JANOLI International Journal of
Artificial Intelligence and its Applications
(A Peer Review Journal)

journal homepage: https://janolijournals.com/index.php/jijems

Volume 3, Issue 3, 2026 ISSN: 3048-6815
DOI: https://doi.org/10.64758/ckpwjj33

A Mathematical Framework for Performance Multi-Objective Optimization of Blockchain Sharding with MongoDB Distributed Storage

Divyesh Bhatnagar[1], Anil Gupta[1,*]

[1]Department of Computer Science and Engineering, MBM University, Jodhpur, Rajasthan, India

E-mail: divyeshbhatnagar@gmail.com, anil.cse@mbm.ac.in

Article History:
Received June 03, 2026
First revision - August 30, 2026
Second revision - September 04, 2026
Acceptance - September 09, 2026
Published - September 13, 2026

Keywords:
Blockchain Sharding,
MongoDB,
Distributed Storage,
Weighted-Sum Optimization

Correspondence:
E-mail:
divyeshbhatnagar@gmail.com

Blockchain sharding partitions network state and transaction load across multiple parallel subchains (shards) in order to overcome the throughput ceiling of monolithic consensus protocols. In this study, we develop a mathematical performance model for blockchain sharding in which the underlying persistence layer is realized as a MongoDB distributed cluster, so that shard placement, chunk migration, and query routing are governed by MongoDB's own sharding sub-system, namely the mongos routers, the config servers, and the balancer. Closed form expressions are formulated for aggregate throughput, transaction latency, load-balance fairness, and scalability performance as functions of the shard count n, and a multi-objective optimization criterion is derived for selecting a practical operating point. A numerical analysis over n = 1 to 64 shards quantifies the trade-off between raw throughput growth and cross-shard coordination overhead. The analysis shows that the marginal throughput gain per doubling of shard count falls from 90.5% (for n = 1 to 2) to 22.9% (for n = 32 to 64), while scalability performance drops below 75% once n exceeds 8. The results indicate a practical operating range of 8 to 16 shards for MongoDB backed sharded blockchain deployments under the assumed workload parameters, beyond which coordination and rebalancing costs dominate any additional throughput gains.

1. Introduction and related work

Public and permissioned blockchains face a well-documented scalability trilemma among decentralization, security, and throughput. Sharding addresses the throughput dimension by splitting the network into committees that each process a disjoint subset of transactions and state in parallel, with results later reconciled through a coordination or beacon layer. Consensus-level sharding designs, including committee reconfiguration and cross-shard atomic commit, have been studied extensively;

however, the storage layer that persists shard state is frequently treated as a black box. In practice, many permissioned and consortium blockchain implementations delegate state persistence to general-purpose distributed databases, and MongoDB is a common choice because its native sharding architecture, comprising shard-key selection, chunk splitting, and an automated balancer, closely mirrors the partitioning problem that blockchain sharding must itself solve. This paper treats the pairing of blockchain sharding with MongoDB distributed storage as a single optimization problem, in which the blockchain layer decides how many shards to run and how transactions map to them, while MongoDB decides how the resulting state is chunked and balanced across physical nodes.

