168–80. doi:10.1109/tdsc.2024.3428617.
77. Jial J, Wu O, Li S, Mao R, Zhang H. A scheduling algorithm for hyperledger fabric based on transaction batch processing. In: 2024 23rd International Symposium on Parallel and Distributed Computing, ISPDC 2024; 2024 Jul 8–10; Chur, Switzerland. p. 1–8.
78. Thakkar P, Nathan SN, Viswanathan B. Performance benchmarking and optimizing hyperledger fabric blockchain platform. In: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS); 2018 Sep 25–28; Milwaukee, WI, USA. p. 264–76.
79. Amiri MJ, Agrawal D, El Abbadi A. ParBlockchain: leveraging transaction parallelism in permissioned blockchain systems. In: 2019 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019); 2019 July 7–10; Dallas, TX, USA. p. 1337–47.
80. Qi X, Chen Z, Zhuo H, Xu Q, Zhu C, Zhang Z, et al. SChain: scalable concurrency over flexible permissioned blockchain. In: 2023 IEEE 39th International Conference on Data Engineering (ICDE); 2023 Apr 3–7; Anaheim, CA, USA. p. 1901–13.

81. Jin C, Pang S, Qi X, Zhang Z, Zhou A. A high performance concurrency protocol for smart contracts of permissioned blockchain. IEEE Transact Know Data Eng. 2022;34(11):5070–83. doi:10.1109/tkde.2021.3059959.
82. Yu B, Zhao H, Zhou T, Chen L, Fan Y. TCHAO: parallelism with transaction classification and grouping by historical access objects to scale blockchain. J King Saud Univ Comput Inform Sci. 2025;37(4):67. doi:10.1007/s44443-025-00090-7.
83. Javaid H, Hu C, Brebner G. Optimizing validation phase of hyperledger fabric. In: 2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2019); 2019 Oct 21–25; Rennes, France. p. 269–75.
84. Kaul S, Piduguralla M, Patnala GS, Peri S. Dependency-aware execution mechanism in hyperledger fabric architecture. arXiv:2509.07425. 2025.
85. Xiao K, Li J, He Y, Wang X, Wang C. A secure multi-party payment channel on-chain and off-chain supervisable scheme. Future Generat Comput Syst Int J Esci. 2024;154:330–43. doi:10.1016/j.future.2024.01.012.
86. Sui Y, Yang X, Wang B, Zhang Y, Wang W, Wang L. ScorpioBase: an efficient batched blockchain system for traceability applications in supply chain. IEEE Transact Indust Inform. 2025;21(5):3575–84. doi:10.1109/tii.2025.3528530.
87. Shi H, Chen Z, Cheng Y, Liu X, Wang Q. PB-Raft: a Byzantine fault tolerance consensus algorithm based on weighted PageRank and BLS threshold signature. Peer-to-Peer Netw Applicat. 2025;18(1):22.
88. Fu W, Wei X, Tong S. An improved blockchain consensus algorithm based on raft. Arabian J Sci Eng. 2021;46:8137–49. doi:10.1007/s13369-021-05427-8.
89. Lu S, Zhang X, Zhao R, Chen L, Li J, Yang G. P-Raft: an efficient and robust consensus mechanism for consortium blockchains. Electronics. 2023;12(10):2271. doi:10.3390/electronics12102271.
90. Kondru KK, Rajiakodi S. RaftOptima: an optimised raft with enhanced fault tolerance, and increased scalability with low latency. IEEE Access. 2024;12:105974–89. doi:10.1109/access.2024.3435978.
91. Shang J, Lu T, Cai Y, Li Y. DRaft: a double-layer structure for Raft consensus mechanism. J Netw Comput Appl. 2025;236:104111. doi:10.1016/j.jnca.2025.104111.
92. Ye Z, Tong X, Fan W, Zhang Z, Jin C. A raft variant for permissioned blockchain. In: International Conference on Database Systems for Advanced Applications. Cham, Switzerland: Springer; 2023. p. 685–9.
93. Yang H, Feng Y, Zhang W. LRD-Raft: log replication decouple for efficient and secure consensus in consortium-blockchain-based IoT. IEEE Internet Things J. 2025;12(7):8807–20. doi:10.1109/jiot.2024.3506601.
94. Barger A, Manevich Y, Meir H, Tock Y. A byzantine fault-tolerant consensus library for hyperledger fabric. In: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC); 2021 May 3–6; Sydney, NSW, Australia.
95. Yao D, Wang Y, Ding Y, Wu Q, Qin B, Yang C, et al. DyBFT: leaderless BFT protocol based on locally adjustable valid committee set mechanism. World Wide Web-Internet Web Inform Syst. 2025;28:17. doi:10.1007/s11280-024-01324-w.