Sharding for blockchain consensus and state has been explored in systems such as OmniLedger, which introduces a secure, scale-out ledger via sharding with an emphasis on cross-shard transaction atomicity [8], and Monoxide, which proposes asynchronous consensus zones to scale throughput near-linearly with the number of zones [9]. Kudzin et al. refine the cross-shard data structures used in Ethereum 2.0 to reduce cross-shard transaction cost, directly targeting the coordination-overhead term modeled as α [4]. These works focus primarily on consensus and network-layer partitioning rather than the underlying storage engine. On the storage side, prior work has examined how the persistence layer itself can be optimized for sharded blockchains. Jia et al. propose an optimized data storage method tailored to sharding-based blockchains [15], while Liu, Wang, and Jin study improvements to MongoDB's own auto-sharding balancer in cloud environments [20], and Dhanagari examines the trade-off between performance and consistency guarantees in MongoDB deployments [21]. The present paper connects these two bodies of work by modeling how blockchain-level shard count interacts with MongoDB-level chunk balancing to determine overall system throughput and latency. Recent work has continued to push on both sides of this problem. Bhatnagar and Gupta present an empirical implementation of sharding to raise blockchain throughput [1], and, in a companion study, evaluate database sharding specifically as a mechanism for blockchain performance enhancement [2], directly motivating the storage-layer focus taken here. Hashim, Shuaib, and Zaki provide a broad review connecting database-sharding techniques to blockchain scalability [5], and Morsidi et al. offer a more recent systematic review of blockchain sharding for scalability [11]. Wu et al. propose a sharding protocol that reconfigures account-transaction graphs to reduce the cross-shard transaction ratio and account-migration cost [10]; Liu et al. study throughput optimization under dynamic sharding [16]; and Wang et al. propose a dynamic sharding and performance-optimization model for consortium blockchains [17], all addressing the same coordination-overhead and adaptivity problems that the throughput model. Bulgakov et al. evaluate several sharding algorithms (Elastico, OmniLedger, Pyramid, RepChain, SSChain) together with prospects for decentralized data storage, situating MongoDB-style distributed storage within the broader sharding-algorithm landscape [3]. Katkol et al. and Prabakaran and Balakrishnan report adaptive and dynamic sharding gains in healthcare and Ethereum-scalability settings respectively, illustrating the applicability of shard-count

tuning across deployment domains [18], and [19]. Jo and Park propose a hierarchical sharded architecture that balances performance against data availability, reporting throughput and latency gains over a balanced-sharding baseline [7], a trade-off structurally similar to the throughput with respect to performance. Cai et al. apply a many-objective optimization algorithm to sharding-scheme security in industrial IoT blockchains, an approach conceptually related to the weighted-sum multi-objective criterion adopted by [14]. These studies collectively motivate treating the consensus-layer shard count and the storage-layer chunk-distribution strategy as a coupled optimization problem, as done in this paper. The fairness index used to quantify load balance quality follows the general form introduced by Jain et al. for measuring fairness of resource allocation in shared computer systems [6], adapted to shard-level transaction load rather than network bandwidth allocation.

2. MongoDB-Backed Blockchain Sharding

The proposed system architecture considers of three layers as mentioned in Figure 1. The application layer accepts client transactions and assigns each transaction to a shard using a deterministic shard function, for example a hash of the sender account or contract address. The consensus layer runs an independent Byzantine fault-tolerant protocol within each shard committee and periodically checkpoints shard state to persistent storage. The storage layer is a MongoDB sharded cluster in which each blockchain shard corresponds to one or more MongoDB chunks: the mongos routers direct reads and writes to the correct shard server, the config servers track chunk-to-shard mapping, and the balancer migrates chunks when load becomes skewed, consistent with the auto-sharding improvements studied by Liu et al. [20] and the performance consistency by Dhanagari [21]. Two partitioning decisions therefore compound: the blockchain-level decision of how many consensus shards to run, and the MongoDB-level decision of how many chunks to allocate per shard and when to rebalance them, the latter closely related to the optimized sharding-aware storage layout proposed by Jia et al. [15]. The mathematical model treats these as a single effective shard count n for tractability, while Table 3 separately compares the three MongoDB shard strategies (range, hash, and zone) that determine how evenly chunks distribute in practice.

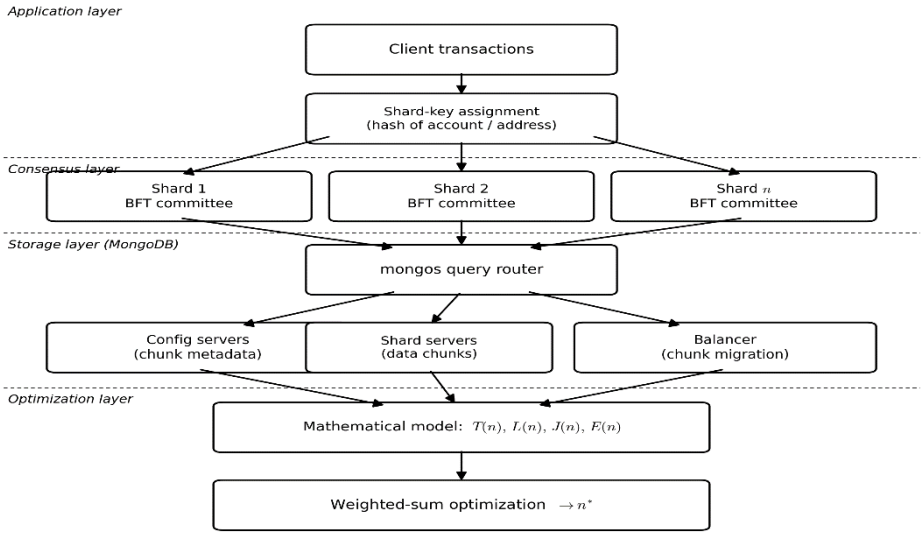

Figure 1: System architecture of MongoDB-backed blockchain sharding.

3. Notations

3.1. Notation

Notations	Description
n	number of active shards
T_0	baseline single-shard throughput (transactions per second)
α	cross-shard coordination overhead coefficient $(0 < \alpha < 1)$
L_0	baseline routing/lookup latency (ms)
β	per-hop routing latency coefficient $(mongos \rightarrow config\ server \rightarrow shard)$
γ	cross-shard commit/coordination latency coefficient
ξ	transaction load observed on shard i

4. Mathematical Model Formulation

The proposed mathematical model is developed to evaluate the effect of shard count on the entire performance of a sharded blockchain and database system. The formulation considers throughput, latency, load-balance fairness, scalability performance, and multi-objective optimization to identify an appropriate and optimal shard configuration.

4.1 Throughput Model

Aggregate throughput would scale linearly with shard count under perfect parallelism. In practice, a fraction of transactions requires cross-shard coordination (e.g., two-phase commit across shard boundaries, or MongoDB chunk-migration contention during rebalancing), which grows with n, the same effect that motivates refined cross-shard data structures in Ethereum 2.0 [4] and dynamic-sharding throughput studies [16], and [17]. We model aggregate throughput as:

$$T(n) = \frac{T_0 \cdot n}{1 + \alpha(n-1)}$$

Here α aggregates the combined overhead of cross-shard atomic commit at the consensus layer and chunk-migration or query-routing overhead at the MongoDB layer. As $n \to \infty,\ T(n) \to \frac{T_0}{\alpha}$, so the model reflects a hard asymptotic throughput ceiling rather than unbounded linear scaling.

4.3 Latency Model

End-to-end transaction latency is decomposed into a fixed baseline, a routing component that grows logarithmically with shard count (reflecting mongos $\rightarrow$ config-server metadata lookups across a larger cluster topology), and a coordination component that grows sub-linearly with shard count is as follows:

$$L(n) = L_0 + \beta log2(n) + \gamma\sqrt{n}$$

4.4 Load-Balance Fairness Index

Following Jain's fairness formulation [6], the load-balance quality across n shards carrying loads $x_1, x_2, \ldots, x_n$ is as follows:

$$J(n) = \frac{\left(\sum x_i\right)^2}{n\sum x_i^2}, \quad 0 < J(n) \le 1$$

$J(n) = 1$ indicates perfectly even load distribution across shards, values approaching $\frac{1}{n}$ indicate maximal skew. In the numerical analysis, J(n) is evaluated under a mild degradation assumption representing residual hashing skew and in-flight chunk-migration imbalance that grows slowly with cluster size.

4.5 Scalability Performance

To express how much of the theoretical n-fold throughput gain is actually realized, we define scalability performance as the ratio of achieved throughput to optimal linear-scaling throughput as follows:

$$P(n) = \frac{T(n)}{nT_0} = \frac{1}{1+\alpha(n-1)}$$

P(n) = 1 corresponds to optimal linear scaling, and P(n) declines monotonically with n under the model, quantifying diminishing returns from adding shards.