96. Ye J, Hu H, Liang J, Yin L, Kang J. Lightweight adaptive Byzantine fault tolerant consensus algorithm for distributed energy trading. Comput Netw. 2024;251:110635. doi:10.1016/j.comnet.2024.110635.
97. Hyperledger Foundation. What's New in Hyperledger Fabric v3.1: byzantine Fault Tolerant (BFT) ordering service [Internet]. 2024 [cited 2025 Oct 29]. Available from: https://hyperledger-fabric.readthedocs.io/en/latest/whatsnew.html#byzantine-fault-tolerant-bft-ordering-service.
98. Cheng T, Zhou W, Yao S, Feng L, He J. Multi-pipeline HotStuff: a high performance consensus for permissioned blockchain. In: 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TRUSTCOM; 2022 Dec 9–11; Wuhan, China. p. 1008–19.
99. Wu G, Wang H, Yang Z, He D, Chan S. Electronic health records sharing based on consortium blockchain. J Med Syst. 2024;48(1):106.
100. Tian J, Tian J, Xu H. TSBFT: a scalable and efficient leaderless byzantine consensus for consortium blockchain. Comput Netw. 2023;222:109541. doi:10.1016/j.comnet.2022.109541.

101. Gao Y, Lu Y, Lu Z, Tang Q, Xu J, Zhang Z. Dumbo-ng: fast asynchronous BFT consensus with throughput-oblivious latency. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security; 2022 Nov 7–11; Los Angeles, CA, USA. p. 1187–201.
102. Zarbafian P, Gramoli V. Lyra: fast and scalable resilience to reordering attacks in blockchains. In: 2023 IEEE International Parallel and Distributed Processing Symposium; 2023 May 15–19; St. Petersburg, FL, USA. p. 929–39.
103. Gueta GG, Abraham I, Grossman S, Malkhi D, Pinkas B, Reiter M, et al. SBFT: a scalable and decentralized trust infrastructure. In: 2019 49TH Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2019); 2019 Jun 24–27; Portland, OR, USA. p. 568–80.
104. Shi J, Zeng X, Li Y. Reputation-based sharding consensus model in information-centric networking. Electronics. 2022;11(5):830. doi:10.3390/electronics11050830.
105. Tian J, Tian J, Du R. MSLShard: an efficient sharding-based trust management framework for blockchain-empowered IoT access control. J Parallel Distr Comput. 2024;185:104795. doi:10.1016/j.jpdc.2023.104795.
106. Xi J, Xu G, Zou S, Lu Y, Li G, Xu J, et al. A blockchain dynamic sharding scheme based on hidden markov model in collaborative IoT. IEEE Internet Things J. 2023;10(16):14896–907. doi:10.1109/jiot.2023.3294234.
107. Zhen Z, Wang X, Lin H, Garg S, Kumar P, Hossain MS. A dynamic state sharding blockchain architecture for scalable and secure crowdsourcing systems. J Netw Comput Appl. 2024;222:103785. doi:10.1016/j.jnca.2023.103785.
108. Cai T, Chen W, Zhang J, Zheng Z. SmartChain: a dynamic and self-adaptive sharding framework for IoT blockchain. IEEE Transact Serv Computg. 2024;17(2):674–88. doi:10.1109/tsc.2024.3376242.
109. Wang Y, Gong Z, Jia D, Tan A, Liu M. Dynamic sharding model and performance optimization method for consortium blockchain. J Supercomput. 2025;81(2):411. doi:10.1007/s11227-024-06870-8.
110. Tian J, Tian J, Du R. MSLTChain: a trust model based on the multi-dimensional subjective logic for tree sharding blockchain system. IEEE Transact Netw Service Manag. 2024;21(5):5566–81. doi:10.1109/tnsm.2024.3427139.
111. Hu S, Lin J, Du X, Huang W, Lu Z, Duan Q, et al. ACSarF: a DRL-based adaptive consortium blockchain sharding framework for supply chain finance. Digital Communicat Netw. 2025;11(1):26–34. doi:10.1016/j.dcan.2023.11.008.
112. Zhang Z, Yu G, Sun C, Wang X, Wang Y, Zhang M, et al. TBDD: a new trust-based, DRL-driven framework for blockchain sharding in IoT. Comput Netw. 2024;244:110343. doi:10.1016/j.comnet.2024.110343.
113. Chen T, Lin H, Wu L, Wang X. Optimized block-K clustering IoT clustering and blockchain sharding strategy using deep reinforcement learning. In: 2024 IEEE Cyber Science and Technology Congress, CYBERSCITECH; 2024 Nov 5–8; Boracay Island, Philippines. p. 45–51. doi:10.1109/cyberscitech64112.2024.00018.