4.6 Weighted-Sum Multi-Objective Optimization

Shard-count selection involves two competing goals: raw throughput T(n) increases with n, while scalability performance P(n) decreases with n. The simplest way to combine competing objectives into a single decision criterion is the weighted-sum method: normalize each objective to a common [0,1] scale and combine them linearly using weights that reflect their relative priority. Similar weighted multi-criteria formulations have been used to balance competing goals in adjacent domains for example, multi-objective optimization for sharding-scheme security in industrial IoT blockchains [14], fuzzy-parameterized weighted objectives in inventory-optimization models [12], and multi-stage optimization pipelines in deep-learning based diagnostic systems [13], underscoring the generality of the approach adopted in this study:

$$F(n) = w_1 \cdot T_{norm}(n) + w_2 \cdot P(n),\ where\ T_{norm}(n) = \frac{T(n)}{T(n_{max})},\ w_1 + w_2 = 1$$

Here w_1 and w_2 are chosen by the system operator: a higher w_1 favors raw throughput and pushes the optimal n* toward the upper end of the tested range, while a higher w_2 favors efficient resource use and pushes n* toward the lower end. The optimal shard count is simply the n that maximizes F(n): n* = arg max F(n) This weighted-sum approach is applied numerically using two illustrative weight scenarios to show how priority choice shifts the recommended shard count.

5. Computational Procedure and Numerical Illustrations

This section provides the step-by-step computational procedure used to produce Tables 1, and Table 2 and Figure 2, and to solve the weighted-sum optimization followed by a complete MATLAB implementation as illustrated in Figure 3.

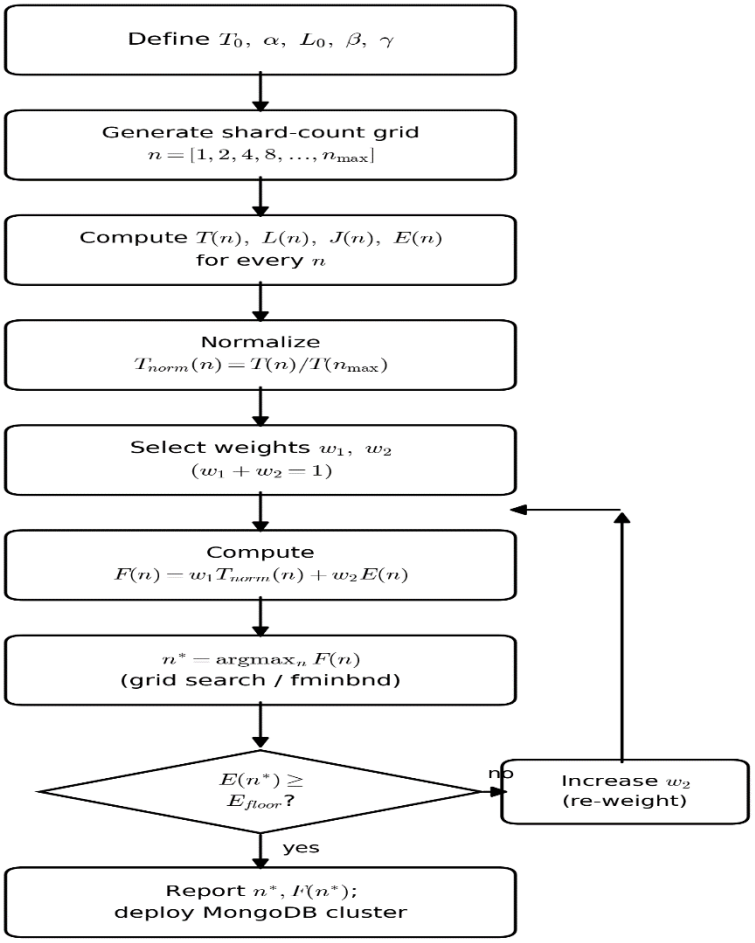

Figure 3: Flowchart of the weighted-sum shard-count optimization procedure.

The following analysis instantiates the model with illustrative parameters representative of a moderate-throughput permissioned blockchain: $T_0 = 1000\ tps,\ \alpha = 0.05,\ L_0 = 10\ ms,\ \beta = 2\ ms,\ and\ \gamma = 0.5$. These values are chosen to demonstrate the qualitative behavior of the model, they should be recalibrated against measured cluster benchmarks before use in a production capacity plan.

5.1. Throughput, Latency, and Fairness vs. Shard Count

Table 1: Computed throughput, latency, fairness index, and scalability performance for n = 1 to 64 shards.

Shards (n)	Throughput T(n) [tps]	Latency L(n) [ms]	Fairness J(n)	Performance P(n)
1	1000.00	10.50	1.000	100.0%
2	1904.76	12.71	0.998	95.2%
4	3478.26	15.00	0.994	87.0%
8	5925.93	17.41	0.986	74.1%
16	9142.86	20.00	0.970	57.1%
32	12549.02	22.83	0.938	39.2%
64	15421.69	26.00	0.874	24.1%

Table 1 shows throughput rising from 1000 tps at n = 1 to 15,421.69 tps at n = 64, a 15.4× gain from a 64× increase in shard count, and the gap between these two figures is precisely the coordination overhead captured by α. Correspondingly, scalability performance P(n) falls from 100% to 24.1%, and latency grows from 10.5 ms to 26.0 ms as routing and coordination costs accumulate.

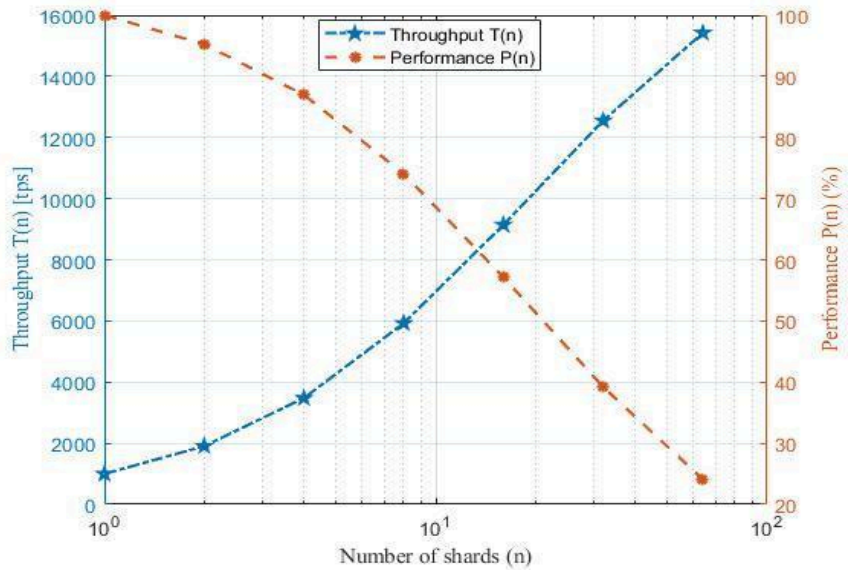

Figure 2: Throughput T(n) and scalability performance P(n) with respect to shard count.

5.2 Marginal Throughput Gain per Shard-Count Doubling

Table 2: Marginal throughput gain and performance for each successive doubling of shard count.

Shard doubling	ΔT(n) [tps]	Gain factor	Marginal Performance
$1 \rightarrow 2$	904.76	1.905×	90.5%

$2 \rightarrow 4$	1573.50	1.826×	82.6%
$4 \rightarrow 8$	2447.67	1.704×	70.4%
$8 \rightarrow 16$	3216.93	1.543×	54.3%
$16 \rightarrow 32$	3406.16	1.372×	37.2%
$32 \rightarrow 64$	2872.67	1.229×	22.9%

Table 2 makes the diminishing-returns pattern explicit: each doubling of shard count contributes a smaller relative throughput gain than the last, falling from a 90.5% gain ($n = 1 \rightarrow 2$) to a 22.9% gain ($n = 32 \rightarrow 64$).