114. Zhu X, Yuan Z, Ni Q. A self evolving high performance sharded consortium blockchain designed for secure and trusted resource sharing for 6G networks. Sci Rep. 2025;15(1):28098. doi:10.1038/s41598-025-11705-4.
115. Lee J, Yoo Y, Yoo C, Yang G. Predictive placement of geo-distributed blockchain nodes for performance guarantee. In: 2024 IEEE 17th International Conference on Cloud Computing (CLOUD); 2024 Jul 7–13; Shenzhen, China. p. 66–8.
116. Xu Z, Ye K, Dong X, Han H, Yan Z, Chen X, et al. DC-Gossip: an enhanced broadcast protocol in hyperledger fabric based on density clustering. In: Wireless algorithms, systems, and applications. Vol. 13473. Cham, Switzerland: Springer; 2022. p. 3–19. doi:10.1007/978-3-031-19211-1_1.
117. Kang J, Liao J, Gao R, Wen J, Huang H, Zhang M, et al. Efficient and trustworthy block propagation for blockchain-enabled mobile embodied AI networks: a graph resfusion approach. arXiv:2502.09624. 2025.
118. Liu M, Yu FR, Teng Y, Leung VCM, Song M. Performance optimization for blockchain-enabled industrial internet of things (IIoT) systems: a deep reinforcement learning approach. IEEE Transact Indust Inform. 2019;15(6):3559–70. doi:10.1109/tii.2019.2897805.
119. Pei Y, Zhu M, Zhu C, Song W, Sun Y, Li L, et al. Meta reinforcement learning based dynamic tuning for blockchain systems in diverse network environments. Blockchain Res Applicat. 2025;6(2):100261. doi:10.1016/j.bcra.2024.100261.
120. Liao J, Went J, Kang J, Zhang Y, Du J, Li Q, et al. Optimizing information propagation for blockchain-empowered mobile AIGC: a graph attention network approach. In: 20th International Wireless Communications & Mobile Computing Conference, IWCMC 2024; 2024 May 27–31; Ayia Napa, Cyprus. p. 685–90.

121. Wang J, Wang Y, Zhang X, Jin Z, Zhu C, Li L, et al. LearningChain: a highly scalable and applicable learning-based blockchain performance optimization framework. IEEE Transact Netw Serv Manag. 2024;21(2):1817–31. doi:10.1109/TNSM.2023.3347789.
122. Jung S, Yoo Y, Yang G, Yoo C. Prediction of permissioned blockchain performance for resource scaling configurations. ICT Express. 2024;10(6):1253–8. doi:10.1016/j.icte.2024.09.003.
123. Sapkota S, Huang H, Hu Y, Hussain F. A deep neural network (DNN) based contract policy on hyperledger fabric for secure internet of things (IoTs). In: Advanced information networking and applications, AINA 2024. Vol. 201. Cham, Switzerland: Springer; 2024. p. 313–25 doi:10.1007/978-3-031-57870-0_28.
124. Wu C, Mehta B, Amiri MJ, Marcus R, Loo BT. AdaChain: a learned adaptive blockchain. Proc VLDB Endowment. 2023;16(8):2033–46. doi:10.14778/3594512.3594531.
125. Zhang Y, Lin J, Lu Z, Duan Q, Huang SC. PBRL-TChain: a performance-enhanced permissioned blockchain for time-critical applications based on reinforcement learning. Future Generat Comput Syst Int J Esci. 2024;154:301–13. doi:10.1016/j.future.2023.12.031.
126. Li J, Liu R, Liu W. FBMTO: fusion of blockchain and multi-agent reinforcement learning for task offloading in edge computing. Inf Fusion. 2025;124:103344. doi:10.1016/j.inffus.2025.103344.
127. Samy A, Elgendy IA, Yu H, Zhang W, Zhang H. Secure task offloading in blockchain-enabled mobile edge computing with deep reinforcement learning. IEEE Transact Netw Serv Manag. 2022;19(4):4872–87. doi:10.1109/tnsm.2022.3190493.
128. Wu M, Zhang Y, Yu J, Zhou Z. A dynamic resource-aware endorsement strategy for improving throughput in blockchain systems. Expert Syst Applicat. 2023;225:119989. doi:10.1016/j.eswa.2023.119989.
129. Yu J, Ge L, Wu M. Proposal distribution optimization for endorsement strategy in hyperledger fabric. J Supercomput. 2024;80(10):15038–65. doi:10.1007/s11227-024-06056-2.
130. Alotaibi R, Alassafi M, Bhuiyan MSI, Raju RS, Ferdous MS. A reinforcement-learning-based model for resilient load balancing in hyperledger fabric. Processes. 2022;10(11):2390. doi:10.3390/pr10112390.