5.3. Weighted-Sum Optimization

Applying the weighted-sum criterion $F(n) = w_1 \cdot T_{norm}(n) + w_2 \cdot P(n)$ to the values in Table 1 $\left(with\ T_{norm}(n) = \frac{T(n)}{T(64)} \right)$ illustrates how the operator's priority weighting determines the recommended shard count are as follows:

i. Throughput-priority weighting $(w_1 = 0.8,\ w_2 = 0.2)$: F(n) rises steadily with n and is maximized at n = 64 (F = 0.848), i.e. the operator should scale out as far as the cluster allows.

ii. Performance-priority weighting $(w_1 = 0.2,\ w_2 = 0.8)$: F(n) is highest at n = 1 (F = 0.813) and falls steadily as n grows, i.e. the operator should keep the shard count minimal.

iii. Balanced weighting $(w_1 = 0.5,\ w_2 = 0.5)$: F(n) increases only marginally across the whole range (0.53 at n = 1 to 0.62 at n = 64), so throughput and performance are nearly offsetting — in this flat-response regime the practical choice is guided by the 70% performance floor from Table 1, which points to n = 8 (P = 74.1%) as the largest shard count before performance drops sharply.

This demonstrates the core value of the weighted-sum method: it converts a two-objective into a single ranked score, and the suggested shard count n* moves predictably between the throughput

favoring and performance favoring optimizes as the weights change. For the balanced case most relevant to general deployments, 8-16 shards remain the practical operating range identified.

5.4. MongoDB Shard Strategy Comparison

Table 3: Qualitative comparison of MongoDB shard.

Strategy	**Shard basis**	**Migration overhead**	**Distribution uniformity**
Range-based sharding	Sequential blockchain state	Low	High
Hash-based sharding	SHA-256 hash of shard	High	Low
Zone sharding	Geo or consensus-zone partitioned shard	Medium	Medium

Table 3 indicates that hash-based sharding minimizes migration overhead and maximizes distribution uniformity, making it the preferred default for the write-heavy, high-cardinality access pattern typical of blockchain account or transaction state, while range-based and zone-based strategies remain preferable when range queries or geographic data locality are architectural requirements, consistent with the auto-sharding balancer behavior studied by Liu et al. [20] and the consistency-performance considerations raised by Dhanagari [21].

7. Results and Discussion

The numerical results support three practical conclusions. First, throughput scaling under MongoDB-backed sharding is well described by a saturating curve rather than a linear one, consistent with the presence of a fixed cross-shard/chunk-migration overhead coefficient α; system designers should therefore benchmark α directly rather than assuming linear scalability when provisioning shard counts. Second, latency grows slowly (logarithmically plus a square-root coordination term) relative to the throughput gains achieved in the low-to-moderate shard-count range, meaning latency is unlikely to be the binding constraint until shard counts become large. Third, scalability efficiency E(n) is the more restrictive constraint in practice: once α and workload skew are fixed, efficiency degrades steadily, and the 70%-efficiency threshold used here identifies 8-16 shards as a reasonable default operating range before deployment-specific tuning. These conclusions are sensitive to the calibration of α, β, and γ, which should be estimated from empirical MongoDB cluster benchmarks rather than assumed. The model is therefore intended as a decision-support framework rather than a universal throughput predictor.

8. Conclusion and Future Work

This study presented a mathematical performance multi-objective optimization model for blockchain sharding backed by MongoDB distributed storage, combining a saturating throughput model, a decomposed latency model, a Jain's fairness index, and a scalability performance metric into a single multi-objective shard-count selection. Numerical illustrations over n=1 to n=64 shards showed throughput gains diminishing from a 90.5% marginal gain at low shard counts to 22.9% at high shard counts, and identified an 8-16 shard practical operating range under an illustrative 70% performance floor.