131. Kim S, Ibrahim ASS. Byzantine-fault-tolerant consensus via reinforcement learning for permissioned blockchain-empowered V2X network. IEEE Transact Intell Veh. 2023;8(1):172–83. doi:10.1109/tiv.2022.3168575.
132. Nguyen DC, Ding M, Pathirana PN, Seneviratne A, Li J, Poor HV. Cooperative task offloading and block mining in blockchain-based edge computing with multi-agent deep reinforcement learning. IEEE Transact Mobile Comput. 2023;22(4):2021–37. doi:10.1109/tmc.2021.3120050.
133. Xie M, Ye J, Zhang G, Ni X. Deep Reinforcement Learning-based computation offloading and distributed edge service caching for Mobile Edge Computing. Comput Netw. 2024;250:110564. doi:10.1016/j.comnet.2024.110564.
134. Xie B, Cui H. Deep reinforcement learning-based dynamical task offloading for mobile edge computing. J Supercomput. 2025;81(1):35. doi:10.1007/s11227-024-06603-x.
135. Ritual Foundation. Infernet ↔ Chain [Internet]. 2024 [cited 2025 Oct 29]. Available from: https://ritualfoundation.org/docs/architecture/infernet-to-chain.
136. Choumas K, Korakis T. When machine learning meets raft: how to elect a leader over a network. In: IEEE Conference on Global Communications, GLOBECOM; 2023 Dec 4–8; Kuala Lumpur, Malaysia. p. 3705–10.
137. Yang D, Doh I, Chae K. Cell based raft algorithm for optimized consensus process on blockchain in smart data market. IEEE Access. 2022;10(1):85199–212. doi:10.1109/access.2022.3197758.
138. Deng X, Li K, Wang Z, Liu H. A novel consensus algorithm based on segmented DAG and BP neural network for consortium blockchain. Secur Communicat Netw. 2022. doi:10.1155/2022/1060765.
139. Riahi K, Brahmia M, Abouaissa A, Idoumghar L. Multi-task learning for PBFT optimisation in permissioned blockchains. Blockchain Res Applicat. 2024;5(3):100206. doi:10.1016/j.bcra.2024.100206.
140. Qiu C, Yu FR, Yao H, Jiang C, Xu F, Zhao C. Blockchain-based software-defined industrial internet of things: a dueling deep Q-learning approach. IEEE Internet Things J. 2019;6(3):4627–39. doi:10.1109/jiot.2018.2871394.
141. Ameri R, Meybodi MR. The cellular goore game-based consensus protocol: a cognitive model for blockchain consensus. Cluster Comput J Netw Softw Tools Applicat. 2024;27(3):2715–40. doi:10.1007/s10586-023-04108-5.
142. Sanghami SV, Lee JJ, Hu Q. Machine-learning-enhanced blockchain consensus with transaction prioritization for smart cities. IEEE Internet Things J. 2023;10(8):6661–72. doi:10.1109/jiot.2022.3175208.

143. Wang X, Zhang H, Zhang J, Ge Y, Cui K, Peng Z, et al. An improved practical byzantine fault-tolerant algorithm based on XGBoost grouping for consortium chains. Comput Mater Contin. 2025;82(1):1295–311. doi:10.32604/cmc.2024.058559.
144. Ding J, Wu X, Tian J, Li Y. RE-BPFT: an improved PBFT consensus algorithm for consortium blockchain based on node credibility and ID3-based classification. Appl Sci. 2025;15(13):7591. doi:10.3390/app15137591.
145. Wu C, Qin H, Amiri MJ, Loo BT, Malkhi D, Marcus R. {BFTBrain}: adaptive {BFT} consensus with reinforcement learning. In: 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25); 2025 Apr 28–30; Philadelphia, PA, USA. p. 1563–83.
146. Yang L, Park SJ, Alizadeh M, Kannan S, Tse D. DispersedLedger: high-throughput byzantine consensus on variable bandwidth networks. In: Proceedings of the 19th Usenix Symposium on Networked Systems Design and Implementation (NSDI '22); 2022 Apr 4–6; Renton, WA, USA. p. 493–512.
147. Lu Y, Cui L, Wang Y, Sun J, Liu L. Residential energy consumption forecasting based on federated reinforcement learning with data privacy protection. Comput Model Eng Sci. 2023;137(1):717–32.
148. Guo C, Zhong Z, Zhang Z, Song J. NeurstrucEnergy: a bi-directional GNN model for energy prediction of neural networks in IoT. Digital Communicat Netw. 2024;10(2):439–49. doi:10.1016/j.dcan.2022.09.006.