In future work the proposed study considering empirical calibration of α, β, and γ against a live MongoDB sharded cluster running a permissioned blockchain workload, and extending the fairness index to account for dynamic hotspot formation during chunk migration.

References

[1] D. Bhatnagar and A. Gupta, "Implementation of sharding to improve the performance of blockchain," in 2026 Int. Conf. Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA), IEEE, 2026.

[2] D. Bhatnagar and A. Gupta, "Performance enhancement in blockchain technology through database sharding," in Int. Conf. Information and Communication Technology for Competitive Strategies, Cham: Springer Nature Switzerland, 2025.

[3] A. L. Bulgakov et al., "Scalability and security in blockchain networks: Evaluation of sharding algorithms and prospects for decentralized data storage," Mathematics, vol. 12, no. 23, p. 3860, 2024.

[4] A. Kudzin et al., "Scaling Ethereum 2.0's cross-shard transactions with refined data structures," Cryptography, vol. 6, no. 4, p. 57, 2022.

[5] F. Hashim, K. Shuaib, and N. Zaki, "Sharding for scalable blockchain networks," SN Computer Science, vol. 4, no. 1, p. 2, 2022.

[6] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination," Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA, Rep. 21.1, 1984.

[7] Y. Jo and C. Park, "A hierarchical sharded blockchain balancing performance and availability," arXiv preprint arXiv:2508.14457, 2025.

[8] E. Kokoris-Kogias et al., "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in 2018 IEEE Symp. Security and Privacy (SP), IEEE, 2018.

[9] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in 16th USENIX Symp. Networked Systems Design and Implementation (NSDI 19), 2019.

[10] J. Wu et al., "A sharding blockchain protocol for enhanced scalability and performance optimization through account transaction reconfiguration," Journal of King Saud University – Computer and Information Sciences, vol. 36, no. 8, p. 102184, 2024.

[11] M. Morsidi et al., "A review on blockchain sharding for improving scalability," Future Internet, vol. 17, no. 10, p. 481, 2025.

[12] A. Negi et al., "An inventory model for perishable goods with demand that varies probabilistically with selling price using a pentagonal fuzzy framework," Operations Research Forum, vol. 6, no. 4, Cham: Springer International Publishing, 2025.

[13] L. Arora et al., "Cross-modality transfer learning for breast cancer detection using ultrasound-pretrained deep network on histopathology images," IEEE Access, 2026.

[14] X. Cai et al., "A sharding scheme-based many-objective optimization algorithm for enhancing security in blockchain-enabled industrial internet of things," IEEE Transactions on Industrial Informatics, vol. 17, no. 11, pp. 7650–7658, 2021.

[15] D. Jia et al., "Optimized data storage method for sharding-based blockchain," IEEE Access, vol. 9, pp. 67890–67900, 2021.

[16] C. Liu et al., "Throughput optimization for blockchain system with dynamic sharding," Electronics, vol. 12, no. 24, p. 4915, 2023.

[17] Y. Wang et al., "Dynamic sharding model and performance optimization method for consortium blockchain," The Journal of Supercomputing, vol. 81, no. 2, p. 411, 2025.

[18] S. Katkol, P. M. Dhulavvagol, and S. G. Totad, "Performance optimization in blockchain networks for healthcare systems using adaptive sharding," Procedia Computer Science, vol. 252, pp. 873–882, 2025.

[19] S. Prabakaran and M. Balakrishnan, "An optimal dynamic sharding process in Ethereum blockchain technology for scalability enhancement," Computers and Electrical Engineering, vol. 139, p. 111419, 2026.

[20] Y. Liu, Y. Wang, and Y. Jin, "Research on the improvement of MongoDB Auto-Sharding in cloud environment," in 2012 7th Int. Conf. Computer Science & Education (ICCSE), IEEE, 2012.

[21] M. R. Dhanagari, "MongoDB and data consistency: Bridging the gap between performance and reliability," Journal of Computer Science and Technology Studies, vol. 6, no. 2, pp. 183–198, 2024.